TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Anton Zakatov

Abivahenditega varustatud koduülesannete koostamine Tartu Ülikooli kursusele „Objektorienteeritud programmeerimine”

Bakalaureusetöö (9 EAP)

Juhendaja: Marina Lepp

Tartu 2023

Abivahenditega varustatud koduülesannete koostamine Tartu Ülikooli kursusele „Objektorienteeritud programmeerimine”

Lühikokkuvõte:

Praktika on programmeerimise õppimisel üks tähtsamaid koostisosi. Suurel määral arenevad praktilised oskused just koduülesannete lahendamisel, seda eriti ümberpööratud õppega kursustel. Innustamaks õppureid kodutöid tegema, peavad ülesanded olema aktuaalsed, selged, põnevad ning mõistliku keerukuse ja mahuga. Samuti on tähtis toetada neid õppijaid, kellel tekib raskusi harjutuste sooritamisega. Seega käesoleva lõputöö raames koostati nelja õppenädala jaoks Tartu Ülikooli kursuse „Objektorienteeritud programmeerimine“ tarbeks koduülesandeid, automaatkontrolle ja murelahendajaid. Koostamisel arvestati seotud valdkondade eelnevaid uurimusi. Uute kodutööde ja abivahendite kvaliteedi hindamiseks koguti õppeaine osalejatelt tagasisidet. Kogutud arvamuste järgi võib väita, et üldjuhul jäid üliõpilased loodud didaktiliste materjalide ja vahenditega rahule.

Võtmesõnad: koduülesanded, murelahendaja, automaatkontroll, ümberpööratud õpe, objektorienteeritud programmeerimine

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

Creating homework assignments with supporting tools for the University of Tartu course “Object-Oriented Programming”

Abstract:

Practice is one of the essential components of learning programming. To a large extent, practical skills are developed through solving homework assignments, especially in courses that use a flipped classroom approach. To motivate students to solve homework, exercises should be topical, clear, exciting and with reasonable complexity and quantity. It is also important to support learners who may experience difficulties in completing the exercises. Thus, this thesis focused on creating homework assignments, automatic tests and troubleshooters for four weeks of the course "Object-Oriented Programming" at the University of Tartu. The creation process took into account previous research in related fields. In order to assess the quality of the new homework assignments and supporting tools, course participants’ feedback was collected. Based on the opinions received, students were generally satisfied with the created didactic materials and tools.

Keywords: homework assignments, troubleshooters, automatic tests, flipped classroom, object-oriented programming

CERCS: P175 Informatics, systems theory, S270 Pedagogy and didactics

Sisukord

Sissejuhatus

Tänapäeval on programmeerimise oskus kasulik ja aktuaalne erinevates valdkondades, näiteks tarkvaratehnikas, matemaatikas, andmeteaduses, aga ka teistes. Arvutiprogrammide loomiseks on erinevaid paradigmasid. Üks populaarsemaid on objektorienteeritud programmeerimine: selle õppimiseks on veebipõhiste kursuste platvormil Coursera rohkem kui 150 õppeainet [1]. Vastav kursus „Objektorienteeritud programmeerimine“ on olemas ka Tartu Ülikoolis [2].

Programmeerimise puhul on olulisel kohal praktiliste oskuste arendamine, milleks tavaliselt lahendatakse praktikumide ja koduseid ülesandeid. Kursusel „Objektorienteeritud programmeerimine“ on kasutatud samu koduülesandeid juba mitu aastat [3–6]. Sellega on seotud hulk probleeme. Esiteks on tõenäoline, et varasemate aastate kursuse läbijad koostavad lahenduste avalikke kogumikke, mistõttu võib üliõpilastel väheneda motivatsioon iseseisvalt kodutöid lahendada. Teiseks, kuna ülesanded on üsna vanad, siis nende kontekst[1] ei pruugi enam aktuaalne ja huvitav olla. Kolmandaks oli varasemalt tuvastatud, et mõned kodutööd on kohati ebaselged [7]. Loetletud põhjustel tekkis vajadus uute koduülesannete järele. Lisaks tuleb uuendada ka nendega kaasnevaid abivahendeid. Siinse bakalaureusetöö eesmärk ongi koostada Tartu Ülikooli kursusele „Objektorienteeritud programmeerimine“ uusi koduülesandeid koos automaatkontrollide ja murelahendajatega ning hinnata loodud materjalide ja abivahendite kvaliteeti. Tasub mainida, et selle kursuse jaoks pole varasemalt lõputööde raames kodutöid välja töötatud. Samuti eristub käesolev bakalaureusetöö süstematiseeritud lähenemisega teoreetilisele raamistikule, millel põhineb ülesannete ja abivahendite arendamine.

Varem on ilmunud käesoleva lõputööga seotud valdkondades palju uuringuid, mida töötati läbi kvaliteetsete didaktiliste materjalide ja vahendite koostamiseks. Siinses lõputöös arvestatakse ümberpööratud õppe (kursuse õppemetoodika) erilaadi käsitlustega [8–11]. Uusi ülesandeid koostatakse, olles kursis didaktika valdkonnas toimuvaga: leidub palju uurimusi materjalide loomisest, programmeerimise õppimisest ja kodutööde kontseptsioonist üldisemalt [12–25]. Automaatkontrollide ja murelahendajate väljatöötamiseks võetakse arvesse abivahendite koostamisega seotud teadustöid [7, 15, 26–29].

Lõputöö on jaotatud kuueks peatükiks. Esimeses peatükis tutvustatakse Tartu Ülikooli kursuse „Objektorienteeritud programmeerimine“ olemust ja eripärasid. Teise peatüki eesmärk on määrata teoreetilist raamistikku, millel põhineb lõputöö praktiline osa. Kolmandas peatükis kirjeldatakse koduülesannete koostamist. Automaatkontrollide ja murelahendajate loomise protsessi käsitletakse vastavalt neljandas ja viiendas peatükis. Koostatud ülesannete ja abivahendite tagasisidet analüüsitakse kuuendas peatükis. Töö lõpus on olemas lisad, sealhulgas kodutöö koostamise teoreetilise raamistiku kokkuvõte, uute ülesannete tekstid, lingid loodud abivahenditele ning näide küsimustikust kursuse osalejatelt tagasiside saamiseks.

[1] Siin ja edaspidi on mõeldud ülesande seos mingi valdkonna, nähtuse või objektiga. Näiteks ülesande „Koosta poekassa tööd simuleeriv programm“ kontekst on poeskäik või klienditeenindus.

1. Kursuse „Objektorienteeritud programmeerimine“ kirjeldus

Kvaliteetsete koduülesannete ja abivahendite koostamiseks kursuse „Objektorienteeritud programmeerimine“ jaoks on tähtis arvestada õppeaine olemust ja eripärasid. Seega järgnevalt kirjeldatakse kursusega seotud asjaolusid.

1.1 Kursuse olemus

Tartu Ülikooli õppeinfosüsteemi järgi [2] kursuse eesmärk on anda alusteadmisi objektorienteeritud programmeerimisest ning arendada oskusi programmeerimiskeeles Java programmide koostamiseks. Tegemist on õppeainega, mis kestab 16 nädalat ning annab 6 ainepunkti (EAP). Selle raames käsitletakse mitu teemat, kaasa arvatud mitteabstraktsed ja abstraktsed klassid, liidesed, polümorfism, meetodite ülekatmine, graafika, erindid, vood, dünaamilised andmestruktuurid ja lõimed. Kursuse läbimine on kohustuslik vaid informaatika üliõpilastele, aga mittekohustusliku ainena kuulub ka arvutitehnika, matemaatika ja matemaatilise statistika õppekavadesse. Informaatika puhul tegemist on eeldusainega kursustele „Tarkvaratehnika“, „Algoritmid ja andmestruktuurid“, „Programmeerimine keeles C++“, aga ka teistele.

„Objektorienteeritud programmeerimine“ on sobilik kõigile huvilistele, kellel on varem olnud mingi kokkupuude programmeerimisega: kõik kursuse osalejad peavad olema läbinud eeldusaineid „Programmeerimine“ või „Programmeerimise alused II“. Osalejaid leidub väga erinevatest õppekavadest[2]. Samas, kuna see aine on tihedalt seotud arvutiteaduse instituudi kursustega, siis enamik osalejaid on loodus- ja täppisteaduste valdkonna esimese või teise õppeaasta üliõpilased.

Õppeaine „Objektorienteeritud programmeerimine“ on eristava hindamisega (hinded A, B, C, D, E, F). Kursuse punktisüsteemi [30] järgi võib maksimaalselt saada 102 punkti (välja arvatud lisapunktid). Õppeaine jooksul rakendatakse mitu kontrollivormi, kusjuures iga neist annab teatud punktide arvu [30]. Sealhulgas peab sooritama kaks kontrolltööd (iga eest saab kuni 16 punkti) ja üks eksam (kuni 33 punkti) ning lahendama loengute testid (kokku kuni 12 punkti). Lisaks tuleb näidata oma oskusi kahe rühmatöö raames: nende eest saab teenida kuni 13 punkti. Igal nädalal saab osaleda praktikumil (osalemine annab kuni 6,5 punkti) ja lahendada koduülesandeid (kokku kuni 5,5 punkti).

Kursusel on 11 koduülesannete komplekti[3] [30]. Iga neist on seotud ühe kuni kolme teemaga ning koosneb mitmest ülesandest. Kodutööde lahendamine pole otseselt kohustuslik, kuid võimaldab paremat hinnet saada ning on tähtis osa kursusel kasutatavast õppemetoodikast.

[2] Tartu Ülikooli õppeinfosüsteemi andmete järgi.

[3] Kuigi kursus kestab 16 nädalat, kodutöid ei anta õppenädalatel, mis on eraldatud rühma- ja kontrolltöödele ning õppeaine sissejuhatusele.

1.2 Kursusel kasutatav õppemetoodika

Kursus on üles ehitatud ümberpööratud õppe (ingl *flipped classroom*) metoodika järgi. See tähendab, et kontakttunnid (praktikumid) pole mõeldud eeskätt materjali läbimiseks, vaid iseseisvalt omandatud teadmiste ja oskuste kinnistamiseks. Õppevormi järgi suur osa õppeprotsessist (nii õppimine, kui ka õpiväljundite osaline kontroll) viiakse üle veebikeskkonda [9]. Üliõpilaste ülesanne on läbida õppeteemasid õppejõu poolt välja pakutud videote või kirjanduse toel ning seejärel lahendada koduülesandeid [9]. Selleks kasutatakse kursusel peamiselt kolm keskkonda:

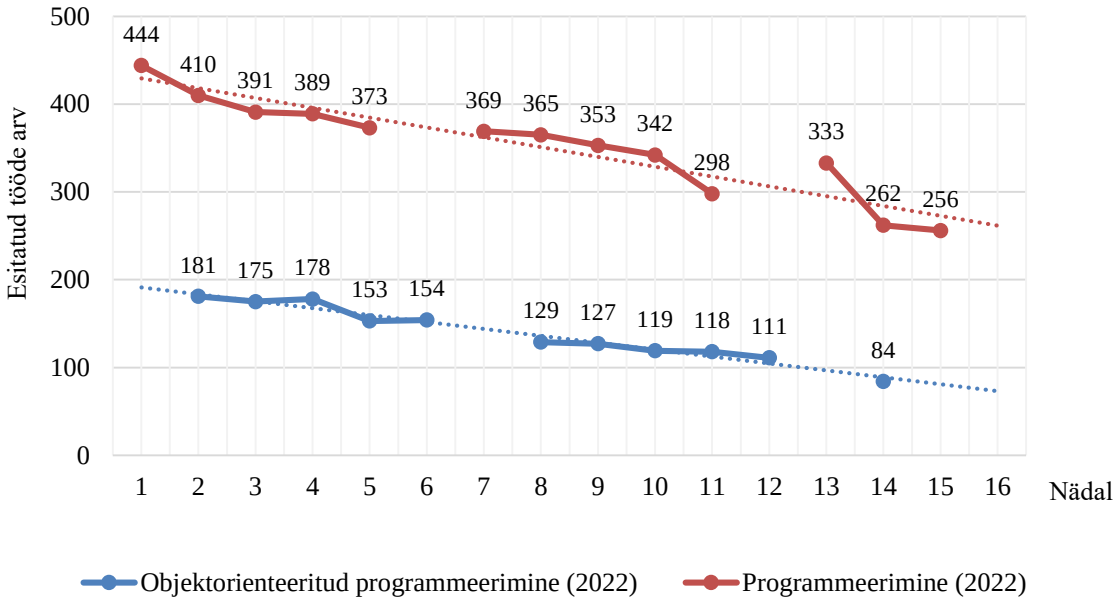
1. õpikeskkond Moodle, läbi mille kursuse osalejad vaatavad videoloenguid, lahendavad teste ning esitavad kodutööde, praktikumide ja kontrolltööde lahendusi. Kasutades õpikeskkonna moodulit Virtual Programming Lab (VPL) lisatakse mõnede ülesannete juurde automaatkontrollid;
2. kursuse veebileht (2023. aastal asus aadressil https://courses.cs.ut.ee/2023/OOP), mis sisaldab korralduslikku infot, juhendeid, praktikumide harjutusi, lugemismaterjale ning kodutöid;
3. murelahendajate keskkond, kust saavad õppurid vajadusel vihjeid kodutööde kohta.

Üldiselt peetakse, et käsitletav metoodika on programmeerimise õppimisel efektiivne nii teadmiste edasiandmisel, kui ka uute oskuste omandamisel [8–11]. Siiski mõnikord tõstetakse esile, et praegustest uurimustulemustest ei pruugi piisata õppevormi tõhususe tõendamiseks [8–10]. Pööratud õppe kasutamisega kaasneb aga mitu riskifaktorit [11]. Esiteks, iga inimene on oma taustaga, mille tõttu uue õppevormiga kohanemine võib võtta palju aega või ei pruugi see olla iga õppuri jaoks sobilik. Mõned eelistavad klassikalist õppetööd[4]. Teiseks eeldab sellise metoodika kasutamine, et õppuril peab õigeaegselt olema võimalus pöörduda õppejõu poole abi ja tagasiside saamiseks [11]. Selle jaoks kasutatakse kursusel e-kirjavahetust, õpikeskkonda Moodle ning vestlusrakendust Zulip. Küsimusi saab esitada nii avalikult (sel juhul näevad küsimust ka teised kursuse osalejad) kui ka privaatselt. Kolmandaks tuleb üliõpilastel omal vastutusel pühendada piisavalt palju aega eeltööle [11]. Siia kuulub ka kodutööde lahendamine.

1.3 Koduülesannete lahendamise aktiivsus

Kuna kodutööd on ümberpööratud õppe metoodika tähtis osa [9], siis on need ka lahutamatu osa õppeainest. Parimal juhul saavad kodutööd mõju avaldada aga vaid siis, kui neid lahendatakse. Hea ülevaate tegelikust olukorrast annab joonis 1.

[4] Ümberpööratud õppe korral õppenädala voog on järgmine: videoloeng → kodutöö lahendamine → praktikum; klassikalise õppetöö puhul: auditoorne loeng → praktikum → kodutöö lahendamine.



Joonis 1. Esitatud kodutööde arv õppenädalate lõikes

Joonise koostamiseks kasutati Tartu Ülikooli kahe programmeerimise kursuse 2022. aasta andmeid[5]. Jooniselt on näha, et keskmiselt esitatud kodutööde arv nädalate kaupa kahaneb. Seda asjaolul, et aines „Programmeerimine“ oli osalejate arv 469 (nendest positiivselt lõpetanuid on 403) ning aines „Objektorienteeritud programmeerimine“ – 241 (positiivselt lõpetanuid on 210). Esitatud lahenduste arvu kahanemise tendentsil võib olla mitu põhjust. On võimalik, et üksikud üliõpilased otsustavad programmeerimise õppimisest loobuda või mõned süstemaatiliselt ei tee kodutöid. Veel üks võimalik põhjus on aga ülesannete kvaliteet: kui ülesanded on igavad, liiga rasked ja töömahukad, siis ei pruugi õppurid olla huvitatud kodutööde lahendamisest. Kindlasti leidub sel tendentsil veel teisigi põhjuseid, seega pole korrektne (vähemalt ilma tõestusteta) väita, et kodutööde headus mõjutab lahendajate arvu kõige rohkem. Oleks aga hea, kui pärast uute koduülesannete koostamist see dünaamika mõnevõrra aeglustuks.

[5] Tartu Ülikooli õpikeskkonna Moodle andmete järgi.

2. Kodutöö koostamise teoreetiline raamistik

Kõrge kvaliteediga ülesannete koostamiseks on tähtis arvestada mitme teguriga. Seega järgnevalt käsitletakse, millele tuleb kodutööde väljatöötamisel lisatähelepanu pöörata. Peatüki tulemustest tehti ka lühikokkuvõte (lisa I), mis lihtsustab ülesannete ja abivahendite loomist.

2.1 Ülesannete keerukuse tasakaal

Darling-Hammondi ja Ifill-Lynch [23] kirjutavad oma artiklis, et kodutööd jäävad tihti esitamata seetõttu, et õppijad ei oska ülesandeid lahendada. Ilmselt nii ongi: kui koduülesanne on liiga keeruline, siis ka õppurite arv, kes on suutelised lahendust lõpuni viia, on väiksem, võrreldes kodutöödega, mille keerukus on paras. Epstein ja Van Voorhis [21] väidavad, et liigselt rasked kodutööd põhjustavad frustratsiooni, mille tõttu ei pruugi üliõpilased enam ülesandeid lahendada. Järelikult ei tohi kodutööd liiga komplitseeritud olla.

Samas ka liigne lihtsus ja sirgjoonelisus ei ole aktsepteeritavad, sest muidu võivad õppuritel tekkida illusioonid tegelike pädevuste ja oskuste kohta. Näiteks ei pruugi enam õppija ainesse tõsiselt suhtuda, kui ta saab hakkama pakutud ülesannetega ilma mingite raskusteta. Vaid läbi mõistlikku väljakutset pakkuvate kodutööde saavad üliõpilased adekvaatset tagasisidet oma oskuste ja teadmiste kohta [24].

Järelikult nõuab ülesannete koostamine keerukusastme tasakaalu säilitamist. Selle tagamiseks peab ülesannete koostaja mõistma, kas ja kuidas on seotud uus harjutus eelnevalt läbitud materjaliga [23]. Head ülesanded nõuavad ka selget struktuuri ja teksti [24]. Kuna tegemist on programmeerimisainega ning üldjuhul üliõpilaste ülesanne seisneb väikeste programmide kirjutamises, siis võib eeldada, et näited programmi tööst teevad ülesande arusaadavamaks. Näiteks Craig jt [18] järgi on näidete puudus üks võimalik põhjus, miks õppijad ei tee kodutööd või miks see tekitab muresid. Veel kaks aspekti, millest sõltub ülesande keerukus, on harjutuse teksti pikkus ja terminoloogia kasutamine [18].

2.2 Ülesannete mahu tasakaal

Samuti peab säilitama ka ülesannete mahu tasakaalu. Ebapiisava mahuga kodutöö korral on võimalik, et see on läbitud materjali suhtes pealiskaudne ning õppijal pole võimalik siduda kõiki uusi teadmisi tegelike probleemide lahendamisega. Kui aga kodutöö on liiga ajanõudlik, siis see võib põhjustada erinevaid tagajärgi. Sellega seotud mõjusid on vaadelnud Galloway jt [25] ning on jõudnud järgmistele arusaamadele. Esiteks mõjub ülekoormatus negatiivselt vaimsele tervisele. Teiseks, suur osa ajast, mis muidu oleks veedetud pere ja sõprade seltsis, kulub hoopis koduülesannete lahendamisele, suurendades sotsiaalprobleemide tekkimise tõenäosust. Kolmandaks väidavad autorid, et mõned õppijad olid sunnitud loobuma hobidest või muutsid kuidagi teisiti oma igapäevaseid tegevusi (näiteks hakkasid vähem magama).

Kirjeldatud efektid leidsid aset juba siis, kui kodutööde lahendamine võttis rohkem aega kui 3 tundi päevas[6] (uuring oli läbi viidud 10.–12. klasside õpilaste seas). Sealjuures samas vanuserühmas kodutööde lahendamise orienteeruv ajakulu võiks olla 1,5 kuni 2,5 tundi päevas [20]. Ülikooli puhul peab aga arvestama, et võrreldes kooli tasemega, oluliselt suuremat osa õppeprotsessist moodustab just iseseisev töö.

Eeltoodust järeldub tarvilikkus hoida koduülesannete mahu tasakaalu. Paulu [24] järgi on kodutöö kvaliteet kvantiteedist tähtsam ning ülesannete maht peab olema selline, et õppejõud oleks suuteline lahenduse järgi otsustada, kas õppija on omandanud vastavaid oskusi või mitte. Samas on kasulik kodutöö mahu ja keerukuse hindamisel saada tagasisidet aine õppejõududelt ja üliõpilastelt, et vajadusel korrigeerida koostatud ülesandeid.

2.3 Seos kodutöö eesmärkide ja õppeaine õpiväljundite vahel

Enne ülesannete koostamist tuleb otsustada, mis on kodutöö eesmärk. Tavaliselt võimaldab kodutöö lahendamine õppuritel teoreetilist materjali praktikas rakendada, valmistuda tunniks ning õppida kirjandust kasutama [24]. Informatsiooni otsimise oskus on programmeerimise puhul eriti tähtsal kohal. Tänapäeval on programmeerimiskeelte võimalused liiga suured, et kõiki mäletada ja teada, selle asemel peab oskama kasutada dokumentatsiooni ja vastavaid allikaid. Iga õppeaine eesmärk seisneb selles, et õppijad omandavad õppimise käigus uusi teadmisi ja/või oskusi. Seega suuremas kontekstis peab kodutöö olema tihedalt seotud õppeaine õpiväljunditega. Rõhutatakse seda ka Paulu raamatus [24]: iga kodutöö peab olema osa millestki suuremast.

Siiski lihtsalt eesmärgi olemasolust ülesandes ei piisa, kui see eesmärk on õppija jaoks lahti seletamata. Kuigi õppejõud tavaliselt teab, miks läbitud materjali kontekstis konkreetne ülesanne tähtis on, ei pruugi õppurite jaoks see nii ilmne olla. Paljud õppijad aga tunnevad vajadust harjutuse eesmärgist arusaamise järele [19, 24]. Järelikult sõltub sellest ka esialgne üliõpilaste suhtumine ülesandesse.

Kui õppeaines on kasutusel ümberpööratud õppe metoodika, siis õppenädala voog on umbes järgmine: videoloengu vaatamine → kodutöö lahendamine → praktikumi läbimine. Selle järgi praktikumi läbimine järgneb koduülesannete lahendamisele. Seetõttu pole võimalik praktikumis anda piisavalt palju infot järgmise kodutöö kohta. Seega tutvustada konkreetse harjutuse eesmärki ja õpiväljundit oleks kõige mõistlikum ülesande sissejuhatuses.

2.4 Motiveerivad ülesanded

Lihtsalt ülesande eesmärgi arusaamisest aga ei piisa. See, et ülesanne on õppeaine kontekstis kasulik, ei pruugi iseenesest lisada motivatsiooni ega huvi kodutöö lahendamiseks. „Kodutöö

[6] Selles ja järgmises lauses kodutööde lahendamise aeg on toodud mitte ühe õppeaine kohta, vaid see on kõikide õppeainete kodutöödele kulutatud aegade summa.

on igav“ – fraas, mida tihti võib kuulda üliõpilastelt. Tegelikult võib igavus põhjustada ka õppetulemuste halvenemist ja soovimatust õpitavas valdkonnas edasi areneda [31].

See, kas ülesanne kohe muutub motiveerivaks, kui see on huvitava kontekstiga, on kehtiv ainult vähesel määral. Teatatakse, et õppijad hea meelega lahendavad ülesandeid, mis on näiteks seotud igapäevase eluga [24]. Kuid Craig jt [18] väidavad, et konteksti olemasolu üldjuhul ei mõjuta üliõpilaste tulemusi. Samuti mainivad nad, et õppurid võivad jätta ülesannete kirjeldused lugemata, kui probleemi kirjeldatakse liiga detailselt. Üks võimalus on koostada ülesandeid ilma kontekstita. See aga ei tähenda, et ülesandesse konteksti lisamine on halb tava. Samas kui kontekst on tähtis, siis peab see olema mõistliku keerukusega (mida käsitletakse käesoleva töö alamosas 2.1) [18].

Üks viis motivatsiooni puudusega võidelda ja suurendada huvi valdkonna vastu on pakkuda ülesandeid, mis pole puhtalt teoreetilised, vaid mille lahendused on rakendatavad igapäevases elus [13, 14]. Selliste puhul veendub õppija, et arendatavad oskused on tõesti kasulikud edasise õppimise ja töötamise jaoks [14]. Selles kontekstis sõltub kodutöö kvaliteet suures osas loengu õppe-eesmärkide ja materjalide aktuaalsusest. Kui loengutes käsitletakse aktuaalseid teemasid (mille abil saab reaalseid probleeme lahendada), siis kodutöö koostaja ülesandeks jääb sobilike probleemide väljamõtlemine, mis võimaldaksid õppenädala õpiväljundeid saavutada.

2.5 Probleemi osandamine

Programmeerimise õppimisel peab pöörama tähelepanu probleemi osandamise[7] oskuse arendamisele. Leidub seisukohti, et selle harjutamine on kodutööde lahendamisel üsna muret-tekitav [17, 32]. Samas väidetakse [32], et ülesande dekomponeerimise õpetamine on väga tähtis ning kõiki teisi programmeerimise oskusi peab arendama probleemi osandamise alusel. Seega võiks programmeerimiskursuste üks eesmärk seisneda just selle oskuse üliõpilastele õpetamises. Üks võimalik lähenemine on ilmutatud kujul näidata, kuidas probleemi alamosadeks jaotada, kuid jätta nende tehniline lahendamine iseseisvaks ülesandeks [32].

Kui järgida sama lähenemist kodutöö teksti koostamisel, siis võib see väljenduda läbi ülesande struktuuri. Näiteks võiks iga harjutuse alguses olla üldine probleemi kirjeldus, millele järgneksid konkreetsed sammud ülesande lahendamiseks. Siinjuures peavad need sammud olema alamülesanded, mis on tekkinud osandamise teel. Iga neist peab olema seotud mingi konkreetse programmi osaga, näiteks „koosta klass `Auto`” või „realiseeri isendimeetod `kasOnElektriauto`”.

[7] IT terministandardi sõnastiku järgi (https://www.eki.ee/dict/its/index.cgi?Q=probleemi+osandamine): probleemi osandamine (või probleemi taandamine) on probleemilahendus, milles kasutatakse operatsioone probleemi dekomponeerimiseks mitmeks alamprobleemiks, mida on tavaliselt algsest probleemist kergem lahendada.

2.6 Tagasiside

Kindlasti üks tähtis osa kodutööde juures on ka õppuritele tagasiside andmine nende lahenduste kohta. Konkreetne ja põhjendatud tagasiside annab õppijale ülevaadet, mis osa õppematerjalist on tal omandatud või omandamata [24]. Elawar ja Corno [33] rõhutavad, et tagasisides tuleb esile tuua lahenduse vigu ja puudujääke just motiveerival viisil: õppijal peab säilima huvi valdkonna ja arenemise vastu ka ebaedu korral. Nende uurimuse järgi üks viis kvaliteetse tagasiside koostamiseks on järgmine. Esiteks peab selgesti väljendama, mis vead tehtud on. Teiseks, puudujääkide olemasolul tuleb ka pakkuda võimalikke lahendusi. Viimaks on tähtis, et tagasisides tuuakse välja ka edukad lahenduse sammud. Uurijad leidsid, et selline lähenemine tagasiside andmisel mõjutab õppimist positiivselt [33].

On kaks tagasiside liiki, mida enamasti kasutatakse programmeerimise kursustel. Esimene on õppejõude kommentaarid, parandused ja hinded. Seega võivad õppejõud anda tagasisidet pakutud mustri järgi. Kuid suure osalejate arvuga kursuste puhul on olulisem teine liik – automaattestid. Nende abil kontrollitakse automaatselt, kas esitatud programm töötab korrektselt teatud sisendandmete korral. Automaatkontroll annab üliõpilasele teada, mis testid on lahendus läbinud edukalt või edutult. Saadud automaattagasiside põhjal jääb üliõpilastel võimalus oma programmi vajadusel parandada ning uuesti esitada. Automaatkontrollid oma olemuselt annavad tagasisidet kiiremini, mis positiivselt mõjutab üliõpilaste kogemusi õppeaine läbimisel [29, 34, 35]. Samuti automaattestid oluliselt lihtsustavad õppejõudude jaoks kodutööde lahenduste hindamist: kui automaatkontrollide kvaliteet on piisavalt kõrge, siis juba sel tasemel tuvastatakse suur osa lahenduse vigadest (kui neid leidub).

Automaatkontrollide arendamine on tihedalt seotud murelahendajate koostamisega [27]. Kuna mõlemad abivahendid on seni osutunud kasulikeks [7, 27, 28, 34, 35], on otstarbekas neid edasi kasutada.

2.7 Murelahendajad

Kuigi automaattestide olemasolu mõnevõrra lihtsustab ülesannete lahendamist, võib õppuril siiski tekkida kodutöö sooritamisega probleeme. Sel juhul üks võimalik abiline on selline veebirakendus nagu murelahendajate keskkond. Murelahendaja on põhimõtteliselt küsimuste kogum, millele eelnevalt on valmistatud ka vastused ehk vihjed kodutöö lahendamiseks.

Lepp jt [27] ja Vihavainen jt [28] rõhutavad, et murelahendajate ja automaatkontrollide rakendamisel on oht abitust õpetada. Selle riski leevendamiseks soovitatakse õppeaines kasutada mõlemat tüüpi ülesandeid: nii abivahenditega kui ka ilma [28]. Ehk on tähtis, et õppuril oleks vajadus ka iseseisvalt oma oskusi ja teadmisi kasutada.

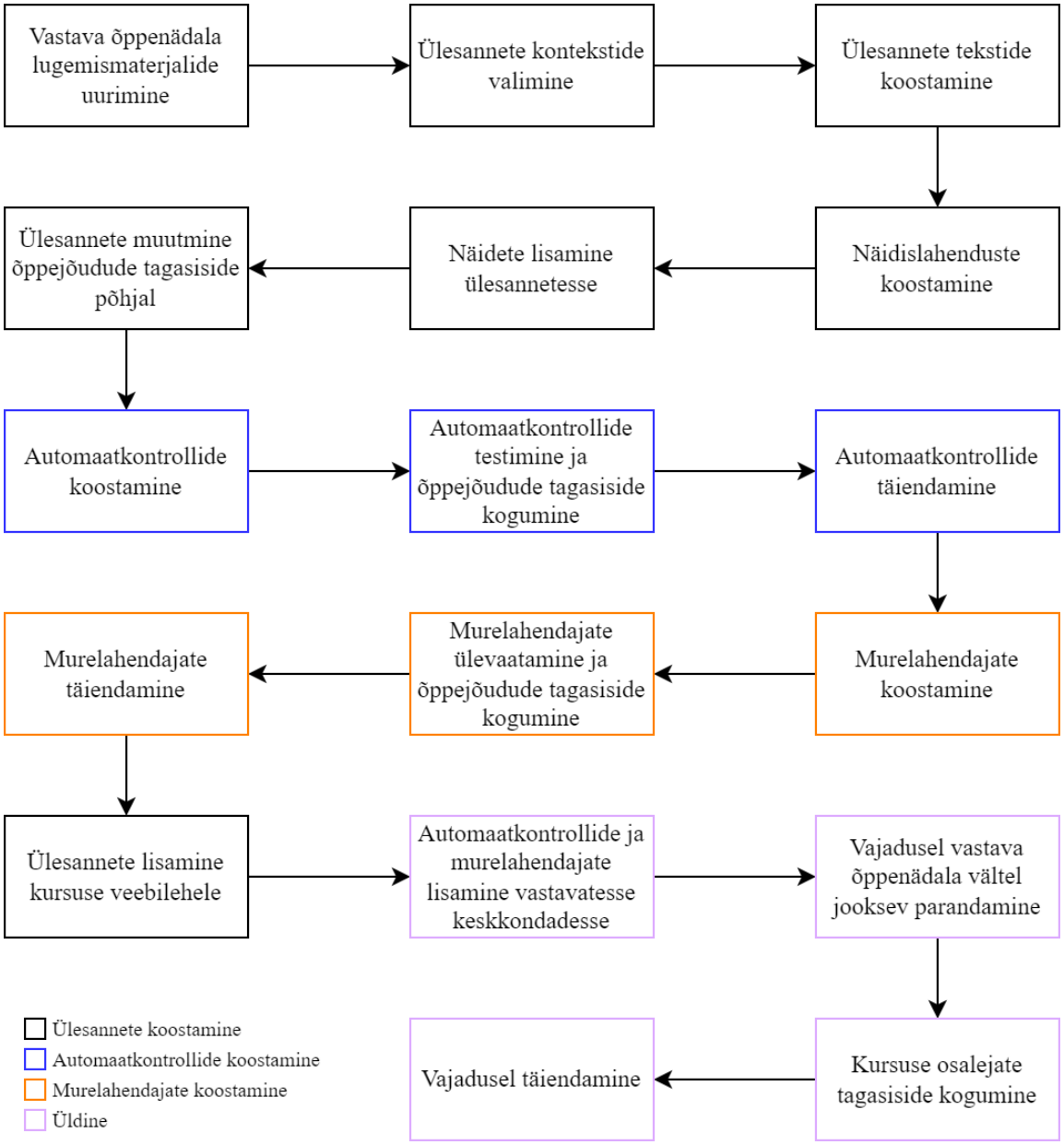
Uute murelahendajate koostamisel tuleb eelkõige ennustada, mis ülesande osad võivad õppijale probleeme tekitada ning võimalike murede jaoks lahendusi pakkuda. Siinjuures koostatud küsimused ja vastused peavad olema piisavalt detailsed ja selged, et murelahendaja oleks kasutaja jaoks kasulik, aga samas ei või valmis lahendust anda.

3. Ülesannete koostamine

Siinse töö üks eesmärk on koostada uusi koduülesandeid. Kuna eelnevalt kirjeldati kodutööde nõudeid, siis järgnevalt käsitletakse koostamise protsessi ja asendatavate harjutuste valimist.

3.1 Ülesannete koostamise protsess

Ülesannete koostamine koosnes mitmest etapist, kusjuures kogu protsessi vältel arvestati kodutöö koostamise teoreetilise raamistikuga. Seda protsessi illustreerib joonis 2.



Joonis 2. Ühe nädala jaoks koduülesannete koostamise protsess

Iga nädala jaoks algas koduülesannete koostamine vastava õppenädala lugemismaterjalide uurimisega, et määrata, mis programmeerimise teemad peavad kodutöös käsitletud olema. Järgmine etapp oli ülesannete jaoks selliste kontekstide valimine, mis osutuksid kursuse

osalejate jaoks huvitavateks (et kodutöö lahendamise motivatsiooni suurendada). Pärast kontekstide valimist koostati harjutuste tekste ja näidislahendusi. Viimaste põhjal lisati teksti näiteid programmi tööst. Järgnevalt koguti kursuse õppejõudude tagasisidet ülesannete kontekstide, selguse, keerukuse, mahu ja teiste aspektide kohta. Saadud kommentaaride põhjal tehti vajadusel ka muudatusi ülesannete tekstides. Kaks järgmist etappi oli automaatkontrollide ja murelahendajate koostamine, mida lähemalt käsitletakse neljandas ja viiendas peatükis. Peale kodutöö ülesannete ja abivahendite valmimist lisati need vastavatesse keskkondadesse. Kui oli vaja, täiendati harjutusi, automaatteste ja murelahendajaid vahetult vastava õppenädala keskel (näiteks kui õppurid leidsid, et automaatkontrollis oli mingi viga sees). Pärast ülesannete lahendamist koguti kursuse osalejate tagasisidet, et hinnata kodutöö ja abivahendite kvaliteeti ning tarbe korral teha muudatusi.

3.2 Asendatavate koduülesannete valimine

Asendatavate ülesannete valimisel lähtuti harjutuste väljavahetamise otstarbekusest: mõned kodutööd oli erinevatel põhjustel mõistlik asendamata jätta. Mõned näited:

- 2. nädala kodutöö on mõeldud suures osas programmeerimise põhikonstruktsioonide kordamiseks, programmeerimiskeelte Python ja Java võrdlemiseks ning arenduskeskkonna tutvustamiseks. Praegused ülesanded ei ole otseselt huvitava kontekstiga, mida aga ei saa negatiivseks asjaoluks pidada. Selle kodutöö peamine eesmärk on tutvustada programmeerimiskeele Java põhikonstruktsioone ning liigne eluline kontekst võib üliõpilasi uue informatsiooniga üle koormata. Olemasolevad ülesanded on kodutöö eesmärgiga kooskõlas ning programmeerimise kogemuse olemasolul on keerukus ja maht tasakaalus.
- 5. nädala kodutöö eesmärk on tutvustada polümorfismi ja liidese mõistet. On tähtis mainida, et igale kodutööle eelneb ka vastava teema teooria (lugemismaterjalid). Selle nädala teoreetilist materjali selgitatakse hoopis näidete põhjal ning kodutöö ülesanded on nendega tihedalt seotud. Ülesannete asendamine tähendaks, et tuleb olulisel määral muuta ka lugemismaterjale, mis aga pole lõputöö eesmärk.

Lisaks sellele on 7. ja 8. õppenädala teema graafika. Graafilise lahendusega ülesannete automaatset kontrollimist on oma magistritöös vaadelnud Eerik Muuli [36]. Üldjuhul selliste ülesannete jaoks automaatkontrollide koostamine on väga komplitseeritud. Pealegi ei oma praegused graafilised harjutused üht kindlat lahendust, vaid ergutavad kujutlusvõimet ja tihti lasevad üliõpilastel iseseisvalt ülesandeid välja mõelda. Seega kursusel nende ülesannete korral automaatkontrolle ei kasutatud. Samadel põhjustel ei koostatud käesoleva lõputöö raames 7. ja 8. õppenädala jaoks uusi ülesandeid ega automaatkontrolle.

Lisatähelepanu ülesannete uuendamisel nõudsid esimeste õppenädalate kodutööd, kus vaadeldakse objektorienteeritud programmeerimise aluseid. On tähtis panna üliõpilaste teadmistele kindla põhja alla, et õppurid saaksid edukalt hakkama ka järgmiste kursuse teemadega. Olulist uuendamist vajas ka 9. nädala kodutöö, milles käsitletakse voogude kasutamine. Selle nädala

teema ja kodutöö on läbi aastate olnud kursuste osalejate arvates keeruline ning segadusttekitav, mistõttu oli mõistlik koostada uusi ülesandeid[8].

Kokku koostati lõputöö raames koduülesandeid 4 õppenädala jaoks (lisa II):

1. 3. nädal: ülesanne 6;
2. 4. nädal: ülesanded 2, 7 ja 8;
3. 6. nädal: ülesanded 5 ja 6;
4. 9. nädal: ülesanded 4 ja 7.

Kõiki ülesandeid lisati kursuse veebilehele ning kursuse vastutav õppejõud saab neid vajadusel redigeerida. Koostatud näidislahendused on samas koodihoidlas, kus asuvad automaat-kontrollide koodifailid. Kursuse vastutaval õppejõul on olemas sellele ligipääs koos redigeerimisõigustega.

[8] Kursuse vastutava õppejõu sõnul.

4. Automaatkontrollide koostamine

Iga koostatud ülesandega kaasneb automaatkontroll. Järgmiselt käsitletakse, kuidas selles lõputöös automaatkontrolle koostati ning mis tööriistu selleks kasutati.

4.1 Tööriistad automaatkontrollide koostamiseks

Kursusel „Objektorienteeritud programmeerimine“ kodutööde lahendused peab esitama läbi õpikeskkonna Moodle, seega ka automaatkontrollid integreeritakse sinna. See on võimalik tänu keskkonna moodulile Virtual Programming Lab (VPL) [37], mida on kasutatud sellel kursusel ka eelnevalt. Käesoleva lõputöö raames koostatud testide kasutuselevõtt seisnes vanade testide uutega väljavahetamises.

Selle kursuse ainus õpetatav programmeerimiskeel on Java, seega ka automaatkontrollid loodi sama keele abil. Automaattestide loomise ja käivitamise lihtsustamiseks eksisteerib raamistik JUnit [38], mille kõige uuem (seisuga 01.03.2023) peaversioon (ingl *major version*) on 5. Kuna aga nende ülesannete puhul, mis jäid uuendamata, on kasutusel JUnit 4, siis ka uute testide koostamiseks kasutati raamistikku peaversiooniga 4. Samuti kursusel kasutatavas jooksutus-serveris on paigaldatud vaid JUnit 4. Lisaks ülaltoodule kasutati teeki VPLJava [39] (põhineb raamistikul JUnit 4), mis lihtsustab nõutud klasside, meetodite ning piiritlejate olemasolu kontrolli.

Automaatkontrollide efektiivsemaks arendamiseks kasutati ehitussüsteemi Gradle, mille abil ühendati kõikide nädalate ülesanded ja testid üheks multimodulaarseks projektiks (ingl *multi-project build*). Ligikaudset struktuuri võib näha joonisel 3. Tänu sellisele arhitektuurile asuvad iga kodutöö näidislahendused ja testid eraldi moodulis, kusjuures taoline korrapärasus muudab kogu projekti hallatavamaks [40].

```
Koodihoidla/
├── gradle/
│   └── // Ehitussüsteemi Gradle funktsioneerimiseks vajalikud failid
├── kodu3/
│   ├── src/
│   │   ├── main/
│   │   │   └── java/
│   │   │       └── // Näidislahedused 3. nädala kodutöö jaoks
│   │   └── test/
│   │       └── java/
│   │           └── // Automaattestid 3. nädala kodutöö jaoks
│   └── build.gradle
├── // Teised kodutööd
├── utlis/
│   └── src/
│       └── main/
│           └── java/
│               └── // Abiklassid ja -meetodid, mida kasutavad automaattestid
├── vpljava/
│   └── // Teegi VPLJava kaustad ja failid
├── build.gradle
├── settings.gradle
├── gradlew
└── gradlew.bat
```

Joonis 3. Ligikaudne projekti struktuur

Tööriista Gradle abil lisati projekti ka raamistik JUnit. Samuti võimaldas projekti struktuur arendada automaatteste vaikepaketis – see on tähtis, sest automaatkontrollide süsteem VPL eeldab, et koodifailid ei sisalda võtmesõna *package* (mis paratamatult tekib, kui tegemist pole vaikepaketiga).

4.2 Automaatkontrollide koostamise protsess

Automaatne kontrollimine teostati kahe etapina. Selline lähenemine on pakutud Vihavainenei jt [28] uuringus ning vaadeldava kursuse raames on seda juba kasutatud. Esimene etapp on eelkontroll, mille käigus selgitatakse välja, kas kõik nõutud programmi osad (näiteks klassid või meetodid) on olemas. Näide eelkontrolli testist on kujutatud joonisel 4.

```
@Test
public void eelkontrollKohv() {
    var cls :Class<?> = checkGetClass(COFFEE_CLASS_NAME);
    assert cls != null;
    checkOnlyPrivateFields(cls);
    checkNoStaticFields(cls);

    CustomTestUtils.checkNotPrivateDeclaredConstructor(
            cls, COFFEE_TYPE_FIELD.type(), COFFEE_PRICE_FIELD.type());
    CustomTestUtils.checkDeclaredFields(cls, COFFEE_TYPE_FIELD, COFFEE_PRICE_FIELD);
    CustomTestUtils.checkGettersExist(cls, COFFEE_TYPE_FIELD, COFFEE_PRICE_FIELD);
    CustomTestUtils.checkNotPrivateDeclaredInstanceMethod(
            cls, COST_OF_CUPS_METHOD_NAME, Double.TYPE, Integer.TYPE);
}
```

Joonis 4. Näide ühest eelkontrolli testist

Eelkontroll on vajalik veendumaks, et järgmise etapi teste on üldse võimalik käivitada: kui mõned kohustuslikud programmi osad on puudu või on valede piiritlejatega, siis ei õnnestu sisulisi teste kompileerida ega käivitada. Seega teise etapi teste käivitatakse vaid siis, kui eelkontrolli tasemel ei tuvastata ühtegi viga.

Teine etapp on vahetult lahenduse kontrollimine. Selle käigus tehakse kindlaks, kas esitatud programmid töötavad korrektselt või mitte. Selleks koostati iga alamülesande jaoks eraldi test. Kontrollis kasutatavaid sisendandmeid oli tavaliselt mitu komplekti. Sisendandmete valimiseks kasutati kaht tehnikat:

1. ekvivalentne jagamine (ingl *equivalence partitioning*), mille järgi sisendandmed moodustavad ekvivalentsiklasse tarkvara komponendi sama (eeldatava) käitumise suhtes. Seejärel võetakse iga sisendparameetri igast klassist vähemalt üks väärtus ja kontrollitakse, kas valitud sisendparameetrite väärtuste puhul komponendi väljund on korrektne [41];
2. piirväärtuste analüüs (ingl *boundary value analysis*), mis tugevdab eelmist tehnikat niimoodi, et sisendandmete hulka lisatakse samuti neid väärtusi, mis asuvad ekvivalentsklasside piiride lähedal või vahetult piiri peal [41].

Nende tehnikate kasutamine võimaldab veenduda, et programmi käitumine on korrektne nii tavaliste kui ka eriliste (aga võimalike) sisendandmete korral. Mõned näited:

- nende ülesannete puhul, mis käsitlevad failist lugemist, testiti programmi käitumist ka tühja faili korral;
- kontrolliti, et programmis oli õigesti kasutatud operaatoreid < *vs.* <= või > *vs.* >=. Seda näiteks 6. nädala kodutöö 5. ülesande lahendamisel: *Meetod peab tagastama 15-kordse või 10-kordse ülemklassi samanimelise meetodi poolt arvutatud väärtuse vastavalt sellele, kas auto* ***on vanem kui 70 aastat või mitte***;
- sõnetöötluse korral kontrolliti, et klassi `String` alamsõne eraldamiseks isendimeetodi `substring(int beginIndex, int endIndex)` kasutamisel oli arvestatud, et `endIndex` on välja arvatud.

Näidet ühest testist võib näha joonisel 5. Sõltuvalt automaattestide läbimise tulemustest anti vastav tagasiside. Näited tagasisidest korrektse ja vigase lahenduse korral on vastavalt joonistel 6 ja 7.

```
@Test
public void ül5_test03_arvutaParanduseMaksumus() {
    CarData.forEach(carData -> hoursStream().forEach(hours -> {
        var expected :double = carData.calculateRepairCost(hours);
        var actual :double = carData.createCar().arvutaParanduseMaksumus(hours);
        if (expected != actual) {
            fail(("Testisin klassi '%s' meetodi '%s' (%s; töötunde=%s)."
                    + " Oodatud väärtus: %s, tegelik: %s")
                            .formatted(carData.carType,
                                    CALCULATE_REPAIR_COST_METHOD_NAME,
                                    carData.fieldsToString(),
                                    hours,
                                    expected,
                                    actual));
        }
    }));
}
```

Joonis 5. Näide ühest sisulisest testist

```
Kommentaarid [-]
  [+]ül5_test01_autoliik ... OK
  [+]ül5_test02_autoToString ... OK
  [+]ül5_test03_arvutaParanduseMaksumus ... OK
  [+]ül5_test04_autoteenindusParanda ... OK
  [+]ül5_test05_autodeParandamine ... OK
```

Joonis 6. Tagasiside korrektse lahenduse korral

```
[+]ül5_test02_autoToString ... OK
[-]ül5_test03_arvutaParanduseMaksumus ... FAIL
Testisin klassi 'Auto' meetodi 'arvutaParanduseMaksumus'
(elektriauto=true; töötunde=1.0). Oodatud väärtus: 36.0,
tegelik: 40.0

[+]ül5_test04_autoteenindusParanda ... FAIL
```

Joonis 7. Tagasiside vigase lahenduse korral

Juhul kui teise etapi automaatkontroll tuvastab vigu, sisaldab tagasiside infot testitava klassi ja meetodi kohta koos testi sisendandmetega. Antakse ka teada, mis on oodatud ja tegelik programmi käitumine. Nii saab lahendaja vigu korrata ja parandada.

Automaatkontrolli koostamisele järgnes selle lisamine õpikeskkonda ja kursuse õppejõududelt tagasiside saamine, mille põhjal tehti automaattestides muudatusi. Vajadusel muudeti teste ka õppurite kommentaaride alusel või vigade korral, mis ilmnesid alles õppeprotsessi käigus[9].

Kokku koostati lõputöö raames automaatkontrolle 10 ülesande jaoks (lisa III), neist 2 on 3. nädala vanade harjutuste jaoks, mida selles lõputöös ei asendatud. Samuti loodi abimeetodite teek, mis võib lihtsustada järgnevate automaattestide kirjutamist. Kõik automaatkontrollid on lisatud õpikeskkonda ning kursuse õppejõud saavad neid vajadusel muuta. Kursuse vastutaval õppejõul on samuti olemas ligipääs privaatsele koodihoidlale, kuhu on kogutud automaat-kontrollid, teek ja ülesannete näidislahendused (koos redigeerimisõigustega).

[9] Tasub rõhutada, et sel kursusel automaatkontrollimisele järgneb lahenduste ülevaade praktikumijuhendajate poolt. See üldiselt leevendab võimalikke negatiivseid mõjusid üliõpilaste hinnetele, tingitud automaattestide (võimalikest) valepositiivsetest või -negatiivsetest tulemustest.

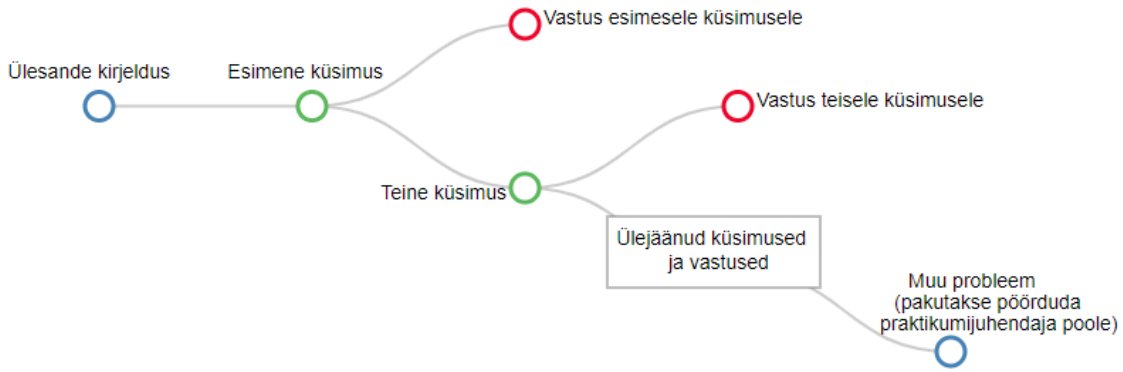
5. Murelahendajate koostamine

Sellel kursusel on 2.–6. õppenädalateks ettevalmistatud iga ülesande jaoks vihjete komplekt ehk murelahendaja (ingl *troubleshooter*). Kuna sellist tüüpi abivahendi kasutamine oli olnud kasulik nii üliõpilaste kui ka õppeaine õppejõudude jaoks [7, 27], siis ülesannete koostamisega kaasnes ka murelahendajate uuendamine.

5.1 Keskkond murelahendajate koostamiseks

Vihjete koostamiseks kasutati murelahendajate keskkonda. Tegemist on veebirakendusega, mis oli loodud Vello Vaherpuu bakalaureusetöö raames [42]. Käesoleva lõputöö kirjutamise hetkel asus keskkond aadressil https://progtugi.cs.ut.ee/.

Iga murelahendaja on tavaliselt seotud ühe kindla ülesandega, mis omakorda koosneb alamülesannetest. Murelahendaja kasutamise käigus küsitakse iga alamülesande kohta, kas see tekitab üliõpilasele muret ning vajadusel proovitakse aidata. Vastavat küsimuste ja vastuste struktuuri võib näha joonisel 8.



Joonis 8. Murelahendaja küsimuste ja vastuste struktuur

Jooniselt on näha, et murelahendaja esimene vaade on lahendatava ülesande kirjeldus. Järgnevad vaated on küsimused ja vastused. Lõpuks, kui küsimused on otsa saanud, soovitab murelahendaja pöörduda praktikumijuhendaja poole abi saamiseks (eeldusel, et lahenduses esineb mingi probleem). Järgnevalt kirjeldatakse, kuidas nägi välja küsimuste ja vastuste koostamine.

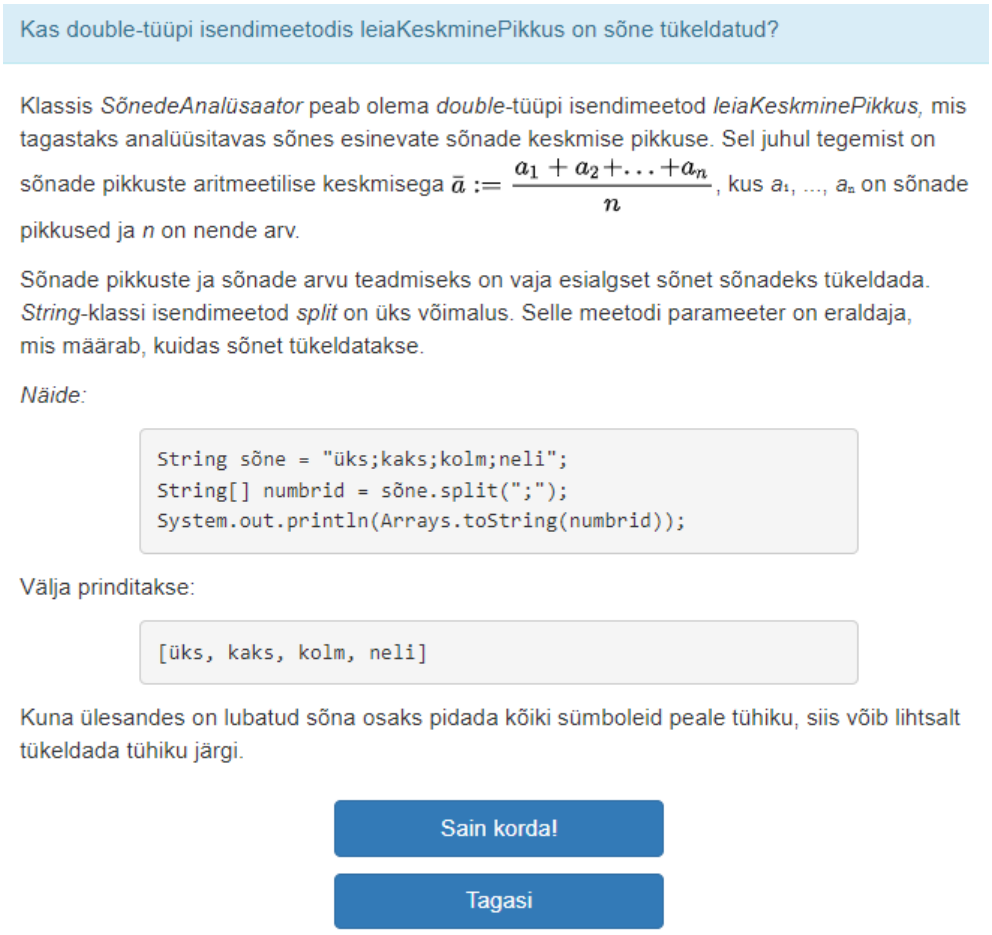
5.2 Murelahendajate koostamise protsess

Murelahendajate koostamise protsess koosnes mitmest etapist.

1. Iga alamülesande puhul prooviti ennustada, mis probleeme võib selle lahendamisel tekkida.
2. Pöörati tähelepanu selle alamülesande jaoks loodud automaattestile koostamaks murelahendaja, mis aitaks lahendada ka neid probleeme, mis said tuvastatud alles automaatkontrolli tasemel.
3. Kui võimalikud mured olid välja selgitatud, koostati iga potentsiaalse probleemi jaoks küsimus (näiteks *Kas* `double`*-tüüpi isendimeetodis* `leiaKeskminePikkus` *on sõne*

tükeldatud?), millele saab murelahendaja kasutamisel vastata eitavalt (*Ei, kuidas seda teha?*) või jaatavalt (*Jah, aga ikka ei tööta*). Esimese vastuse valimisel annab keskkond vihjeid ehk näitab kasutajale selle küsimuse jaoks valmistatud vastust. Vastasel juhul võib kasutaja liikuda järgmise küsimuse juurde.

4. Iga küsimuse jaoks koostati vastus, mis vihjab probleemi võimalikule lahendusele. Siinjuures valmis koodijuppi ei antud, vaid toodi välja näiteid sarnasest ülesandest. Tavaliselt vastus sisaldas mingil määral ka teoreetilist materjali, mis on seotud käsitleva murega. See võimaldab õppuril teooriast paremini aru saada, siduda seda praktikaga ning kasutada neid teadmisi ka teiste ülesannete lahendamisel. Näide ühest murelahendaja vastusest on kujutatud joonisel 9.



Joonis 9. Näide murelahendaja vihjest

5. Pärast vihjete koostamist tehti vajadusel muudatusi kursuse õppejõudude ja osalejate tagasiside põhjal.

Kokku lõputöö raames koostati murelahendajaid 8 ülesande jaoks (lisa IV). Kõiki murelahendajaid lisati vastavasse keskkonda ning kursuse vastutaval õppejõul on olemas nendele ligipääs koos redigeerimisõigustega.

6. Tagasiside

Järgmine etapp seisnes koostatud koduülesannete, automaatkontrollide ja murelahendajate kvaliteedi hindamises. Selleks viidi läbi küsitlus kursuse osalejate seas. Järgnevalt kirjeldatakse küsitluse läbiviimise asjaolusid ning saadud tulemusi.

6.1 Küsitluse läbiviimine

Koostatud kodutöid ja abivahendeid kasutati esmakordselt õppeaines „Objektorienteeritud programmeerimine“ 2022/2023. õppeaasta kevadsemestril. Selle kursuse osalejatelt tagasiside saamiseks koostati kolm küsimustikku platvormil Google Forms (kasutades Tartu Ülikooli Google Workspace teenuseid). Tagasiside andmine polnud kohustuslik, aga vastamise eest oli võimalik õppeaine raames teenida kokku kuni ühe lisapunkti.

Küsimused hõlmasid samu aspekte, mida käsitleti lõputöö peatükis „Kodutöö koostamise teoreetiline raamistik“. Ülesannete ja abivahendite kvaliteeti tuli hinnata Likerti 7-pallisel skaalal. Lisaks oli vaja vastata kodutööle kulutatud aja kohta. Küsimustik sisaldas ka mitte-kohustuslikku küsimust avatud tagasiside (kommentaaride, soovituste ja ettepanekute) jaoks.

Vastused eelmistele küsimustele osaliselt mõjutasid ka edasist küsimustiku läbimist. Näiteks kui vastaja vastas, et ta polnud ülesannet lahendanud, siis selle ülesande kohta küsimusi ei esitatud. Vajadusel esitati aga lisaküsimusi, kui õppija oli suuteline nendele vastata: näiteks murelahendajate kohta, kui üliõpilane oli neid kasutanud (nende kasutamine pole kohustuslik).

Küsimusi esitati erineva detailsusega. Ülesannete kooskõla lugemismaterjalidega, eesmärkide sektsioonide kasulikkuse ja automaatkontrolli kohta tuli vastata ühe nädala lõikes. Siinjuures sisaldasid vastused ka variandi „Ei kehti“ (näiteks juhuks kui vastaja pole vastavaid materjale lugenud või pole automaatkontrolli kasutanud). Kui vastaja oli kasutanud murelahendajaid, siis ka nende kvaliteedi kohta tuli vastata ühe nädala lõikes (ilma variandita „Ei kehti“). Ülesannete keerukuse, mahu, teksti pikkuse, näidete arvu, selguse, põnevuse ja struktuuri kohta pidi vastama iga harjutuse kohta eraldi (ilma variandita „Ei kehti“).

Kui vastaja polnud kodutööd lahendanud, siis tuli valida üht või mitut põhjust. Võimalikud vastused olid „polnud aega“, „polnud motivatsiooni“, „ülesanded tundusid liiga lihtsad“, „ülesanded tundusid liiga keerulised“ ning „muu“ (sel juhul oli vaja põhjust ka täpsustada).

Koostatud koduülesannete ja abivahendite kohta tagasiside kogumiseks loodi kolm küsimustikku (näide ühest neist on lisas V). Iga neist sisaldas küsimusi ühe või kahe nädala kohta ning vastamiseks oli nädal aega. Küsimustike teemad ja vastanute arv olid järgmised:

1. tagasiside 3. ja 4. nädala kohta – 208 vastust ehk 59,6% kursuse osalejatest (tagasiside kogumise hetkel neid oli 349)[10];
2. tagasiside 6. nädala kohta – 164 vastust ehk 48,1% kursuse osalejatest (341)[10];

[10] Tartu Ülikooli õpikeskkonna Moodle andmete järgi.

3. tagasiside 9. nädala kohta – 159 vastust ehk 48,0% kursuse osalejatest (331)[10].

Tagasiside tulemusi iga nädala kohta vaadeldakse allpool eraldi alampeatükkides.

6.2 Tagasiside interpreteerimine

On tähtis täpsustada, kuidas saadud tagasisidet interpreteeriti. Nimelt tulemuste tõlgendamisel järgiti järgmisi põhimõtteid.

- Kuna küsimustikus hinnatav väide „Automaatkontrolli olemasolu tõttu oma lahendusi ise ei testi“ on eitaval kujul, siis analüüs viidi läbi jaatava väite „Automaatkontrolli olemasolul, testin oma lahendusi ka ise“ põhjal (koos vastavate skaala teisendustega).
- Sarnane olukord on ka väitega „Murelahendaja vastustes on liiga palju ette antud“. Analüüsi lihtsustamiseks kasutati väidet „Ei ole nii, et vastustes on liiga palju ette antud“ (koos vastavate skaala teisendustega).
- Tagasiside interpreteerimisel kehtib eeldus, et Likerti tüüpi küsimuste korral on skaala sammud võrdsed. Seda on tähtis eraldi mainida, sest sel viisil saadud andmete puhul mõistete „keskmine” ja „standardhälve” kasutamine ei pruugi alati korrektne olla (Likerti skaala iseloomu tõttu) [43]. Seega lisaks keskmisele ja standardhälbele tuuakse välja ka mediaanväärtus.
- Kodutöö lahendamisele kulutatud aja hindamisel on tähtis arvestada, et vaadeldava kursuse iseseisva töö maht on 92 akadeemilist tundi [2] ehk umbes 6 tundi igal nädalal.

Edasi vaadeldakse küsitluste tulemusi nädalate kaupa.

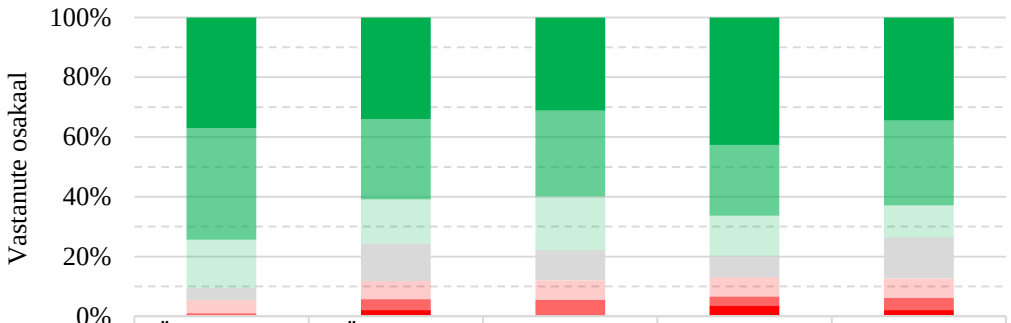
6.3 Tagasiside 3. nädala kohta

Esimese küsitluse tulemuste järgi vähemalt ühe 3. nädala ülesande lahendas (või proovis lahendada) 97,6% küsimustikule vastajatest (203 üliõpilast). Peamised kodutöö mitte-lahendamise põhjused oli aja- ja/või motivatsioonipuudus: mõlemad vastused esinesid 3 korda. Ühele õppurile tundusid ülesanded liiga keerulised, et neid lahendada.

Vastused küsimustele koduülesannete kooskõla lugemismaterjalidega, eesmärkide sektsioonide kasulikkuse ja automaatkontrolli kohta olid suures osas positiivsed (joonis 10). Kõige madalam keskmine hinnang oli 5,5 ning kõige kõrgem – 5,96. Kõigi aspektide mediaanid olid 6.

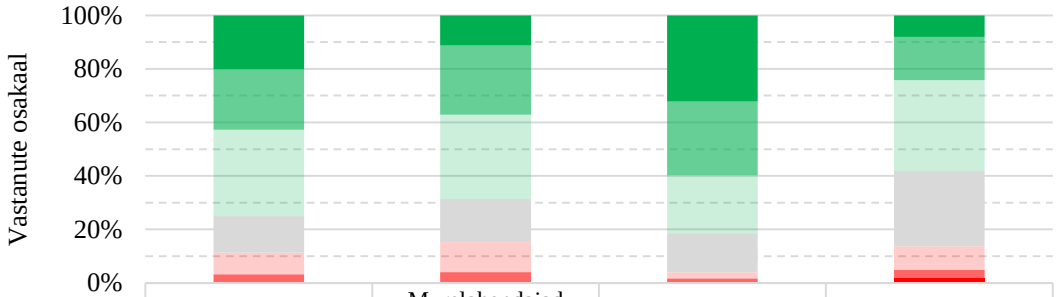
3. nädala kohta küsimustikule vastanute seas oli 124 murelahendaja kasutajat (61,1% vastajatest, kes lahendasid selle nädala kodutöö). Joonisel 11 võib näha, et tagasiside oli pigem positiivne. 81,5% murelahendajate kasutajatest ei arva, et vastustes oli liiga palju ette antud. Murelahendaja ülejäänud aspektide keskmised hinnangud olid alates 4,7 kuni 5,23 ning mediaanväärtused olid 5.

3. nädala kodutöö lahendamisele kulutatud aja osas leiti (joonis 12), et vaid 3 vastajat kulutas lahendamisele rohkem kui 8 tundi. Siiski suuremal osal vastajatest (79,3%) võttis kodutöö tegemine kuni 4 tundi aega ja mediaanväärtus oli 2–4 tundi.



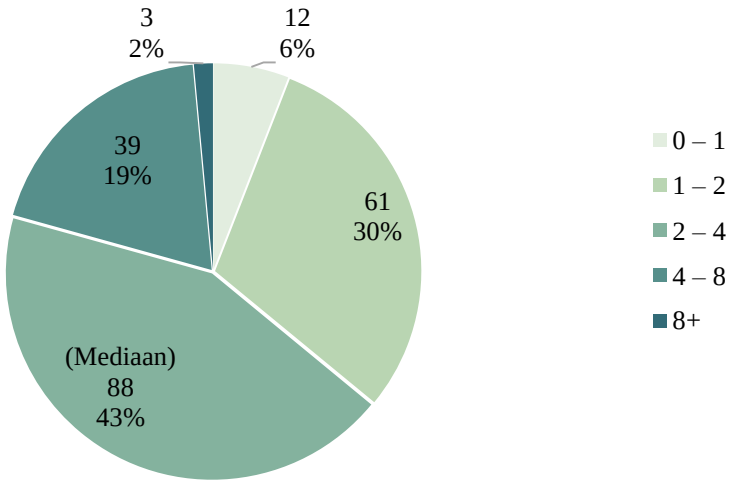
	Ülesanded on lugemis-materjalidega kooskõlas	Ülesannete eesmärkide sektsioonid on kasulikud	Automaat-kontrolli tagasiside on selge	Testin oma lahendusi ka ise	Automaat-kontroll on kasulik
7 (olen täiesti nõus)	75	66	62	85	68
6	76	52	57	47	56
5	33	29	36	27	21
4	8	24	20	14	27
3	9	12	13	13	13
2	2	7	10	6	8
1 (pole üldse nõus)	0	4	1	7	4
Kokku	203	194	199	199	197
Mediaan	6	6	6	6	6
Keskmine hinnang	5,96	5,51	5,51	5,66	5,5
Standardhälve	1,12	1,55	1,47	1,64	1,58

Joonis 10. Tagasiside 3. nädala koduülesannete kooskõla lugemismaterjalidega, eesmärkide sektsioonide kasulikkuse ja automaatkontrolli kohta



	Vaadeldavad probleemid on asjakohased	Murelahendajad aitavad probleemidega toime tulla	Ei ole nii, et vastustes on liiga palju ette antud	Vastused on selged
7 (olen täiesti nõus)	25	14	40	10
6	28	32	34	20
5	40	39	27	42
4	17	20	18	35
3	10	14	3	11
2	4	5	2	4
1 (pole üldse nõus)	0	0	0	2
Kokku	124	124	124	124
Mediaan	5	5	6	5
Keskmine hinnang	5,23	4,98	5,68	4,7
Standardhälve	1,31	1,3	1,22	1,26

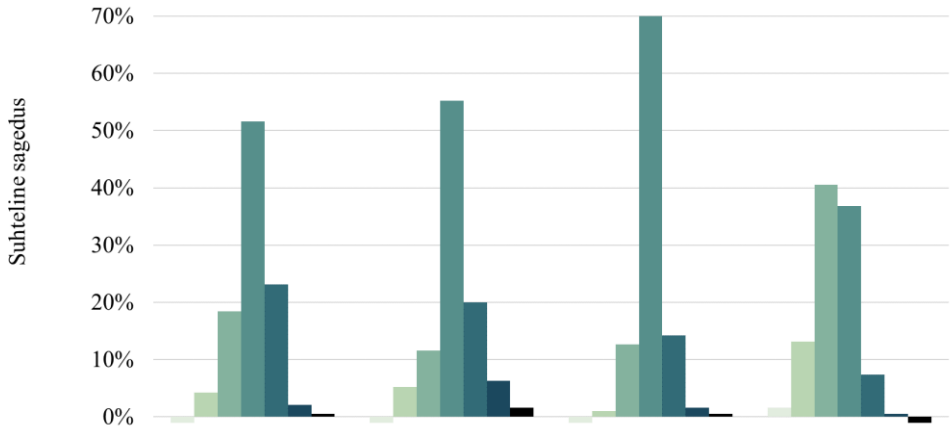
Joonis 11. Tagasiside 3. nädala murelahendajate kohta



Joonis 12. 3. nädala kodutöö lahendamisele kulutatud aeg (tundides)

3. nädala jaoks koostati üks ülesanne. Seda sooritas 93,6% vastajatest, kes olid 3. kodutöö lahendanud (ehk 190 üliõpilast). Mittelahendamise peamised põhjused olid ajapuudus (10 vastust), ülesande keerukus (5 vastust) ja motivatsioonipuudus (2 vastust). Üks vastaja leidis, et harjutuse tekst on liiga pikk, mistõttu jäi tal ülesanne lahendamata.

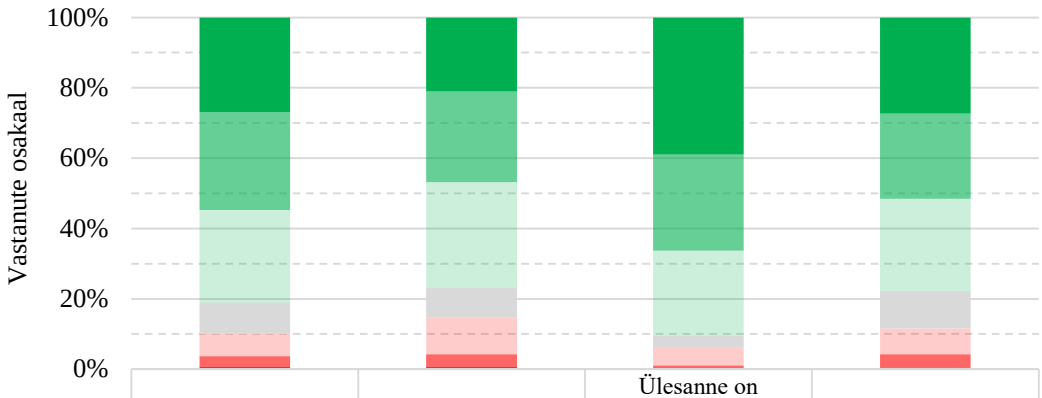
Jooniselt 13 on näha, et keerukus, teksti pikkus ja kulutatud aeg on parajad: keskmised hinnangud olid lähedal arvule 4 (seda arvu võib pidada tasakaalupunktiks). Õppurite arvates oli aga näiteid natuke vähe. Olukorra parandamiseks ülesannet täiendati lisanäidetega.



	Keerukus	Kulutatud aeg	Teksti pikkus	Näiteid
7 (liiga keeruline / palju / pikk)	1	3	1	0
6	4	12	3	1
5	44	38	27	14
4	98	105	133	70
3	35	22	24	77
2	8	10	2	25
1 (liiga lihtne / vähe / lühike)	0	0	0	3
Kokku	190	190	190	190
Mediaan	4	4	4	3
Keskmine hinnang	4,02	4,15	4,04	3,37
Standardhälve	0,85	0,95	0,65	0,88

Joonis 13. Tagasiside 3. nädala 6. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta

Koostatud ülesande selguse ja põnevuse osas on olukord positiivne (joonis 14). Saadud arvamuste järgi on harjutus selge struktuuriga, jaotatud alamülesanneteks mõistlikult ning on huvitav. Väitega „ülesande tekst on üheselt mõistetav“ oli osaliselt või täiesti nõus (vastused „5“–„7“) 76,8% vastajatest, kes olid harjutuse lahendanud.



	Ülesanne on selge struktuuriga	Ülesande tekst on üheselt mõistetav	Ülesanne on jaotatud alam-ülesanneteks mõistlikult	Ülesanne on huvitav
7 (olen täiesti nõus)	51	40	74	52
6	53	49	52	46
5	50	57	46	50
4	17	16	6	20
3	12	20	10	14
2	6	7	2	8
1 (pole üldse nõus)	1	1	0	0
Kokku	190	190	190	190
Mediaan	6	5	6	6
Keskmine hinnang	5,48	5,25	5,88	5,41
Standardhälve	1,35	1,4	1,18	1,4

Joonis 14. Tagasiside 3. nädala 6. ülesande selguse ja põnevuse kohta

Avatud tagasiside võimaldas avastada neid probleeme, mida ettemääratud vastustega küsimuste abil polnud võimalik tuvastada.

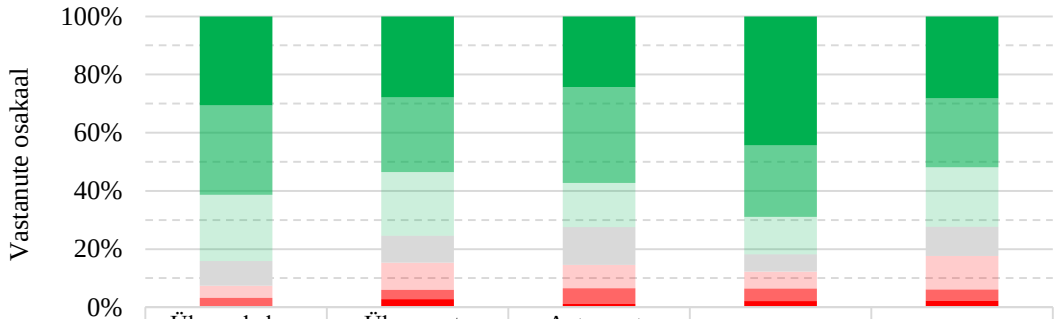
1. Automaatkontroll oli sõnede võrdlemisel liiga range. Need lahendused, mis olid ülesande püstituse järgi korrektsed, ei läbinud automaatkontrolli. Pärast tagasiside tulemuste saamist muudeti automaatkontrolli leebemaks: näiteks teatud meetodite kontrollimisel eeldati varem, et tagastatav sõne sisaldab sõnaühendit *ei joo kohvi*, nüüd piisab vaid fraasist *ei joo*.
2. Mõned vastajad arvasid, et koostatud ülesanne on liiga detailne ning võiks natuke keerulisem olla. Vähem detailsemaks aga seda ei muudetud, sest harjutuse keerulisemaks muutmine tähendaks mahu ja keerukuse tasakaalust väljaminekut.

Üldiselt kinnitas avatud tagasiside eelmiselt käsitletud tulemusi: mõned vastajad leidsid, et kodutöö on liiga mahukas, näiteid programmi tööst on vähe ning murelahendaja ei saanud aidata. Teised aga vastasid, et murelahendaja on suureks abiks, ülesanne on selge ja mõistlik ning kontseptsioon kutsub lahendama.

6.4 Tagasiside 4. nädala kohta

4. nädala ülesandeid lahendas 91,3% küsimustikule vastajatest ehk 190 õppurit. Kodutöö tegemata jätmise populaarne põhjus oli aja- ja/või motivatsioonipuudus (vastused esinevad vastavalt 7 ja 2 korda). Haiguse ja tehnilisemate probleemide (mis polnud seotud ülesannetega) tõttu jäi kodutöö lahendamata 4 üliõpilasel. 3 vastajat leidis, et ülesanded on liiga keerulised.

Hinnang koduülesannete kooskõlale lugemismaterjalidega, eesmärkide sektsioonidele ja automaatkontrollile oli suhteliselt kõrge (joonis 15). Kõikide vaadeldud aspektide mediaanid olid 6. Keskmised hinnangud olid alates 5,27 kuni 5,74.



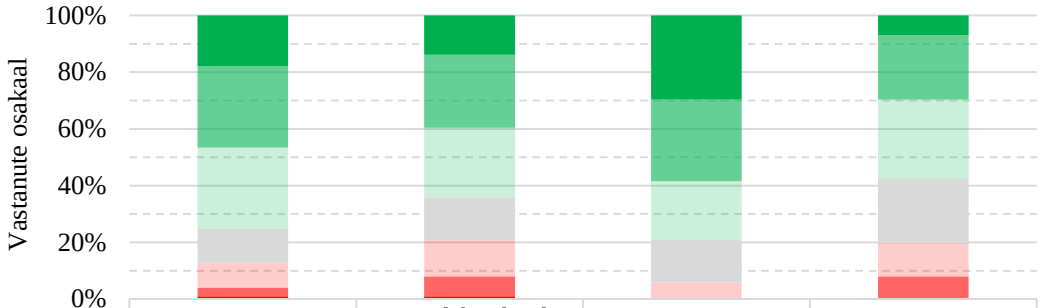
	Ülesanded on lugemis-materjalidega kooskõlas	Ülesannete eesmärkide sektsioonid on kasulikud	Automaat-kontrolli tagasiside on selge	Testin oma lahendusi ka ise	Automaat-kontroll on kasulik
7 (olen täiesti nõus)	58	51	45	83	51
6	58	47	61	46	43
5	43	40	28	24	37
4	16	17	24	11	18
3	8	17	15	11	21
2	6	6	10	8	7
1 (pole üldse nõus)	0	5	2	4	4
Kokku	189	183	185	187	181
Mediaan	6	6	6	6	6
Keskmine hinnang	5,66	5,33	5,32	5,74	5,27
Standardhälve	1,28	1,57	1,52	1,58	1,6

Joonis 15. Tagasiside 4. nädala koduülesannete kooskõla lugemismaterjalidega, eesmärkide sektsioonide kasulikkuse ja automaatkontrolli kohta

Hinnangud murelahendajate jaoks varieerusid natuke rohkem, mida on näha jooniselt 16. Iga väitega oli osaliselt või täiesti nõus 57,4% kuni 79,2% vastajatest. Kõige madalama keskmise hinnangu sai väide „vastused on selged", kuid enamus vastajatest nõustus sellega. Siiski vaadati murelahendajat üle veendumaks, et seal pole segaseid sõnastusi ega puudulikke selgitusi. Võib eeldada, et kõige rohkem probleeme tekkis üliõpilastel failist lugemisega (8. ülesande lahendamisel; tagasisidet selle kohta käsitletakse allpool). Mitu vastajat osutas lugemismaterjalide puudujääkidele ning on võimalik, et murelahendaja vihjetest ei piisanud kõikide lünkade likvideerimiseks.

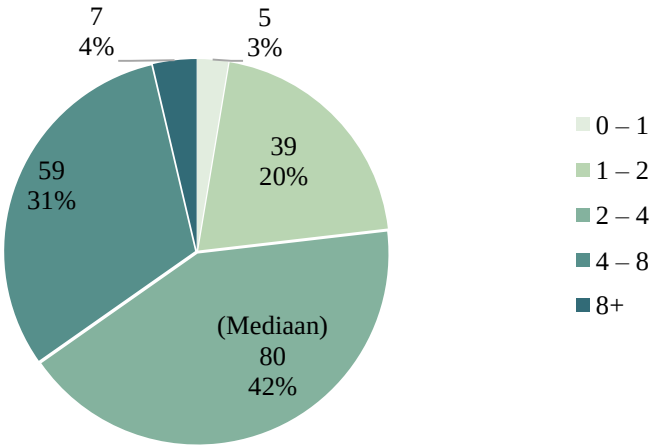
Vaadeldi ka 4. nädala kodutöö lahendamisele kulutatud aega (joonis 17). Rohkem kui 4 tundi kulutas kodutöö lahendamisele 66 vastajat (7 neist väidab, et ülesannete lahendamine võttis

rohkem kui 8 tundi). Enamus (65%) aga sai hakkama selle nädala kodutööga 4 tunniga või vähem. Vaadeldava küsimuse mediaanvastus oli „2–4 tundi“.



	Vaadeldavad probleemid on asjakohased	Murelahendajad aitavad probleemidega toime tulla	Ei ole nii, et vastustes on liiga palju ette antud	Vastused on selged
7 (olen täiesti nõus)	18	14	30	7
6	29	26	29	23
5	29	25	21	28
4	12	15	15	23
3	9	13	6	12
2	3	7	0	8
1 (pole üldse nõus)	1	1	0	0
Kokku	101	101	101	101
Mediaan	5	5	6	5
Keskmine hinnang	5,22	4,88	5,61	4,66
Standardhälve	1,36	1,5	1,22	1,34

Joonis 16. Tagasiside 4. nädala murelahendajate kohta

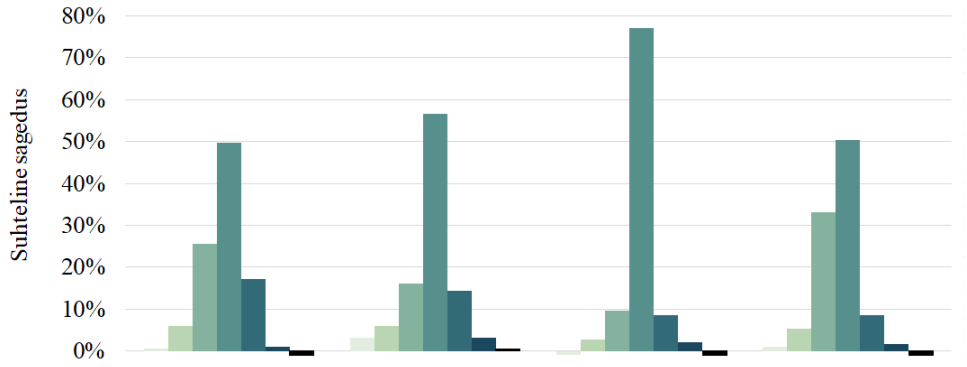


Joonis 17. 4. nädala kodutöö lahendamisele kulutatud aeg (tundides)

4. nädala jaoks koostati 3 ülesannet. Esimese ülesande keerukus, kulutatud aeg, teksti pikkus ja näidete arv on mediaanväärtuste järgi tasakaalus (joonis 18). Samas on näha, et keskmine hinnang näidete arvu jaoks oli 3,65, mistõttu täiendati ülesanne ühe näitega.

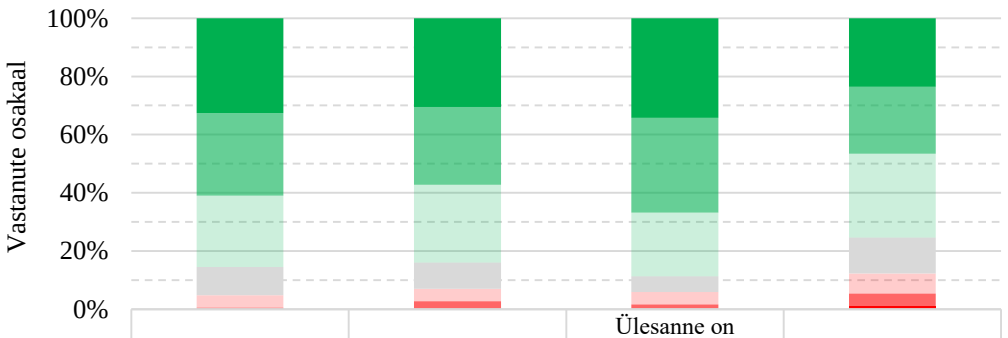
Samuti leiti, et koostatud ülesanne on selge struktuuriga, üheselt mõistetav ning on jaotatud alamülesanneteks mõistlikult (joonis 19): vastavate aspektide mediaanid olid 6. Põnevuse

puhul oli mediaanväärtus aga 5. Samas, vastuseid „5“–„7“ vastava väite kohta oli 141 ehk sellega osaliselt või täiesti nõustus 75,4% vastajatest.



	Keerukus	Kulutatud aeg	Teksti pikkus	Näiteid
7 (liiga keeruline / palju / pikk)	0	1	0	0
6	2	6	4	3
5	32	27	16	16
4	93	106	144	94
3	48	30	18	62
2	11	11	5	10
1 (liiga lihtne / vähe / lühike)	1	6	0	2
Kokku	187	187	187	187
Mediaan	4	4	4	4
Keskmine hinnang	3,8	3,85	3,98	3,65
Standardhälve	0,85	0,99	0,61	0,82

Joonis 18. Tagasiside 4. nädala 2. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta



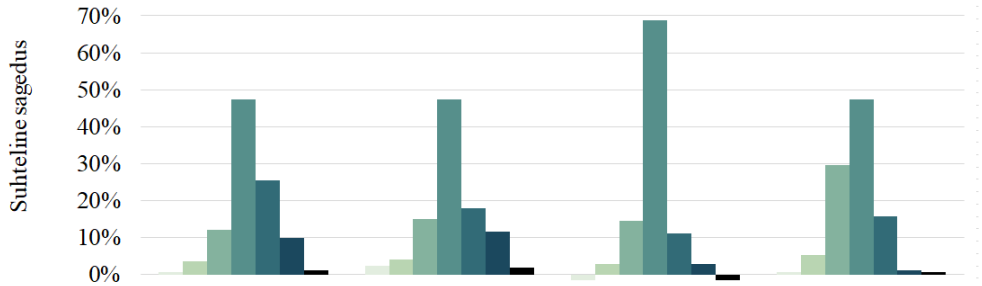
	Ülesanne on selge struktuuriga	Ülesande tekst on üheselt mõistetav	Ülesanne on jaotatud alam-ülesanneteks mõistlikult	Ülesanne on huvitav
7 (olen täiesti nõus)	61	57	64	44
6	53	50	61	43
5	46	50	41	54
4	18	17	10	23
3	8	8	8	13
2	1	5	3	8
1 (pole üldse nõus)	0	0	0	2
Kokku	187	187	187	187
Mediaan	6	6	6	5
Keskmine hinnang	5,74	5,62	5,82	5,27
Standardhälve	1,17	1,27	1,18	1,43

Joonis 19. Tagasiside 4. nädala 2. ülesande selguse ja põnevuse kohta

Kursuse osalejad olid rahul ka 4. nädala 7. ülesandega. Jooniselt 20 on näha, et harjutus pole liiga lihtne ega liiga keeruline. Sama olukord esines ka ülesande lahendamisele kulutatud aja osas. Teksti pikkus ja näidete arv muudatusi ei vaja.

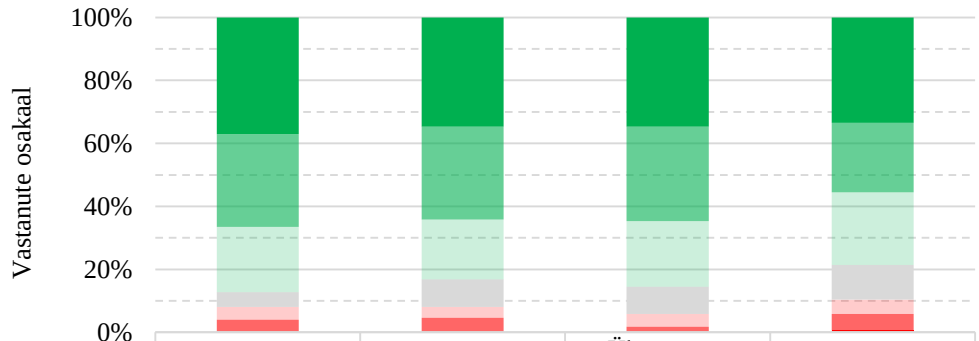
Üliõpilaste tagasiside järgi tegemist on selge ja huvitava ülesandega (joonis 21). Selgusega seotud aspektide keskmised hinnangud varieerusid väiksel määral. Iga väite puhul, mis kirjeldab ülesande selgust, oli vastuste „5“–„7“ osakaal vähemalt 84,0%. Väitega „ülesanne on huvitav“ ei nõustunud osaliselt või täiesti (vastused „1“–„3“) vaid 18 vastajat (10,4%).

Analüüsiti ka selle nädala jaoks koostatud kolmandat ülesannet (joonised 22 ja 23). Tagasisidest selgus, et ülesanne on üsna keeruline. Koos sellega suurenes ka harjutuse lahendamisele kulutatud aeg. Võib eeldada, et seoses suurema keerukusega soovisid üliõpilased näha rohkem näiteid programmi ja selle alamosade tööst. Siinjuures koostatud ülesande eesmärgid on harjutada sõnetöötlust, listide kasutamist ning failist lugemist. Avatud tagasiside järgi võib väita, et kõige rohkem probleeme tekkis just viimasega. Kuna programmeerimise õppimise seisukohalt on tähtis failist lugemise oskust omandada, tehti õppeaine vastutavale õppejõule ettepanek muuta lugemismaterjalide vastavat osa. Vaatamata ülesande keerukusele kursuse osalejad tunnustasid koostatud ülesande selgust ja põnevust (joonis 23).



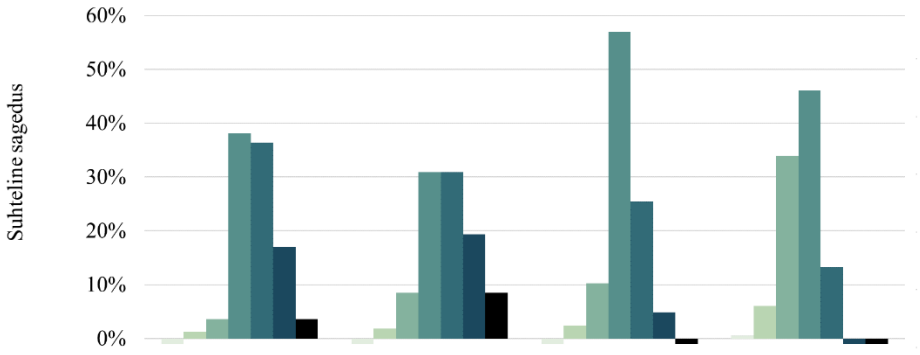
	Keerukus	Kulutatud aeg	Teksti pikkus	Näiteid
7 (liiga keeruline / palju / pikk)	2	3	0	1
6	17	20	5	2
5	44	31	19	27
4	82	82	119	82
3	21	26	25	51
2	6	7	5	9
1 (liiga lihtne / vähe / lühike)	1	4	0	1
Kokku	173	173	173	173
Mediaan	4	4	4	4
Keskmine hinnang	4,28	4,16	3,97	3,78
Standardhälve	0,99	1,14	0,7	0,87

Joonis 20. Tagasiside 4. nädala 7. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta



	Ülesanne on selge struktuuriga	Ülesande tekst on üheselt mõistetav	Ülesanne on jaotatud alam-ülesanneteks mõistlikult	Ülesanne on huvitav
7 (olen täiesti nõus)	64	60	60	58
6	51	51	52	38
5	36	33	36	40
4	8	15	15	19
3	7	6	7	8
2	7	8	3	9
1 (pole üldse nõus)	0	0	0	1
Kokku	173	173	173	173
Mediaan	6	6	6	6
Keskmine hinnang	5,79	5,69	5,77	5,51
Standardhälve	1,31	1,36	1,22	1,47

Joonis 21. Tagasiside 4. nädala 7. ülesande selguse ja põnevuse kohta



	Keerukus	Kulutatud aeg	Teksti pikkus	Näiteid
7 (liiga keeruline / palju / pikk)	6	14	0	0
6	28	32	8	0
5	60	51	42	22
4	63	51	94	76
3	6	14	17	56
2	2	3	4	10
1 (liiga lihtne / vähe / lühike)	0	0	0	1
Kokku	165	165	165	165
Mediaan	5	5	4	4
Keskmine hinnang	4,75	4,83	4,2	3,65
Standardhälve	0,94	1,15	0,78	0,81

Joonis 22. Tagasiside 4. nädala 8. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta

	Ülesanne on selge struktuuriga	Ülesande tekst on üheselt mõistetav	Ülesanne on jaotatud alam-ülesanneteks mõistlikult	Ülesanne on huvitav
7 (olen täiesti nõus)	53	35	56	52
6	39	49	48	44
5	44	49	37	32
4	11	12	13	17
3	10	9	5	13
2	7	9	5	5
1 (pole üldse nõus)	1	2	1	2
Kokku	165	165	165	165
Mediaan	6	6	6	6
Keskmine hinnang	5,54	5,33	5,72	5,5
Standardhälve	1,42	1,42	1,31	1,47

Joonis 23. Tagasiside 4. nädala 8. ülesande selguse ja põnevuse kohta

Avatud tagasiside 3. ja 4. nädala kohta oli mingil määral sarnane. Seega selgus ka siin, et automaatkontroll oli mõnikord liiga range, mistõttu mõnede automaattestide tulemused olid valenegatiivsed[11]. Probleemi kõrvaldamiseks kasutati sama lähenemist, mis 3. nädala puhul. Mitu vastajat osutas failist lugemise keerukusele. Positiivsete kommentaaride järgi on ülesanded aga selgelt sõnastatud, huvitavad ning panevad parajalt proovile.

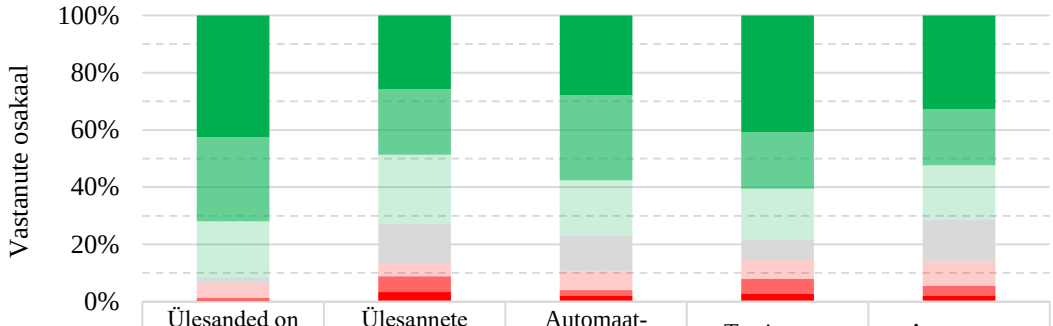
6.5 Tagasiside 6. nädala kohta

Tagasisidest 6. nädala kohta selgus, et vastava kodutöö oli lahendanud 153 üliõpilast, kes küsimustikule vastas, ehk 93,3%. Ülesannete lahendamata jätmise põhjused olid erinevad, näiteks puudus lahendamise motivatsioon (4 vastust) või tundus kodutöö liiga keeruline (3 vastust). Kuid kõige levinum põhjus oli ajapuudus (10 vastust), kusjuures vastajad aga ei täpsustanud, millest see tingitud on. Ühest küljest võib see tähendada, et ülesanded on liiga töömahukad. Teisest küljest on võimalik, et kodutöö lahendamine jäeti viimasele hetkele, mistõttu ei jõutud kodutöö lahendamiseni. Näiteks õpikeskkonna Moodle järgi 6. kodutöö

[11] Sama probleem esineb ka 3. nädala automaatkontrollis, kuid 4. nädala automaatkontrolli koostamise hetkel polnud see puudujääk veel teada.

korral lahendajaid oli 244, kus 126 neist (51,6%) esitas oma lahendusi tähtajaga samal päeval ning 37 (15,2%) veel hiljem[12].

Jooniselt 24 on näha, et ülesanded on lugemismaterjalidega kooskõlas. Selle nädala kodutöö on kõrge kvaliteediga ka eesmärkide sektsioonide kasulikkuse ja automaatkontrolli osas. Automaattestid on selged ja kasulikud, kusjuures ei õpeta abitust: tavaliselt testivad õppurid oma lahendusi ka ise.



	Ülesanded on lugemis-materjalidega kooskõlas	Ülesannete eesmärkide sektsioonid on kasulikud	Automaat-kontrolli tagasiside on selge	Testin oma lahendusi ka ise	Automaat-kontroll on kasulik
7 (olen täiesti nõus)	65	38	42	62	48
6	45	34	45	30	29
5	30	36	29	27	28
4	3	20	19	11	21
3	8	7	10	10	13
2	1	8	3	8	5
1 (pole üldse nõus)	1	5	3	4	3
Kokku	153	148	151	152	147
Mediaan	6	5	6	6	6
Keskmine hinnang	5,97	5,22	5,46	5,55	5,35
Standardhälve	1,2	1,6	1,45	1,66	1,59

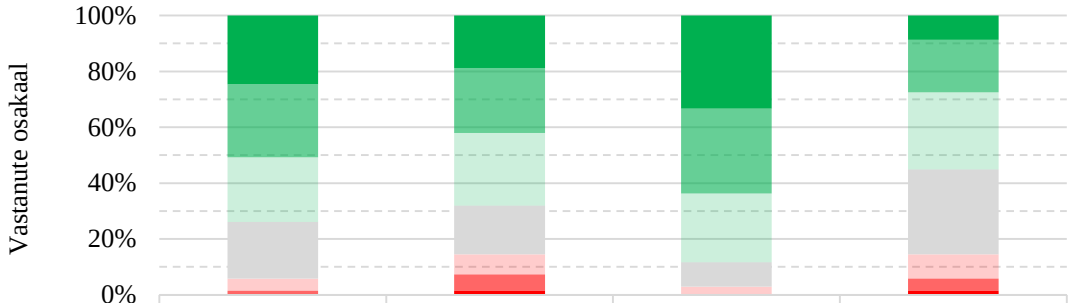
Joonis 24. Tagasiside 6. nädala koduülesannete kooskõla lugemismaterjalidega, eesmärkide sektsioonide kasulikkuse ja automaatkontrolli kohta

Vastajate hulgast oli 69 murelahendaja kasutajat ehk 45,1% õppuritest, kes jättis tagasiside ja lahendas 6. nädala kodutöö. Joonise 25 järgi võib väita, et vaadeldavad probleemid on asjakohased. Samuti õppurid ei arva, et vastustes on liiga palju ette antud. Tasub eraldi vaadelda murelahendajate selgust. On näha, et vastava küsimusega seotud vastuste „1“–„4“ osakaal oli 45,0%. Ühest küljest võib see tähendada, et just koostatud vihjetes leidub puudujääke. Kuigi avatud tagasisides kommentaare murelahendajate kohta polnud, vaadati vihjed üle, et proovida potentsiaalseid ebaselgeid kohti parandada. Teisest küljest on võimalik, et tulemusi mõjutas selle nädala ülesanne (koos murelahendajaga), mis käesoleva lõputöö raames ei uuendatud. Tegemist on 6. nädala 4. ülesandega, kus lahendajal tuli ülesannet iseseisvalt välja mõelda ning lahendada, kasutades objektorienteeritud programmeerimise

[12] Tasub mainida, et kursusel on lubatud oma lahendusi täiendada ja uuesti esitada ühe päeva jooksul pärast esitamise tähtaega.

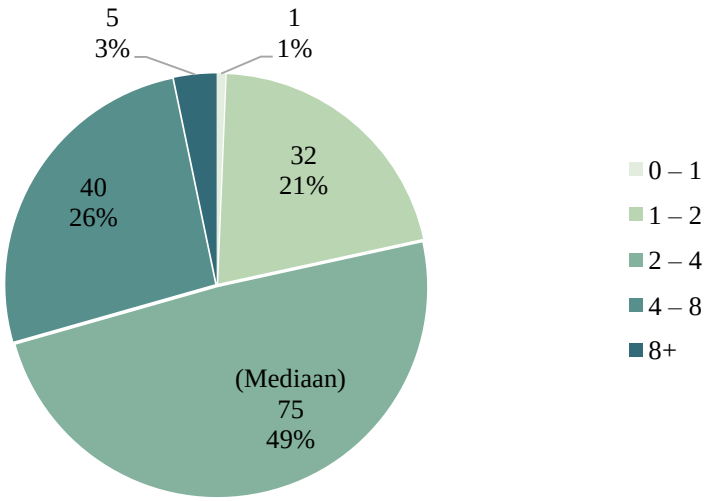
eripärasid. Kursuse õppejõudude sõnul tegu on kasuliku ja vajaliku ülesandega, mistõttu seda ei asendatud. Avatud tagasisidest selgus aga, et ülesanne jättis õppuritele pigem negatiivse mulje, mis võis väljenduda saadud hinnangus.

Järgmine küsimustiku osa käsitles kodutöö lahendamisele kulutatud aega. Vastava küsimuse tulemused on kujutatud joonisel 26. Sealt on näha, et õppurid kulutasid kodutööle vähem aega, võrreldes 4. nädalaga: 71% vastajatest said ülesannetega hakkama maksimaalselt 4 tunniga.



	Vaadeldavad probleemid on asjakohased	Murelahendajad aitavad probleemidega toime tulla	Ei ole nii, et vastustes on liiga palju ette antud	Vastused on selged
7 (olen täiesti nõus)	17	13	23	6
6	18	16	21	13
5	16	18	17	19
4	14	12	6	21
3	3	5	2	6
2	1	4	0	3
1 (pole üldse nõus)	0	1	0	1
Kokku	69	69	69	69
Mediaan	6	5	6	5
Keskmine hinnang	5,42	5,06	5,83	4,7
Standardhälve	1,26	1,48	1,08	1,31

Joonis 25. Tagasiside 6. nädala murelahendajate kohta

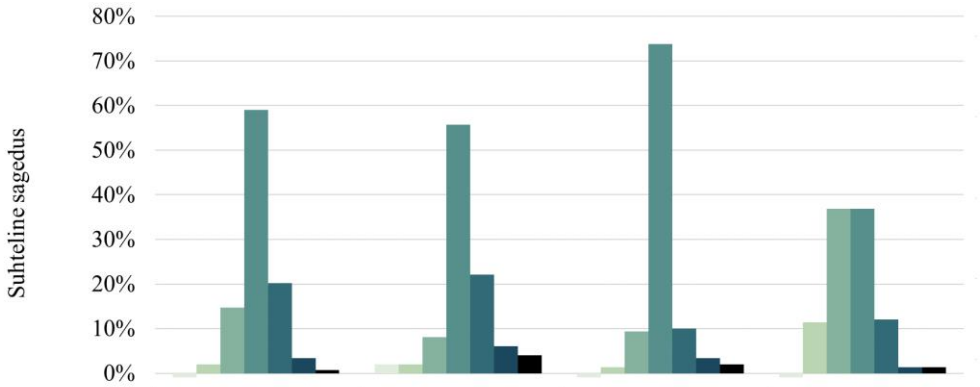


Joonis 26. 6. nädala kodutöö lahendamisele kulutatud aeg (tundides)

6. nädala jaoks koostati 2 ülesannet. Tagasiside esimese ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta on kujutatud joonisel 27. Võib näha, et kõige madalam keskmine hinnang oli 3,59 ning kõige kõrgem – 4,28, kusjuures kõik mediaanväärtused olid 4. Nende väärtuste järgi võib öelda, et hinnatavad kriteeriumid on tasakaalus. Seda arvestades ka hinnangut näidete arvu kohta. Kuigi võib tunduda, et neid oli siiski vähe, tasub mainida, et ülesanne juba sisaldab 3 näidet ning nende arvu suurendamine oleks liigne[13].

Tagasiside ülesande selguse ja põnevuse kohta oli samuti positiivne (joonis 28). Vastajate arvates on koostatud ülesanne selge struktuuriga, üheselt mõistetav ja jaotatud alamülesanneteks mõistlikult. Samuti on see huvitava kontekstiga: vastava väitega on osaliselt või täiesti nõus 79,2% vastajatest.

Tagasiside 6. ülesande kohta eelmisest mõnevõrra erines. Jooniselt 29 on näha, et ülesanne 6 on keerulisem ja aeganõudvam. Selles ülesandes tuli lahendajatel rakendada oma failist lugemise oskust ja on võimalik, et seepärast ülesanne tunduski keerulisem[14]. Teksti pikkus oli paras, aga näiteid oli õppurite arvates veidi vähe. Seetõttu täiendati ülesanne ühe näitega. Siiski jättis see ülesanne aga kursuse osalejatele positiivse mulje selguse ja põnevuse osas, mida on näha jooniselt 30.

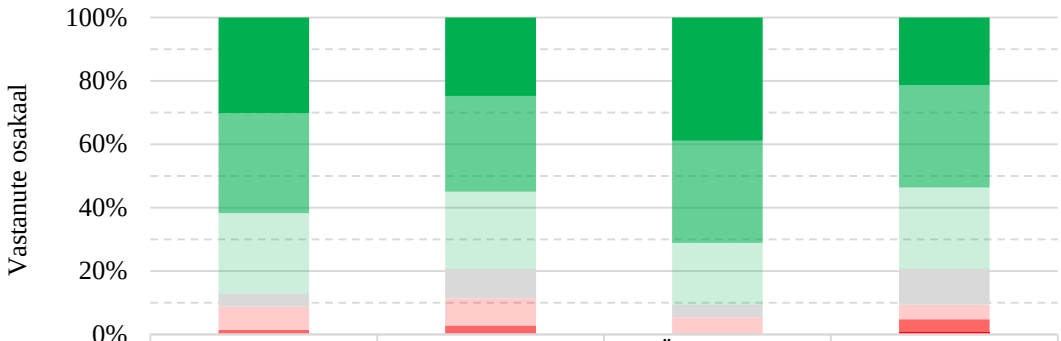


	Keerukus	Kulutatud aeg	Teksti pikkus	Näiteid
7 (liiga keeruline / palju / pikk)	1	6	3	2
6	5	9	5	2
5	30	33	15	18
4	88	83	110	55
3	22	12	14	55
2	3	3	2	17
1 (liiga lihtne / vähe / lühike)	0	3	0	0
Kokku	149	149	149	149
Mediaan	4	4	4	4
Keskmine hinnang	4,1	4,28	4,11	3,59
Standardhälve	0,78	1,04	0,74	0,98

Joonis 27. Tagasiside 6. nädala 5. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta

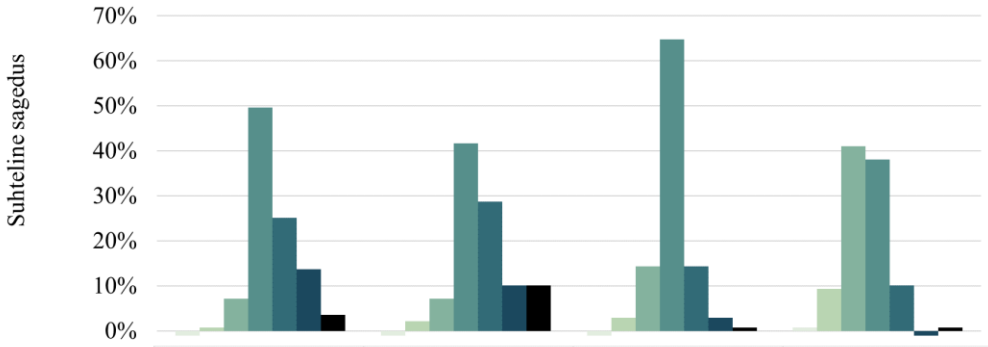
[13] Seda eelkõige kursuse õppejõudude arvates, kuna ülesande tekst muutuks lohakaks ja ülekoormatuks.

[14] Õppurid mainisid juba 4. õppenädalal, et failist lugemise teema on keeruline, aga kuna seda oskust kontrollitakse ka kontrolltööl, siis on mõistlik seda jooksvalt harjutada. Seda enam, et 6. nädala kodutööle järgnebki kontrolltöö.



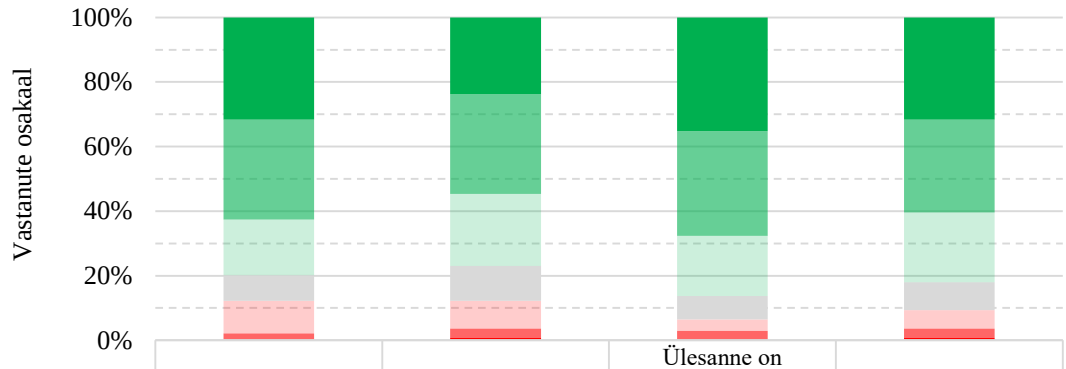
	Ülesanne on selge struktuuriga	Ülesande tekst on üheselt mõistetav	Ülesanne on jaotatud alam-ülesanneteks mõistlikult	Ülesanne on huvitav
7 (olen täiesti nõus)	45	37	58	32
6	47	45	48	48
5	38	36	29	38
4	6	14	6	17
3	11	13	8	7
2	2	4	0	6
1 (pole üldse nõus)	0	0	0	1
Kokku	149	149	149	149
Mediaan	6	6	6	6
Keskmine hinnang	5,69	5,45	5,95	5,4
Standardhälve	1,23	1,33	1,11	1,34

Joonis 28. Tagasiside 6. nädala 5. ülesande selguse ja põnevuse kohta



	Keerukus	Kulutatud aeg	Teksti pikkus	Näiteid
7 (liiga keeruline / palju / pikk)	5	14	1	1
6	19	14	4	0
5	35	40	20	14
4	69	58	90	53
3	10	10	20	57
2	1	3	4	13
1 (liiga lihtne / vähe / lühike)	0	0	0	1
Kokku	139	139	139	139
Mediaan	4	4	4	3
Keskmine hinnang	4,55	4,68	4,02	3,5
Standardhälve	0,96	1,14	0,76	0,88

Joonis 29. Tagasiside 6. nädala 6. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta



	Ülesanne on selge struktuuriga	Ülesande tekst on üheselt mõistetav	Ülesanne on jaotatud alam-ülesanneteks mõistlikult	Ülesanne on huvitav
7 (olen täiesti nõus)	44	33	49	44
6	43	43	45	40
5	24	31	26	30
4	11	15	10	12
3	14	12	5	8
2	3	4	4	4
1 (pole üldse nõus)	0	1	0	1
Kokku	139	139	139	139
Mediaan	6	6	6	6
Keskmine hinnang	5,6	5,39	5,8	5,6
Standardhälve	1,38	1,39	1,25	1,37

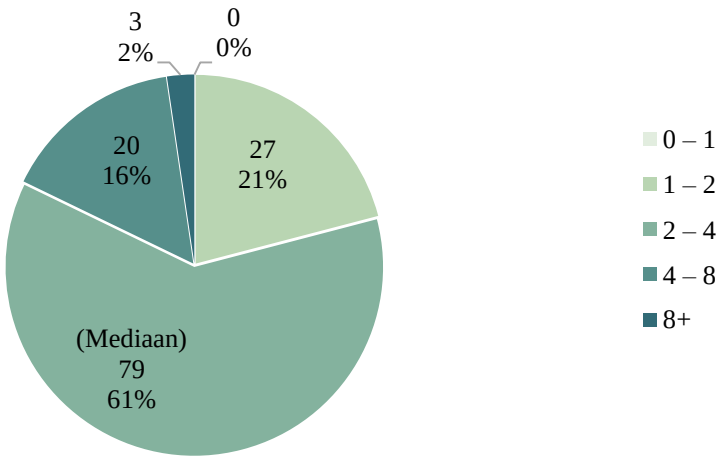
Joonis 30. Tagasiside 6. nädala 6. ülesande selguse ja põnevuse kohta

Küsitluse viimase osa raames koguti avatud tagasisidet. Jäetud kommentaarid kinnitasid, et failist lugemine tekitas probleeme. Lisaks sellele leidus üliõpilasi, kes tahaksid keerulisemaid ülesandeid näha, mis kasutaksid näiteks rekursiooni. Taoline raskendamine oleks aga vastuolus kursuse eesmärgiga anda alusteadmisi objektorienteeritud programmeerimise ja programmeerimiskeele Java kohta. Muidu rõhutasid vastajad, et ülesanded on selgelt sõnastatud, köitvad ning hästi selgitasid päriluse kontseptsiooni[15].

6.6 Tagasiside 9. nädala kohta

Viimase küsitluse eesmärk oli koguda tagasisidet 9. nädala koduülesannete kohta. Nagu oli mainitud peatükis 3, selle nädala teema (vood) on üsna raske ning murettekitav. See mõjutab samuti tagasisidet 9. nädala kohta. Näiteks küsimustikule vastajate seas oli üsna palju neid, kes polnud selle nädala kodutööd lahendanud: harjutused jäid lahendamata 18,9%-l vastajatest. Kolm peamist põhjust oli aja- (25 vastust) ja/või motivatsioonipuudus (11 vastust) ning ülesannete suur keerukus (11 vastust). Kuid need üliõpilased, kes lahendasid koduülesandeid, said enamasti (82% juhtumites) kodutööga hakkama 4 tunniga või kiiremini. Täpsemalt koduülesannete lahendamisele kulutatud aega kirjeldab joonis 31.

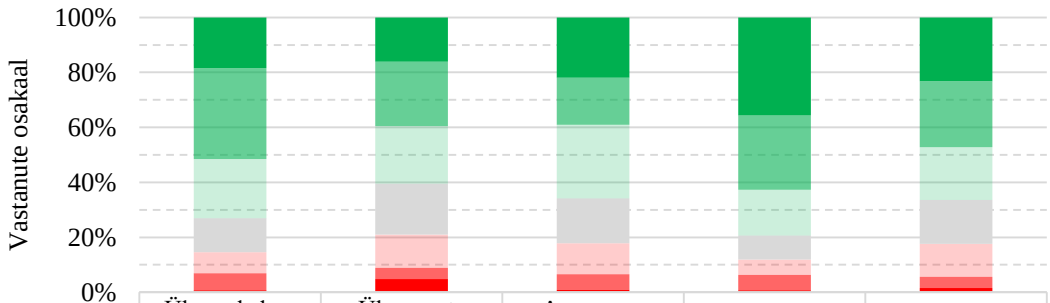
[15] 6. õppenädala teema ongi „Ülemklassid. Alamklassid. Abstraktsed klassid“ [30].



Joonis 31. 9. nädala kodutöö lahendamisele kulutatud aeg (tundides)

Joonisel 32 on kujutatud küsitluse tulemused ülesannete kooskõla lugemismaterjalidega, eesmärkide sektsioonide kasulikkuse ja automaatkontrolli kohta. On näha, et hinnangute mediaanväärtused 9. nädala koduülesannete automaatkontrollide kohta olid võrdsed 5-ga. Avatud tagasiside järgi automaatkontrolli üks probleem seisnes teatud juhtumitel lisakommentaaride puuduses 7. ülesande puhul. Nimelt selles ülesandes tuli üliõpilastel kasutada *while-true* (ehk lõpmatult kestvat) tsüklit. Tsüklist väljumiseks oli vaja keha sees õigel kohal kasutada võtmesõna *break* või *return.* Mõned lahendused aga ei saanud õigel hetkel tsüklit katkestada ning jäid lõpmatu tsüklisse. Automaatkontroll tuvastas neid olukordi, kuid vastav tagasiside koosnes vaid lausest *Java heap error,* mis polnud lahendajate jaoks kõige informatiivsem. Pärast õppuritelt tagasiside saamist täiendati automaatteste lisakommentaaridega, mis viitasid programmi lõpmatu tsükli jäämisele.

Esitati küsimusi ka 9. nädala murelahendajate kohta, tagasiside tulemusi on näha jooniselt 33. Murelahendajaid kasutas 49,6% vastajatest. Joonise järgi kolm esimest väidet sai sarnaseid hinnanguid, mis eelmistel nädalatel. Kuid väite „vastused on selged“ hinnangute mediaanväärtus oli 4. Käsitletava väitega oli osaliselt või täiesti nõus 40,6% vastajatest. Selline tulemus võib suures osas olla seotud just 9. nädala teema keerukusega. Juba eelmistel aastatel on kursuse osalejad maininud, et voogude kasutamine tekitab raskusi. See sai kinnituse ka avatud tagasiside kogumisel. Kuigi koostatud murelahendajad olid kursuste osalejate jaoks kasulikud, leidub probleeme, mida ei saa lahendada vaid vihjete olemasoluga. Kogutud kommentaaridest teavitati ka kursuse vastutavat õppejõudu.



	Ülesanded on lugemis-materjalidega kooskõlas	Ülesannete eesmärkide sektsioonid on kasulikud	Automaat-kontrolli tagasiside on selge	Testin oma lahendusi ka ise	Automaat-kontroll on kasulik
7 (olen täiesti nõus)	24	20	27	45	29
6	43	29	21	34	30
5	28	26	33	21	24
4	16	23	20	11	20
3	10	15	14	7	15
2	8	5	7	7	5
1 (pole üldse nõus)	1	6	1	1	2
Kokku	130	124	123	126	125
Mediaan	6	5	5	6	5
Keskmine hinnang	5,21	4,81	5,02	5,59	5,12
Standardhälve	1,46	1,63	1,52	1,51	1,55

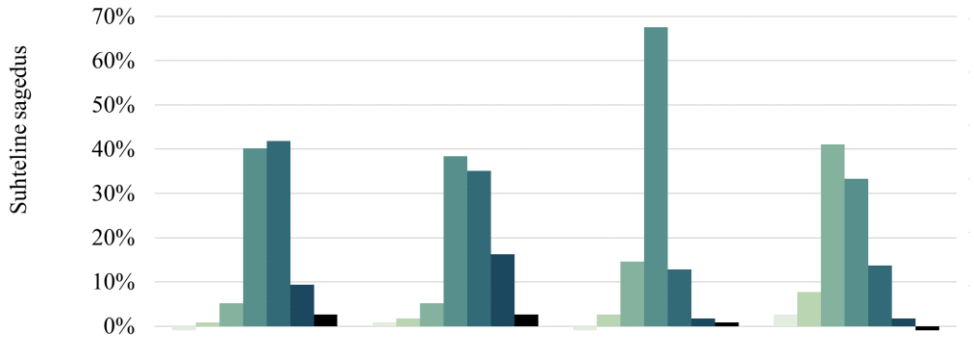
Joonis 32. Tagasiside 9. nädala koduülesannete kooskõla lugemismaterjalidega, eesmärkide sektsioonide kasulikkuse ja automaatkontrolli kohta

	Vaadeldavad probleemid on asjakohased	Murelahendajad aitavad probleemidega toime tulla	Ei ole nii, et vastustes on liiga palju ette antud	Vastused on selged
7 (olen täiesti nõus)	3	1	20	1
6	20	14	23	6
5	28	21	13	19
4	5	13	5	14
3	1	7	3	13
2	5	5	0	9
1 (pole üldse nõus)	2	3	0	2
Kokku	64	64	64	64
Mediaan	5	5	6	4
Keskmine hinnang	4,94	4,41	5,81	3,95
Standardhälve	1,36	1,42	1,1	1,36

Joonis 33. Tagasiside 9. nädala murelahendajate kohta

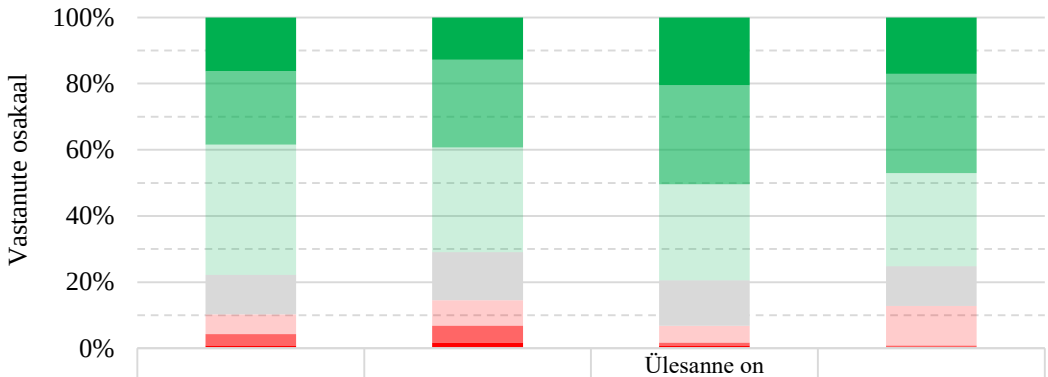
9. nädala jaoks koostati kaks ülesannet. Tagasiside 4. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu on kujutatud joonisel 34. On näha, et keskmine hinnang selle ülesande keerukusele ja lahendamisele kulutatud ajale oli 5 (tasakaalu punktiks peetakse väärtust 4). Kuna voogude teema on kursuse osalejate jaoks keeruline, siis on seda näha ka siin. Teksti pikkus on paras. Kuigi ülesande juures on juba kaks näidet olemas, õppurite arvates võiks neid veel rohkem olla. Lisanäidetega täiendamisel muutuks aga ülesanne ülekoormatuks, mistõttu seda ei tehtud. Pole näha, et ülesanne vajaks täiendusi ka põnevuse ja selguse osas (joonis 35). Tagasiside oli positiivne: iga väitega osaliselt või täiesti nõustus üle 70% vastajatest.

Tagasiside teise koostatud ülesande kohta oli sarnane (joonised 36 ja 37). Harjutus ei tundu liiga keeruline, arvestades käsitletava õppenädala temaatikat. Siinjuures ka näiteid oli lahendajate arvates piisavalt palju. Üldiselt tegemist on selge ülesandega. Hinnangud „5“–„7“ moodustavad vähemalt 70% vastustest harjutuse selguse kohta. Õppurid leidsid, et see kodu-ülesanne on huvitav.



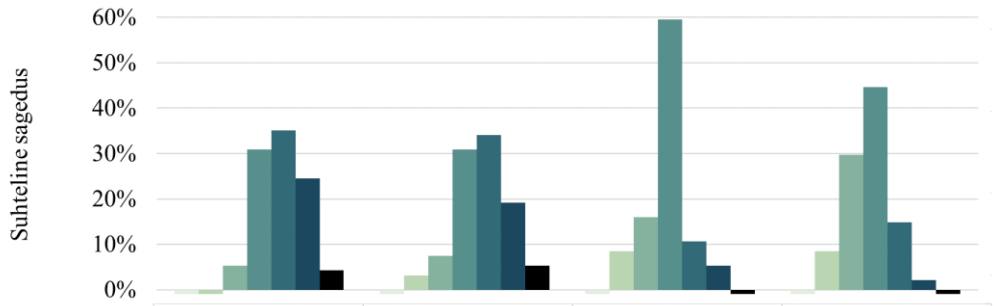
	Keerukus	Kulutatud aeg	Teksti pikkus	Näiteid
7 (liiga keeruline / palju / pikk)	3	3	1	0
6	11	19	2	2
5	49	41	15	16
4	47	45	79	39
3	6	6	17	48
2	1	2	3	9
1 (liiga lihtne / vähe / lühike)	0	1	0	3
Kokku	117	117	117	117
Mediaan	5	5	4	3
Keskmine hinnang	4,62	4,64	3,99	3,53
Standardhälve	0,86	1,01	0,72	0,97

Joonis 34. Tagasiside 9. nädala 4. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta



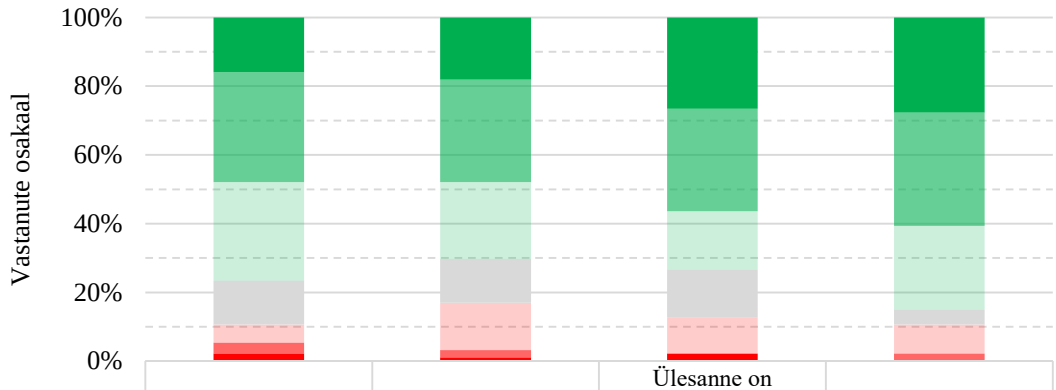
	Ülesanne on selge struktuuriga	Ülesande tekst on üheselt mõistetav	Ülesanne on jaotatud alam-ülesanneteks mõistlikult	Ülesanne on huvitav
7 (olen täiesti nõus)	19	15	24	20
6	26	31	35	35
5	46	37	34	33
4	14	17	16	14
3	7	9	6	14
2	4	6	1	1
1 (pole üldse nõus)	1	2	1	0
Kokku	117	117	117	117
Mediaan	5	5	6	5
Keskmine hinnang	5,17	5	5,41	5,26
Standardhälve	1,28	1,4	1,23	1,26

Joonis 35. Tagasiside 9. nädala 4. ülesande selguse ja põnevuse kohta



	Keerukus	Kulutatud aeg	Teksti pikkus	Näiteid
7 (liiga keeruline / palju / pikk)	4	5	0	0
6	23	18	5	2
5	33	32	10	14
4	29	29	56	42
3	5	7	15	28
2	0	3	8	8
1 (liiga lihtne / vähe / lühike)	0	0	0	0
Kokku	94	94	94	94
Mediaan	5	5	4	4
Keskmine hinnang	4,91	4,74	3,88	3,72
Standardhälve	0,96	1,11	0,9	0,89

Joonis 36. Tagasiside 9. nädala 7. ülesande keerukuse, kulutatud aja, teksti pikkuse ja näidete arvu kohta



	Ülesanne on selge struktuuriga	Ülesande tekst on üheselt mõistetav	Ülesanne on jaotatud alam-ülesanneteks mõistlikult	Ülesanne on huvitav
7 (olen täiesti nõus)	15	17	25	26
6	30	28	28	31
5	27	21	16	23
4	12	12	13	4
3	5	13	10	8
2	3	2	0	2
1 (pole üldse nõus)	2	1	2	0
Kokku	94	94	94	94
Mediaan	5	5	6	6
Keskmine hinnang	5,22	5,15	5,39	5,61
Standardhälve	1,37	1,43	1,45	1,28

Joonis 37. Tagasiside 9. nädala 7. ülesande selguse ja põnevuse kohta

Viimaks oli võimalik jätta avatud tagasiside 9. nädala koduülesannete kohta. Üldiselt kinnitas see, et voogude teema on keerulisem, kui eelmised. Samuti võimaldasid kommentaarid parandada automaatkontrolli ning osutasid lugemismaterjalide puudujääkidele. Lisaks kirjutasid üliõpilased, et selle nädala koduülesanded on huvitava püstituse ja sisuga.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli koostada Tartu Ülikooli kursuse „Objektorienteeritud programmeerimine“ jaoks uusi koduülesandeid koos automaatkontrollide ja murelahendajatega ning hinnata loodud materjalide ja abivahendite kvaliteeti. Lõputöö raames töötati välja 8 koduülesannet, 8 murelahendajat ja 10 automaatkontrolli. Nende kvaliteeti hinnati 2022/2023. õppeaasta kevadsemestri kursuse osalejate tagasiside põhjal, mida koguti kolmeetapilise küsitluse raames. Saadud tulemuste alusel tehti järeldusi didaktiliste materjalide headuse osas ning vajadusel täiendati koduülesannete tekste ja abivahendeid.

Koduülesannete, automaatkontrollide ja murelahendajate välja töötamisel arvestati kursuse olemust ja kasutatavat õppemetoodikat. Samuti loodi varem ilmunud teadustööde alusel teoreetiline raamistik, millel põhineb lõputöö praktiline osa. Didaktiliste materjalide koostamisel võeti arvesse ka eelnevaid praktilisi uuringuid.

Kursuse osalejatelt saadud tagasiside järgi võib väita, et uued koduülesanded on loogilised ja motiveerivad ning nende maht ja keerukus on tasakaalus. Automaatkontrollid olid enamasti kasulikud ja selged. Suureks abiks osutusid ka murelahendajad. Koostatud kodutööde ja abivahendite kvaliteediga jäid rahule ka kursuse praktikumijuhendajad.

Lõputöö tulemustel võib olla mitu rakendust. Esiteks saab loodud teoreetilist raamistikku kasutada ka järgnevate koduülesannete väljatöötamisel. Teiseks võib osutada automaattestide valmistamise käigus arendatud abimeetodite teek kasulikuks järgnevate automaatkontrollide koostamisel. Kindlasti on võimalik küsitluse teel saadud tagasisidet kasutada õppeaine paranemiseks.

Käesoleva bakalaureusetöö puhul leidub mitu edasiarenduse võimalust. Näiteks saab asendada neid koduülesandeid, mille väljavahetamisega kaasneks oluline lugemismaterjalide täiendamine. Selle lõputöö tulemusi võiks arvestada ka graafilise lahendusega uute ülesannete koostamisel, kusjuures automaattestide loomine nõuaks tehnilisi lahendusi programmi staatilise analüüsi või pildituvastuse valdkondadest.

Viidatud kirjandus

[1] Veebipõhiste kursuste platvorm Coursera. Object-oriented programming courses. https://www.coursera.org/search?query=object%20oriented%20programming (22.03.2023)

[2] Tartu Ülikooli õppeinfosüsteem. LTAT.03.003 Objektorienteeritud programmeerimine (6 EAP). https://ois2.ut.ee/#/courses/LTAT.03.003 (12.03.2023)

[3] Tartu Ülikool. Arvutiteaduse instituut. Objektorienteeritud programmeerimine õppeaastal 2018/19. https://courses.cs.ut.ee/2019/OOP (22.03.2023)

[4] Tartu Ülikool. Arvutiteaduse instituut. Objektorienteeritud programmeerimine õppeaastal 2019/20. https://courses.cs.ut.ee/2020/OOP (22.03.2023)

[5] Tartu Ülikool. Arvutiteaduse instituut. Objektorienteeritud programmeerimine õppeaastal 2020/21. https://courses.cs.ut.ee/2021/OOP (22.03.2023)

[6] Tartu Ülikool. Arvutiteaduse instituut. Objektorienteeritud programmeerimine õppeaastal 2021/22. https://courses.cs.ut.ee/2022/OOP (22.03.2023)

[7] Klaanberg A. Murelahendajate koostamine Tartu Ülikooli kursuse „Objektorienteeritud programmeerimine" tarbeks. Tartu Ülikooli arvutiteaduse instituudi bakalaureusetöö, 2020. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69730 (28.01.2023)

[8] Taşpolat A., Özdamli F., Soykan E. Programming language training with the flipped classroom model. *SAGE Open*, 2021, pp. 1–17. https://doi.org/10.1177/21582440211021403

[9] Bishop J., Verleger M. Testing the flipped classroom with model-eliciting activities and video lectures in a mid-level undergraduate engineering course. *2013 IEEE Frontiers in Education Conference (FIE)*, 2013, pp. 161–163. https://doi.org/10.1109/FIE.2013.6684807

[10] Bishop J., Verleger M. The flipped classroom: a survey of the research. *2013 ASEE Annual Conference & Exposition Proceedings*, 2013, pp. 1–18. https://doi.org/10.18260/1-2--22585

[11] Ölmefors O., Scheffel J. High school student perspectives on flipped classroom learning. *Pedagogy, Culture & Society*, 2021, pp. 1–18. https://doi.org/10.1080/14681366.2021.1948444

[12] Luxton-Reilly A., Simon, Albluwi I., Becker B.A., Giannakos M., et al. Introductory programming: a systematic literature review. *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 2018, pp. 55–106. https://doi.org/10.1145/3293881.3295779

[13] Hundley J., Britt W. Engaging students in software development course projects. *The Fifth Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations*, 2009, pp. 87–92. https://doi.org/10.1145/1565799.1565820

[14] Konecki M., Lovrencic S., Kaniski M. Using real projects as motivators in programming education. *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 883–886. https://doi.org/10.1109/MIPRO.2016.7522264

[15] Saarevet T. Koduülesannete ja murelahendajate loomine kursusele „Introduction to Programming". Tartu Ülikooli arvutiteaduse instituudi bakalaureusetöö, 2022. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=74534 (22.03.2023)

[16] Pruunsild P. Tartu Ülikooli kursuse „Programmeerimise alused" uus ülesannete kogu. Tartu Ülikooli arvutiteaduse instituudi bakalaureusetöö, 2022. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=74502 (23.03.2023)

[17] Castro F.E.V., Fisler K. On the interplay between bottom-up and datatype-driven program design. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 205–210. https://doi.org/10.1145/2839509.2844574

[18] Craig M., Smith J., Petersen A. Familiar contexts and the difficulty of programming problems. *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 2017, pp. 123–127. https://doi.org/10.1145/3141880.3141898

[19] Miller L.D., Soh L.-K., Chiriacescu V., Ingraham E., Shell D.F., et al. Integrating computational and creative thinking to improve learning and performance in CS1. *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 475–480. https://doi.org/10.1145/2538862.2538940

[20] Cooper H., Robinson J.C., Patall E.A. Does homework improve academic achievement? A synthesis of research, 1987–2003. *Review of Educational Research*, 2006, pp. 1–62. https://doi.org/10.3102/00346543076001001

[21] Epstein J.L., Van Voorhis F.L. More than minutes: teachers' roles in designing homework. *Educational Psychologist*, 2001, pp. 181–193. https://doi.org/10.1207/S15326985EP3603_4

[22] Trautwein U., Köller O. The relationship between homework and achievement — still much of a mystery. *Educational Psychology Review*, 2003, pp. 115–145. https://doi.org/10.1023/A:1023460414243

[23] Darling-Hammond L., Ifill-Lynch O. If they'd only do their work! *Educational Leadership*, 2006, 63(5), pp. 8–13. https://www.ascd.org/el/articles/if-theyd-only-do-their-work (15.12.2022)

[24] Paulu N. Helping your students with homework: a guide for teachers. Washington, DC: U.S. Department of Education, Office of Educational Research. 1998. https://files.eric.ed.gov/fulltext/ED416037.pdf (14.12.2022)

[25] Galloway M., Conner J., Pope D. Nonacademic effects of homework in privileged, high-performing high schools. *The Journal of Experimental Education*, 2013, pp. 490–510. https://doi.org/10.1080/00220973.2012.745469

[26] Kaimre J. Murelahendajate koostamine Tartu Ülikooli kursuse „Programmeerimine" jaoks. Tartu Ülikooli arvutiteaduse instituudi bakalaureusetöö, 2021. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=71614 (22.03.2023)

[27] Lepp M., Palts T., Luik P., Papli K., Suviste R., et al. Troubleshooters for tasks of introductory programming MOOCs. *International Review of Research in Open and Distributed Learning*, 2018. https://doi.org/10.19173/irrodl.v19i4.3639

[28] Vihavainen A., Luukkainen M., Kurhila J. Multi-faceted support for MOOC in programming. *Proceedings of the 13th annual conference on Information technology education*, 2012, pp. 171–176. https://doi.org/10.1145/2380552.2380603

[29] Annamaa A., Suviste R., Vene V. Comparing different styles of automated feedback for programming exercises. *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 2017, pp. 183–184. https://doi.org/10.1145/3141880.3141909

[30] Tartu Ülikool. Arvutiteaduse instituut. Objektorienteeritud programmeerimine õppeaastal 2022/23. https://courses.cs.ut.ee/2023/OOP (19.01.2023)

[31] Bench S., Lench H. On the function of boredom. *Behavioral Sciences*, 2013, pp. 459–472. https://doi.org/10.3390/bs3030459

[32] Keen A., Mammen K. Program decomposition and complexity in CS1. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, pp. 48–53. https://doi.org/10.1145/2676723.2677219

[33] Elawar M.C., Corno L. A factorial experiment in teachers' written feedback on student homework: changing teacher behavior a little rather than a lot. *Journal of Educational Psychology*, 1985, pp. 162–173. https://doi.org/10.1037/0022-0663.77.2.162

[34] Chen J., Whittinghill D., Kadlowec J. Using rapid feedback to enhance student learning and satisfaction. *Proceedings. Frontiers in Education. 36th Annual Conference*, 2006, pp. 13–18. https://doi.org/10.1109/FIE.2006.322306

[35] Cantero-Chinchilla F.N., Díaz-Martín C., García-Marín A.P., Estévez J. Innovative student response system methodologies for civil engineering practical lectures. *Technology, Knowledge and Learning*, 2020, pp. 835–852. https://doi.org/10.1007/s10758-019-09410-z

[36] Muuli E. Graafiliste ülesannete automaatkontroll programmeerimise vaba juurdepääsuga e-kursuste raames. Tartu Ülikooli arvutiteaduse instituudi magistritöö, 2017. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=58502 (05.04.2023)

[37] Õpikeskkonna Moodle automaatkontrollide moodul. VPL – Virtual Programming Lab. https://vpl.dis.ulpgc.es (17.03.2023)

[38] Automaattestide raamistiku veebileht. JUnit 5. https://junit.org/junit5 (17.03.2023)

[39] Automaattestimise abiteegi koodihoidla. VPLJava. https://bitbucket.org/plas/vpljava (17.03.2023)

[40] Structuring and building a software component with Gradle. https://docs.gradle.org/current/userguide/multi_project_builds.html (17.03.2023)

[41] Burnstein I. Practical software testing. New York: Springer-Verlag. 2003. http://doi.org/10.1007/b97392

[42] Vaherpuu V. Murelahendajate loomise keskkond. Tartu Ülikooli arvutiteaduse instituudi bakalaureusetöö, 2016. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=53433 (18.03.2023)

[43] Sullivan G.M., Artino A.R. Analyzing and interpreting data from Likert-type scales. *Journal of Graduate Medical Education*, 2013, pp. 541–542. https://doi.org/10.4300/JGME-5-4-18

Lisad

I. Koduülesannete koostamise teoreetilise raamistiku kokkuvõte

Keerukus

Ülesanne peab olema:

- mõistlikku väljakutset pakkuv;
- tihedalt seotud vastava loengu materjaliga: ei tohi eeldada lisateadmisi;
- selge struktuuriga;
- arusaadava sõnastusega;
- korrakindla terminoloogia kasutamisega;
- teksti pikkuse mõttes nii lühike kui võimalik, nii pikk kui vajalik;
- raskemate alamülesannete puhul näidetega täiendatud.

Maht

- Ülesanne peab olema sellise mahuga, et lahenduse järgi saaks otsustada, kas õppija on vastavaid oskusi omandanud või mitte.
- Kodutöö lahendamisele aeg sõltub kursuse mahust. Näiteks kui kursus kestab 16 nädalat ning iseseisva töö maht on 92 tundi, siis iseseisvalt tuleb töötada umbes 6 tundi nädalas.

Seos õppeaine õpiväljunditega

- Igal ülesandel peab olema eesmärk, mis lähendab õppeaine õpiväljundite saavutamist.
- Ülesande sissejuhatuses on mõistlik kirja panna, mis on selle harjutuse eesmärgid.

Motiveerimine

- Huvitava konteksti lisamine ei tohi kahjustada ülesande keerukuse tasakaalu.
- Eellugu ei tohi liiga pikk olla.
- Ülesanne peab harjutama aktuaalseid oskusi (see punkt on suures osas tagatud õppematerjalide aktuaalsusega).

Probleemide osandamine

- Probleemi alamosadeks jaotamine peab mõistlik olema.
- Ülesandes peavad olema kirjeldatud konkreetsed sammud, mis on vajalikud ülesande lahendamiseks (kusjuures tehniline realiseerimine jääb iseseisvaks ülesandeks).
- Meetodite päised ning klasside ja isendiväljade nimed võivad samuti olla selgelt välja toodud.

Tagasiside

- Õppuritel peab olema võimalus oma lahenduse kohta tagasisidet saada.
- Tagasisides on tähtis näidata, mis lahenduse sammud olid edukad.
- Tagasisides peab selgelt välja tooma, mis vead tehtud on, ning pakkuda võimalikke lahendusi.
- Tagasiside peab olema kiire.
- Automaatkontrollid peavad olema kvaliteetsed, et tuvastada suurt osa võimalikest vigadest.
- Nende vigade jaoks, mis on tuvastatavad automaatkontrollide abil, on mõistlik luua murelahendajate küsimusi ja vastuseid.

Abi

- Ülesande lahendamise jooksul peab õppijatel olema võimalus õppejõudude poole pöörata.
- Murelahendajate keskkonna kasutamisel tuleb ennustada, mis ülesande osad võivad lahendamisel probleeme tekitada.
- Koostatud vihjed peavad olema selged ja piisavalt detailsed, et kasulikud olla.
- Vihjed ei tohi valmis lahendust anda.

II. Koostatud koduülesanded

Kodutöö 3, ülesanne 6

Ülesande põhieesmärgid on harjutada:

- klasside (koos isendiväljadega) ja konstruktorite (sh üledefineeritud) loomist;
- isendimeetodite (sh `get-`, `set-`, `toString`) loomist ja kasutamist;
- erinevate andmetüüpide (sh iseseisvalt loodud klasside) kasutamist;
- tingimuslausete kasutamist.

Koosta klass `Kohv`, millel on:

1. privaatne `String`-tüüpi isendiväli `kohvisort` kohvisordi jaoks;
2. privaatne `double`-tüüpi isendiväli `hind` tassi kohvi hinna jaoks;
3. konstruktor isendiväljade väärtustamiseks;
4. mõlema isendivälja jaoks `get`-meetodid;
5. `double`-tüüpi isendimeetod `tassideMaksumus`, mille parameeter on tasside arvu (`int`). Meetod peab tagastama kohvi tasside maksumuse.

```
Kohv kohv = new Kohv("Supremo", 2.5);
System.out.println("Tasside maksumus: " + kohv.tassideMaksumus(2));
// Väljastatakse: Tasside maksumus: 5.0
```

Koosta klass `Programmeerija`, millel on:

1. privaatne `String`-tüüpi isendiväli `programmeerijaNimi` programmeerija nime jaoks;
2. privaatne `double`-tüüpi isendiväli `riduKoodi` programmeerija tulemuslikkuse näitamiseks ehk mitu rida koodi päevas ta keskmiselt kirjutab;
3. privaatne `int`-tüüpi isendiväli `tasseKohvi`, mis kirjeldab, mitu tassi kohvi joob programmeerija päeva jooksul;
4. privaatne `Kohv`-tüüpi isendiväli `lemmikkohv` programmeerija lemmikkohvi jaoks;
5. konstruktor kõikide isendiväljade väärtustamiseks;
6. konstruktor juhuks, kui programmeerija kohvi ei joo. Sellel konstruktoril peab olema vaid 2 parameetrit: programmeerija nimi ja koodiridade arv. Konstruktor peab väärtustama vaid vastavaid isendivälju, ülejäänud isendiväljad tuleb jätta väärtustamata;
7. isendiväljade `tasseKohvi` ja `riduKoodi` jaoks `get-` ja `set`-meetodid;
8. parameetriteta `boolean`-tüüpi isendimeetod `kasJoobKohvi`, mis tagastab `true` kui lemmikkohv pole `null`, muidu `false`. Järgnevalt kasuta seda meetodit, kui on vaja kontrollida, kas programmeerija joob kohvi või mitte;
9. parameetriteta `double`-tüüpi isendimeetod `koodiTassiKohta`, mis tagastab, mitu rida koodi suudab programmeerija kirjutada, juues ühe tassi kohvi (valem: `riduKoodi / tasseKohvi`). Kui programmeerija kohvi ei joo, väljastatakse ekraanile programmeerija nimi ja teade, et ta kohvi ei joo, ning tagastatakse `-1`;

10. `toString` meetod, mille tagastus sõltub sellest, kas programmeerija joob kohvi või mitte.
    - Kui programmeerija joob kohvi, siis tagastab meetod:
        - programmeerija nime,
        - lemmikkohvi sordi nime,
        - lemmikkohvi tassi hinna,
        - rahasumma, mis igapäevaselt kulub kohvi joomisele (kasutades `tassideMaksumus` isendimeetodit),
        - meetodi `koodiTassiKohta` väärtuse.
    - Muidu tagastatakse vaid programmeerija nimi koos teatega, et ta kohvi ei joo.

Koosta peaklass nimega `Kohvijoomine`, mille peameetodis luuakse kaks isendit klassist `Kohv` ja kolm isendit klassist `Programmeerija` (kusjuures üks neist kohvi ei joo). Seejärel:

1. väljasta iga programmeerija kohta informatsiooni, kasutades meetodi `toString` tagastatud väärtust;
2. selgub, et programmeerijad jätsid töö viimasele hetkele ja peavad esitama oma projekti järgmisel hommikul. Tähtaja lähenedes kirjutavad nad rohkem koodi ning joovad rohkem kohvi. Suurenda vastavaid isendiväljade väärtusi, kasutades `get-` ja `set-` meetodeid:
    a. iga kohvijooja joob 3 tassi võrra rohkem kohvi kui tavaliselt ning kirjutab kaks korda rohkem koodiridu päevas;
    b. programmeerija, kes kohvi ei joo, kirjutab 100 rea võrra rohkem koodi, aga kohvi endiselt ei joo.
3. väljasta iga programmeerija kohta informatsioon uuesti.

Näide võimalikust peameetodi väljundist. Programmis oli loodud 3 programmeerijat:

```
Peeter. Kohvi ei joo
Indrek. Lemmikkohv on Supremo hinnaga 2.5. Igapäevaselt kohvi joomisele kulub
12.5 eurot. Programmeerija tulemuslikkus on 120.0 rida/tass
Sander. Lemmikkohv on Yirgacheffe hinnaga 3.0. Igapäevaselt kohvi joomisele kulub
6.0 eurot. Programmeerija tulemuslikkus on 350.0 rida/tass

Tähtaeg läheneb

Peeter. Kohvi ei joo
Indrek. Lemmikkohv on Supremo hinnaga 2.5. Igapäevaselt kohvi joomisele kulub
20.0 eurot. Programmeerija tulemuslikkus on 150.0 rida/tass
Sander. Lemmikkohv on Yirgacheffe hinnaga 3.0. Igapäevaselt kohvi joomisele kulub
15.0 eurot. Programmeerija tulemuslikkus on 280.0 rida/tass
```

Kodutöö 4, ülesanne 2

Ülesande põhieesmärgid on harjutada:

- klassi `String` isendimeetodite kasutamist;
- massiivi läbimist (tsükli abil);
- massiivi pikkuse kasutamist.

Koosta klass `SõnedeAnalüsaator`, millel on:

1. privaatne isendiväli `analüüsitavSõne` analüüsitava sõne jaoks;
2. konstruktor isendivälja väärtustamiseks;
3. parameetriteta void-tüüpi isendimeetod `väljastaSõne`, mis väljastab analüüsitava sõne koos tekstiga `Analüüsitav sõne on`;
4. parameetriteta `double`-tüüpi isendimeetod `leiaKeskminePikkus`, mis tagastab analüüsitavas sõnes esinevate sõnade keskmise pikkuse (*Märkus*: lihtsuse huvides võib sõna osaks pidada kõiki sümboleid peale tühiku);
5. parameetriteta `String`-tüüpi isendimeetod `leiaPikimSõna`, mis tagastab pikima sõna (kehtib sama märkus). Kui pikka sõna on mitu, siis meetod tagastab esimese neist.

Kui võtta lauset *Küsimus, kas arvuti suudab mõelda, on sama huvitav kui küsimus, kas allveelaev oskab ujuda*, siis:

- sõnes esinevate sõnade keskmine pikkus on `5.5`;
- pikim sõna on `allveelaev`.

Koosta peaklass nimega `SõnedeAnalüüsimine`, mille peameetodis:

1. luuakse sõnede analüüsaator mingi sõne jaoks;
2. väljastatakse ekraanile analüüsitav sõne, kasutades isendimeetodit `väljastaSõne`;
3. väljastatakse loetaval kujul ekraanile meetodite `leiaKeskminePikkus` ja `leiaPikimSõna` tagastatud väärtused.

Lisa (vabatahtlik): muuta klassi `SõnedeAnalüsaator` nii, et keskmise pikkuse ja pikima sõna leidmisel peetakse sõnaks vaid analüüsitava sõne alamsõnesid, mis koosnevad üksnes tähtedest.

Kodutöö 4, ülesanne 7

Ülesande põhieesmärgid on harjutada:

- failist lugemist klassi `File` ja `Scanner` isendite abil;
- tsükli abil faili läbimist;
- klassi `StringBuilder` isendi ja isendimeetodite kasutamist;
- klassi `String` isendimeetodite kasutamist.

Laurile väga meeldib osaleda erinevatel IT-võistlustel. Failis `võistlused.txt` on kõigi võistluste nimetused, millel ta on osalenud. Näide võimalikust faili sisust:

```
AtCoder Grand Contest 2019
Microsoft Imagine Cup 2019
International Collegiate Programming Contest 2021
Cyber Battle of Estonia 2022
```

Iga faili rida sisaldab võistluse nime (võib ka mitmesõnaline olla), millele järgneb ürituse toimumisaasta (neljakohaline arv). Koosta peaklass `Võistlused`, kus on:

1. staatiline `String`-tüüpi meetod `lühenda`, mille parameeter on üks faili rida. Meetod peab tagastama võistluse nime lühendi, võttes igast sõnast esitähe (suurtähena) ja aastaarvu kaks viimast numbrit ning asetades nende vahele ülakoma (`'`). Näiteks võistluse "**F**acebook **H**acker **C**up 20**20**" puhul peab meetod tagastama lühendi `FHC'20`;
2. peameetod, mis väljastab ekraanile kõigi failis olemasolevate võistluste nimetuste lühendid.

Näide peameetodi tööst ülaltoodud faili `võistlused.txt` korral:

```
AGC'19
MIC'19
ICPC'21
CBOE'22
```

Kodutöö 4, ülesanne 8

Ülesande põhieesmärgid on harjutada:

- failist lugemist klassi `File` ja `Scanner` isendite abil;
- tsükli abil faili läbimist;
- klassi `String` isendimeetodite kasutamist;
- meetodi `equals` kasutamist (sõnede võrdlemisel);
- listi loomist ning elementide lisamist ja eemaldamist.

Eesti Vabariigi 100. aastapäeva (2018. aastal) puhul otsustab Transpordiamet kodanikele kingituse teha. Nimelt on plaanis tasuta ja ilma eksamiteta väljastada juhiluba kõikidele kodanikele, kes on vähemalt 18-aastased (sh ka nendele, kes said täiskasvanuks alles 2018. aasta jooksul) ja kellel seda veel pole. Lisaks sellele tahab Transpordiamet teada, kes on kampaania raames juhiloa saanud ja kes mitte.

Kampaania läbiviimiseks on olemas fail `kodanikud.txt` järgmise struktuuriga:

```
Männik,Mari-Liis,47101010033,Olemas
Jõeorg,Jaak Kristjan,38001085718,Puudub
Pihlakas,Arno,37605030299,Olemas
Paanika,Peeter,36908209993,Puudub
Maasikas-Jõhvikas,Mari,61710020019,Puudub
```

Iga faili rida koosneb neljast osast, mis on omavahel eraldatud komadega. Andmed kodaniku kohta on järgmises järjekorras:

1. perekonnanimi (võib mitu olla);
2. eesnimi (võib mitu olla);
3. isikukood;
4. sõna `Olemas` või `Puudub` vastavalt sellele, kas inimesel on juhiluba juba olemas või mitte.

Koosta klass `Kodanik`, millel on privaatsed isendiväljad eesnime (`String`), perekonnanime (`String`), isikukoodi (`String`) ja juhiloa olemasolu (`boolean`) jaoks. Lisa konstruktor, mille abil saab kõiki isendivälju väärtustada. Lisa järgmised isendimeetodid:

1. `get`-meetod isikukoodi jaoks (`getIsikukood`);
2. parameetriteta `boolean`-tüüpi meetod `kasJuhilubaOnOlemas`;
3. parameetriteta `boolean`-tüüpi meetod `kasOnTäiskasvanuAastal2018`. Näiteks kui kodanik on sündinud 1999. aastal, siis tagastab meetod `true`, 2000. a – `true`, 2001. a – `false`;
4. `toString`-meetod, mis tagastab kodaniku eesnime ja perekonnanime (just selles järjekorras).

Koosta peaklass `TranspordiametiKampaania`.

1. Peaklassis peab olema staatiline abimeetod `loeKodanikud`, mille parameeter on faili nimi (`String`) ning mis tagastab kodanike listi (`ArrayList<Kodanik>`). Meetod peab hakkama saama suvalise failiga.
2. Peaklassis peab olema peameetod, kus:
    a. luuakse kodanike list (eelneva abimeetodi abil) faili `kodanikud.txt` põhjal;
    b. väljastatakse ekraanile eraldi ridadele loetaval kujul kodanike nimed koos infoga, kas kodanik sai juhiloa kingituseks või mitte (sel juhul tuleb väljastada ka põhjus);
    c. eemaldatakse listist need kodanikud, kes on kampaania raames juhiloa saanud. Vihje: üks võimalik variant on kasutada meetodit `removeAll`;
    d. väljastatakse ekraanile ülejäänud listis olevate kodanike isikukoodid.

Näide peameetodi tööst ülaltoodud faili `kodanikud.txt` korral:

```
Mari-Liis Männik juhiluba kingituseks ei saanud, põhjus: juhiluba on juba olemas.
Jaak Kristjan Jõeorg sai juhiloa kingituseks.
Arno Pihlakas juhiluba kingituseks ei saanud, põhjus: juhiluba on juba olemas.
Peeter Paanika sai juhiloa kingituseks.
Mari Maasikas-Jõhvikas juhiluba kingituseks ei saanud, põhjus: pole täiskasvanu.

Kodanikud, kes jäid kingitusest ilma:
47101010033
37605030299
61710020019
```

Kodutöö 6, ülesanne 5

Ülesande põhieesmärgid on harjutada:

- ülem- ja alamklasside loomist;
- polümorfismi kasutamist;
- dünaamilise seostamise kasutamist;
- ülemklassi meetodite kasutamist;
- meetodite ülekatmist.

Üks autoteenindus palub arendada autoremondi haldussüsteemi. Autoteenindus tegeleb sõidu-, veo- ja luksusautode parandamisega. Süsteemi koostamisel tuleb arvestada sellega, et iga autoliigi puhul kehtib oma hinnakiri.

Koosta klass `Auto`, millel on privaatsed isendiväljad omaniku nime (`String`) ja automudeli (`String`) jaoks ning privaatne isendiväli, mis määrab, kas tegemist on elektriautoga (`boolean`). Lisaks on klassis:

1. konstruktor isendiväljade väärtustamiseks;
2. `double`-tüüpi isendimeetod `arvutaParanduseMaksumus`, millel on üks `double`-tüüpi parameeter auto parandamisele kulutatud aja jaoks (tundides). Meetod peab tagastama remondi maksumuse, eeldusel, et töötunni hind elektriauto korral on 36 eurot ja muudel juhtudel 40 eurot (vt *näide 1*);
3. parameetriteta `String`-tüüpi isendimeetod `autoliik`, mis tagastab sõna `Sõiduauto`;
4. `toString` meetod, mis tagastab autoliigi (kasutades eelmist meetodit) koos automudeli ja omanikuga (nt kujul: `Sõiduauto. Mudel: Audi A4 Avant; omanik: Peeter Paanika`).

Näide 1:

```
Auto auto = new Auto("Peeter Paanika", "Audi A4 Avant", false);
System.out.println(auto.arvutaParanduseMaksumus(2.5));
// Väljastatakse: 100.0
```

Koosta klassi `Auto` alamklass `Veoauto`, millel on:

1. privaatne `boolean`-tüüpi isendiväli, mille väärtus näitab, kas veoauto omanik on füüsiline isik (välja väärtuseks `true`) või mitte;
2. konstruktor isendiväljade väärtustamiseks (kokku 4 parameetrit);
3. ülekaetud isendimeetod `arvutaParanduseMaksumus`. Meetod peab tagastama kahekordse või kolmekordse ülemklassi samanimelise meetodi poolt arvutatud väärtuse vastavalt sellele, kas omanik on füüsiline isik või ettevõte (vt *näide 2*);
4. ülekaetud isendimeetod `autoliik`, mis tagastab sõna `Veoauto`.

Näide 2:

```
Veoauto veoauto = new Veoauto("Mu firma", "Volvo", false, false);
System.out.println(veoauto.arvutaParanduseMaksumus(1));
// Väljastatakse: 120.0
```

Koosta klassi `Auto` alamklass `Luksusauto`, millel on:

1. privaatne `int`-tüüpi isendiväli tootmisaasta jaoks;
2. konstruktor isendiväljade väärtustamiseks;
3. ülekaetud isendimeetod `arvutaParanduseMaksumus`. Meetod peab tagastama 15-kordse või 10-kordse ülemklassi samanimelise meetodi poolt arvutatud väärtuse vastavalt sellele, kas auto on vanem kui 70 aastat või mitte;
4. ülekaetud meetod `autoliik`, mis tagastab sõna `Luksusauto`.

Koosta klassi `Luksusauto` alamklass `Limusiin`, millel on:

1. konstruktor ülemklassi isendiväljade väärtustamiseks;
2. ülekaetud isendimeetod `arvutaParanduseMaksumus`. Meetod peab tagastama ülemklassi samanimelise meetodi 1,5-kordse väärtuse;
3. ülekaetud meetod `autoliik`, mis tagastab sõna `Limusiin`.

Koosta klass `Autoteenindus`, millel on:

1. privaatne `int`-tüüpi isendiväli parandatud autode arvu jaoks;
2. privaatne `double`-tüüpi isendiväli autoteeninduse teenitud tulu jaoks;
3. `void`-tüüpi isendimeetod `paranda`, millel on 2 parameetrit: auto (`Auto`-tüüpi), mida parandatakse, ja remondile kulutatud aeg tundides (`double`-tüüpi). Meetod peab väljastama ekraanile info auto kohta koos remondi hinnaga ning vastavalt suurendama autoteeninduse parandatud autode ja tulu isendiväljade väärtusi (vt *näide 3*);
4. `toString` meetod, mis tagastab parandatud autode arvu koos teenitud tulu suurusega (loetaval kujul).

Näide 3:

```
Autoteenindus autoteenindus = new Autoteenindus();
autoteenindus.paranda(auto, 2.5);
autoteenindus.paranda(veoauto, 1);
System.out.println(autoteenindus);
```

Ekraanile väljastatakse:

```
Sõiduauto. Mudel: Audi A4 Avant; omanik: Peeter Paanika – 100.0
Veoauto. Mudel: Volvo; omanik: Mu firma – 120.0
Parandatud autosid: 2, tulu: 220.0
```

Koosta peaklass `AutodeParandamine`, kus peameetodis:

1. luuakse autoteenindus ja vähemalt üks auto igast klassist (`Auto`, `Veoauto`, `Luksusauto`, `Limusiin`);
2. kõiki autosid parandatakse loodud autoteeninduses, kusjuures remondile kulutatud aeg tuleb iga auto jaoks genereerida juhuslikult täisarvude poollõigust `[1, 21)` ja jagada seda ujukomaarvuga `2.0`. *Vihje*: juhuslike arvude genereerimiseks saab kasutada klassi `Random` isendit;
3. väljastatakse ekraanile info autoteeninduse kohta.

Kodutöö 6, ülesanne 6

Ülesande põhieesmärgid on harjutada:

- abstraktse klassi loomist;
- polümorfismi kasutamist;
- dünaamilise seostamise kasutamist;
- liidese kasutamist.

Pagarikojas "GLaDOSil külas" küpsetati hulk erinevaid kooke: ümmargusi, ristkülikukujulisi ja kolmnurkseid. Info kõigi valmistatud kookide kohta on failis `koogid.txt`.

Näide faili võimalikust sisust:

```
Laimitort; 2023-03-10; 0.04; 20
Kirsitort; 2023-03-09; 0.06; 20; 25
Sidrunibeseekook; 2023-03-11; 0.07; 20; 20; 30
; 2023-03-12; 0.07; 25
Mangotort; 2023-03-06; 0.05; 15
Kohvitort; 2023-03-10; 0.055; 22; 22
```

Iga faili rida koosneb vähemalt neljast osast, mis on omavahel eraldatud semikooloni ja tühikuga. Esimesed kolm on koogi nimetus (võib ka tühi sõne olla), "parim enne" kuupäev (kujul `AAAA-KK-PP`) ja koogi ruutsentimeetri hind (eurodes). Nendele järgnevad koogi mõõtmed sentimeetrites (üks kuni kolm suurust). Suuruste arv sõltub koogi kujust:

- ümmargusel koogil on vaid üks suurus – koogi läbimõõt;
- ristkülikukujulise koogi mõõtmed on koogi laius ja pikkus;
- kolmnurkse koogi puhul on mõõtmed külgede pikkused.

Pagarikoja töötajad tahavad failis olevate kookide hindu teada saada. Lisaks sellele on vaja järjestada kooke "parim enne" kuupäeva järgi, et kookide müümisel vältida müügitähtaja ületamist.

Koosta abstraktne klass `Kook`, millel on privaatsed isendiväljad koogi nimetuse (`String`), kuupäeva "parim enne" (`LocalDate`) ja ruutsentimeetri hinna (`double`) jaoks. Klassis peavad olema:

1. konstruktor isendiväljade väärtustamiseks. Kui koogi nimetus oli tühi sõne, siis omistada nimetuse isendiväljale väärtus `The cake is a lie` (lisainfo);
2. parameetriteta abstraktne `double`-tüüpi isendimeetod `pindala`;
3. parameetriteta `double`-tüüpi isendimeetod `koogiHind`, mis tagastab koogi hinna, kasutades meetodit `pindala` ja ruutsentimeetri hinda. Koogi hind peab olema ümardatud kahe komakohani;
4. `toString` meetod, mis tagastab loetaval kujul koogi nimetuse, koogi hinna ja kuupäeva "parim enne".

Klass `Kook` peab realiseerima liidest `Comparable<Kook>`. Võrrelda tuleb kuupäeva "parim enne" põhjal põhimõtte "mida hilisem kuupäev, seda "suurem" see on" järgi.

Vihje: klassis `LocalDate` on olemas meetod `compareTo`.

Koosta klassi `Kook` mitteabstraktne alamklass `ÜmmarguneKook`. Klassis peab olema privaatne `double`-tüüpi isendiväli läbimõõdu jaoks ning konstruktor isendiväljade väärtustamiseks (kokku 4 parameetrit). Realiseeri meetod `pindala` järgmise valemi järgi: `S = π * r²`.

```
ÜmmarguneKook kook = new ÜmmarguneKook("Laimitort", LocalDate.parse("2023-03-10"), 0.04, 20);
System.out.println(kook.pindala());
// Väljastatakse: 314.1592653589793
System.out.println(kook.koogiHind());
// Väljastatakse: 12.57
```

Koosta klassi `Kook` mitteabstraktne alamklass `RistkülikukujulineKook`. Klassis peavad olema privaatsed `double`-tüüpi isendiväljad laiuse ja pikkuse jaoks ning konstruktor isendiväljade väärtustamiseks. Realiseeri meetod `pindala` järgmise valemi järgi: `S = a * b`.

Koosta klassi `Kook` mitteabstraktne alamklass `KolmnurkneKook`. Klassis peavad olema privaatsed `double`-tüüpi isendiväljad külgede pikkuste jaoks ning konstruktor isendiväljade väärtustamiseks. Realiseeri meetod `pindala` Heroni valemi järgi.

Koosta peaklass `Pagarikoda`. Klassis peavad olema:

1. staatiline abimeetod `loeKoogid`, mille parameeter on faili nimi ning mis tagastab kookide listi (`List<Kook>`).
   Vihje: sõne põhjal klassi `LocalDate` isendi loomisel on abiks meetod `LocalDate.parse`.
   Nõuanne: sõne põhjal koogi loomine on mõistlik realiseerida eraldi staatilise meetodina, kuid see pole kohustuslik.
2. peameetod, kus
   a. luuakse kookide list faili `koogid.txt` põhjal (eelneva abimeetodi abil);
   b. järjestatakse mittekahanevalt listi elemente (*Vihje*: meetod `Collections.sort`);
   c. väljastatakse eraldi ridadele listis olevad koogid ekraanile.

Näide peameetodi tööst ülaltoodud faili `koogid.txt` korral:

```
Mangotort – 8.84 eurot – parim enne 2023-03-06
Kirsitort – 30.0 eurot – parim enne 2023-03-09
Laimitort – 12.57 eurot – parim enne 2023-03-10
Kohvitort – 26.62 eurot – parim enne 2023-03-10
Sidrunibeseekook – 13.89 eurot – parim enne 2023-03-11
The cake is a lie – 34.36 eurot – parim enne 2023-03-12
```

Kodutöö 9, ülesanne 4

Ülesande põhieesmärgid on harjutada voogude kasutamist:

- tekstifaili lugemiseks;
- binaarfaili lugemiseks;
- binaarfaili kirjutamiseks.

Peeter Paanika soovib luua oma Moodle'i ning tal on vaja realiseerida funktsionaalsust, mis võimaldaks faili sisu põhjal määrata ainete hindeid. Nimetame seda faili punktide failiks (`punktid.txt`). Näide faili võimalikust sisust:

```
OOP:10,10,10,10,10,9.5,10,10,10,8
Programmeerimine II:5,4,5,5,10,5,5,5,4,10,4,25.5
Andmebaasid:5,10,2,7,20,2.5,8,6,5.5,3,1
```

Faili iga rida koosneb kahest osast, mis on omavahel eraldatud kooloniga. Esimeses neist on õppeaine nimetus. Teises on selle aine jooksul erinevate tööde eest teenitud punktid. See osa koosneb omakorda vähemalt ühest või mitmest osast, kus igaüks neist on täis- või ujukomaarv. Need on omavahel eraldatud komaga.

Koosta klass `AineHinne`, millel on privaatsed isendiväljad aine nimetuse (`String`) ja tähelise hinde (`char`) jaoks vastavalt nimedega `nimetus` ja `hinne`. Klassis peab olema vastav konstruktor, mis võimaldab isendivälju väärtustada. Lisaks peab klassis olema:

1. `get`-meetodid mõlema isendivälja jaoks;
2. isendimeetod `toString`, mille tulemus sisaldab nii aine nimetust kui ka tähelist hinnet;
3. privaatne `char`-tüüpi isendimeetod `hinnePunktideMassiivist`, mille parameeter on punktide sõnede massiiv (`String[]`). Meetod peab tagastama sümboli `A`, `B`, `C`, `D`, `E` või `F` vastavalt punktide summale.
   Skaala: `A > 90, B > 80, C > 70, D > 60, E ≥ 51, F < 51` (vt *näide 1*);
4. teine konstruktor, mille parameetrid on aine nimetus (`String`) ja punktide massiiv (`String[]`) ning mis väärtustab isendivälju, kasutades hinde määramiseks eelmist meetodit.

Näide 1:

```
String[] punktid = new String[]{"29.5", "30", "28.1"}; // Summa: 87.6
System.out.println(hinnePunktideMassiivist(punktid));
// Väljastatakse: B
```

Koosta klass `Tudeng`, millel on privaatsed isendiväljad nime (`String`) ja hinnete listi (`List<AineHinne>`) jaoks. Klassis peab olema konstruktor, mis võimaldab isendivälju väärtustada. Lisaks peab klassis olema:

1. isendimeetod `toString`, mille tulemus sisaldab nii tudengi nime kui ka tema hindeid kursuste kaupa (vt *näide 2*);

2. `void`-tüüpi isendimeetod `salvestaBinaarfaili`, mille parameeter on faili nimi (`String`). Meetod peab kasutama klassi `DataOutputStream` isendit. Meetod peab salvestama faili tudengi nime koos tema hinnetega.
   Vihje 1: iga klassi `AineHinne` isend on kombinatsioon aine nimetusest (`String`) ja hindest (`char`). Mõlema tüübi jaoks on klassis `DataOutputStream` olemas `write`-isendimeetodid.
   Vihje 2: hilisema sisselugemise lihtsustamiseks võiks paaride `nimetus-hinne` arvu faili algusesse kirjutada.
3. staatiline `Tudeng`-tüüpi meetod `loeBinaarfailist`, mille parameeter on faili nimi (`String`). Meetod peab kasutama klassi `DataInputStream` isendit. Meetod peab looma ja tagastama klassi `Tudeng` isendi faili sisu põhjal (vt *näide 2*). Võib eeldada, et faili struktuur on sama, mis eelmise meetodi puhul.

Näide 2:
```
// Eeldusel, et tudeng.bin juba eksisteerib
Tudeng tudeng = Tudeng.loeBinaarfailist("tudeng.bin");

System.out.println(tudeng);
// Väljastatakse:
Peeter Paanika: [OOP: A, Programmeerimine II: B, Andmebaasid: D]
```

Koosta peaklass `TudengPeaklass`, millel on:

1. staatiline abimeetod `loePunktideFail`, mille parameeter on punktide faili nimi (`String`, formaat on ülalpool toodud) ning mis tagastab ainete hindeid listina (`List<AineHinne>`). Meetod peab lugemiseks kasutama klassi `BufferedReader` isendit.
2. peameetod, kus luuakse mingi tudeng (faili `punktid.txt` abil) ning testitakse meetodeid `salvestaBinaarfaili` ja `loeBinaarfailist`.

Lisaülesanne (pole kohustuslik): soovi korral uuri jadastuse (ingl *serialization*) ja objektimise (ingl *deserialization*) operatsioone. Esimene on isendi muutmine baidijadaks, teine – jadastuse pöördtoiming. Lingid:

- https://www.baeldung.com/java-serialization
- https://www.geeksforgeeks.org/serialization-in-java/

Kodutöö 9, ülesanne 7

Ülesande põhieesmärgid on harjutada voogude kasutamist: • tekstifaili kirjutamiseks; • baithaaval faili lugemiseks.

On antud mitu faili, kuhu on peidetud mõned salafraasid (lihtsuse mõttes saab neid fraase näha faili sisust). Näiteks on 3 neist:

- fail1.txt – sisaldab salafraasi *Objektorienteeritud programmeerimine on tore*;
- fail2.txt – salafraasi ei sisalda;
- fail3.txt – sisaldab salafraasi *Hea uni on oluline*.

NB! Oma arvutis programmi testimiseks tasub faili sisu kopeerimisele eelistada salvestamist (`Ctrl+S`). Muidu ei pruugi toodud failide sisu täiesti samaks jääda.

Koosta klass `SalafraasiLeidmine`, millel on:

1. staatiline `String`-tüüpi meetod `leiaEluUniversumiJaKõigeSalafraas`, mille parameeter on faili nimi (`String`), kust tuleb salafraasi otsida. Meetod peab kasutama klassi `RandomAccessFile` isendit. Meetod peab tagastama salafraasi, kui see on failis olemas, muidu tagastatakse `null` (vt *näide 1*).
   On teada, et:
   a. kui salafraas on failis olemas, siis see lause algab alati 42. baidist.
      Vihje: klassi `RandomAccessFile` isendimeetod `seek`;
   b. kui salafraas on failis olemas, siis see lause lõpeb alati nullbaidiga (vastava baidi lugemine klassi `RandomAccessFile` isendimeetodiga `read` tagastab `0`);
   c. failis on nullbait olemas siis ja ainult siis, kui salafraas on failis olemas;
   d. alates 42. baidist (kaasa arvatud) kuni nullbaidini (välja arvatud) on iga loetud bait vaja teisendada tüübiks `char`. Nendest sümbolitest tuleb kokku panna salafraas sõnena.
      Vihje: `char`-idest sõne moodustamiseks on mugav kasutada klassi `StringBuilder` isendit.

2. peameetod, milles kasutaja saab korduvalt sisestada failide nimesid, kust tuleb salafraase otsida (vt *näide 2*). Tühja sõne sisestamisel peab programm töö lõpetama (selleks ei tohi kasutada meetodit `System.exit()`). Muudel juhtudel peab programm salafraase otsima (kasutades eelmist meetodit) ning salvestama need eraldi ridadele klassi `BufferedWriter` isendi abil faili `salafraasid.txt` (vt *näide 3*). Salafraasi puudumise korral väljastatakse ekraanile teade `Salafraasi ei leidnud` ning sel juhul faili kirjet ei teki.

Näide 1:

```
System.out.println(leiaEluUniversumiJaKõigeSalafraas("fail1.txt"));
// Väljastatakse: Objektorienteeritud programmeerimine on tore

System.out.println(leiaEluUniversumiJaKõigeSalafraas("fail2.txt"));
// Väljastatakse: null
```

Näide 2:

Näide peameetodi tööst. Loetavuse huvides kasutaja sisendi jaoks mõeldud read algavad sümboliga > :

```
Sisesta faili nimi:
> fail1.txt

Sisesta faili nimi:
> fail2.txt
Salafraasi ei leidnud

Sisesta faili nimi:
> fail3.txt

Sisesta faili nimi:
>
```

Näide 3:

Näide failist `salafraasid.txt` pärast peameetodi käivitamist:

```
Objektorienteeritud programmeerimine on tore
Hea uni on oluline
```

III. Koostatud automaatkontrollid

Privaatse koodihoidla link: https://github.com/azakatov/oop-koduylesanded-2023.

Tegemist on privaatse koodihoidlaga, kuna see sisaldab koostatud koduülesannete näidislahendusi. Valmis lahenduste avalik levitamine ei ole kursuse õppejõudude huvides. Ligipääs koodihoidlale on olemas käesoleva lõputöö autoril, retsensendil ja kursuse vastutaval õppejõul.

IV. Koostatud murelahendajad

Kodutöö 3, ülesanne 6: https://progtugi.cs.ut.ee/#/ts/63de4bdd953066fc5c9fb6e8/.

Kodutöö 4, ülesanne 2: https://progtugi.cs.ut.ee/#/ts/63de4c39953066fc5c9fb718/.

Kodutöö 4, ülesanne 7: https://progtugi.cs.ut.ee/#/ts/63de4c3c953066fc5c9fb733/.

Kodutöö 4, ülesanne 8: https://progtugi.cs.ut.ee/#/ts/63de4c3f953066fc5c9fb76e/.

Kodutöö 6, ülesanne 5: https://progtugi.cs.ut.ee/#/ts/63fe6530953066fc5ca0e090/.

Kodutöö 6, ülesanne 6: https://progtugi.cs.ut.ee/#/ts/63fe6535953066fc5ca0e0af/.

Kodutöö 9, ülesanne 4: https://progtugi.cs.ut.ee/#/ts/6428573c953066fc5ca1de07/.

Kodutöö 9, ülesanne 7: https://progtugi.cs.ut.ee/#/ts/6428574f953066fc5ca1de2a/.

V. Näide küsimustikust kursuse osalejatelt tagasiside saamiseks

Kuna küsimustik on keerulisema struktuuriga (koosneb mitmest jaotisest), siis parema näitlikkuse huvides loodi interaktiivne näide. Koostamisel kasutati küsimusi 6. õppenädala tagasisideküsitlusest: https://azakatov.github.io/interaktiivne-kysimustik/.

VI. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Anton Zakatov**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Abivahenditega varustatud koduülesannete koostamine Tartu Ülikooli kursusele „Objektorienteeritud programmeerimine”**, mille juhendaja on **Marina Lepp**, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Anton Zakatov

08.05.2023