TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Nikita Chernov

Ingliskeelsete õppematerjalide koostamine ainele „Objektorienteeritud Programmeerimine“

Bakalaureusetöö (9 EAP)

Juhendaja: Ljubov Jaanuska, PhD

Tartu 2025

Ingliskeelsete õppematerjalide koostamine ainele „Objektorienteeritud Programmeerimine“

Lühikokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua ingliskeelsed õppematerjalid Tartu Ülikooli kursusele „Objektorienteeritud programmeerimine“ viienda ja kuuenda nädala jaoks. Materjalid sisaldavad lühikesi teoreetilisi ülevaateid klasside, objektide, päriluse, polümorfismi ja liideste kohta; praktilisi ülesandeid; *JUnit*-testidel põhinevat automaatkontrolli Moodle-i Virtual Programming Labis; ning H5P interaktiivseid enesekontrolliteste. Materjalide väljatöötamisel tugines autor ümberpööratud klassiruumi meetodile ja ning järgis õppematerjalide koostamise häid praktikaid.

Võtmesõnad: objektorienteeritud programmeerimine; ingliskeelsed õppematerjalid; ümberpööratud klassiruum; automaatkontroll; enesekontroll

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

Development of Learning Materials for the “Object-Oriented Programming” Course in English

Abstract

The aim of this bachelor’s thesis was to create two English-language modules for the course “Object-Oriented Programming” taught at the University of Tartu during week 5 and week 6. The materials include concise theoretical overviews of classes, objects, inheritance, polymorphism, and interfaces; practical assignments; automated assessments based on JUnit tests in Moodle’s Virtual Programming Lab; and H5P interactive self-assessment quizzes. Developing the materials, the author relied on the flipped classroom method and best practises.

Keywords: object-oriented programming; English language study materials; inverted classroom; automatic check; self-check.

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

Sisukord

Sissejuhatus

Objektorienteeritud programmeerimine (OOP) on üks enim levinud programmeerimisparadigmasid, mida õpetavad nii veebipõhised õppeplatvormid kui ka kõrgkoolide kursused. Coursera platvorm pakub alates 2025. aastast üle 2500 objektorienteeritud programmeerimise kursust, mis võimaldavad eri tasemega õppijatel omandada OOP-põhimõisteid ning praktilisi oskusi (Coursera, 2025). Codecademy platvormis on üle 10 erineva kursuse, mille põhirõhk on OOP mõistete õpetamisel (Codecademy, 2025).

Tartu Ülikoolis on samuti vastav eestikeelne kursus LTAT.03.003 „Objektorienteeritud programmeerimine“, mis annab 6 EAP mahus põhjaliku sissejuhatuse OOP-i kontseptsioonidesse ja rakendustesse (Tartu Ülikooli õppeinfosüsteem). Ingliskeelsete materjalide puudumine piirab mitte-eestikeelsete üliõpilaste võimalust kursusel osaleda. Ingliskeelsete õppematerjalide loomine aitab tutvuda tarkvaraarenduse terminoloogiaga algusest peale inglise keeles ning valmistab ette osalemiseks globaalsetes meeskondades. Töö eesmärgiks on seega ingliskeelsete õppematerjalide väljatöötamine, mis edaspidi integreeritakse Tartu Ülikooli objektorienteeritud programmeerimise kursusele, täpsemalt:

1. teoreetilised ülevaated klasside, objektide, päriluse, polümorfismi ja liideste kontseptsioonidest;
2. praktilised ülesanded Java-keeles, mis kinnistavad õpitut;
3. automaatkontrolli testid Virtual Programming Lab-i, mis annab kohest tagasisidet;
4. enesekontrollitestid, mis toetavad iseseisvat õppimist.

Töö on jagatud neljaks peatükiks. Esimene peatükk tutvustab kursuse ülesehitust, mahtu ja sihtrühma. Teine peatükk käsitleb õppematerjalide koostamise didaktilist kavandamist. Kolmandas peatükis kirjeldatakse ingliskeelsete õppematerjalide väljatöötamise protsessi. Neljandas peatükis kirjeldatakse kahte valminud õppemoodlid, analüüsitakse üliõpilaste tagasisidet ning esitatakse mõtisklusi töö tulevaste arenguvõimaluste üle. Lõpus esitatakse kokkuvõte peamistest tulemustest.

Tekstide sõnastuse parandamiseks kasutati firma OpenAI juturoboti ChatGPT (versioonid 4o, o3, o4-mini-high). Tehisintellekti abil parendati tekstide loetavust nii õppematerjalides kui ka bakalaureusetöö teoreetilises osas.

1. Kursuse “Objektorienteeritud programmeerimine” kirjeldus

Käesoleva peatüki põhieesmärgiks on anda ülevaade loodavast Tartu Ülikooli kursusest „Objektorienteeritud programmeerimine“, keskenduses selle struktuurile, sihtrühmale ja eesmärkidele.

1.1 Kursuse eesmärk

Tartu Ülikooli kursuse „Objektorienteeritud programmeerimine” eesmärk on anda üliõpilastele põhjalik ülevaade objektorienteeritud programmeerimise (OOP) alustest. Õpetatavate teemade hulka kuuluvad Java programmeerimiskeele baasstruktuurid, objektide ja klasside loomine ning nende rakendamine, sisend- ja väljundvoogude käsitlemine, failidega töötamine, loendid ja andmestruktuurid, polümorfism, liidesed, ülem- ja alamklassid, abstraktsed klassid, graafiline kasutajaliides, sündmustepõhine programmeerimine ning erindite töötlemine. Iga teema on valitud eesmärgiga toetada üliõpilaste võimekust arendada terviklikke OOP-rakendusi ja lahendada reaalseid programmeerimisülesandeid.

Õppeaine on suunatud üliõpilastele, kelle eriala ei ole informaatika. Kursus sobib samuti ka üliõpilastele, kellel on varasem minimaalne programmeerimiskogemus, näiteks läbitud sissejuhatav kursus “Pythoni programmeerimise alused”.

1.2 Õppemetoodika

Käesolevat kursust arendatakse ümberpööratud klassiruumi (ingl *flipped classroom*) põhimõttel. Selle meetodi raames tutvuvad üliõpilased uue materjaliga esmalt iseseisvalt ning osalevad seejärel kontakttundides, kus eelnevalt õpitud teadmised kinnistatakse praktiliste harjutuste abil (Engel & Kapp, 2017; Bishop & Verleger, 2013; Chen & Hsu, 2022).

Uuringud on näidanud ümberpööratud õppimise positiivset mõju õppijate akadeemilisele tulemuslikkusele ja kaasatusele (Engel & Kapp, 2017). Näiteks oma uurimuses Engel ja Kapp (2017) rõhutavad ümberpööratud klassiruumi meetodi positiivset mõju õppe kvaliteedile ja tulemustele programmeerimise kursustel. Chen ja Hsu (2022) on tõendanud, et ümberpööratud klassiruumi meetodi kasutamine suurendab üliõpilaste kaasatust ja parandab nende sooritust programmeerimisülesannete lahendamisel, kuna õppijad puutuvad õpitavaga kokku aktiivsemalt ja praktilisemalt. Lisaks Bishop ja Verleger (2013) suurendavad ümberpööratud klassiruumi meetod

õppijate motivatsiooni, sest üliõpilased saavad ise aktiivselt teadmisi kasutada ja lahenduste loomisest osa võtta. Kuigi mõningad õppijad võivad esmalt eelistada traditsioonilist auditoorset loengut, näitavad Chen ja Hsu (2022) ning Bishop ja Verleger (2013), et üldine rahulolu ja õpiväljundite saavutamine on ümberpööratud klassiruumi puhul kõrgemad.

2. Õppematerjalide koostamise didaktilised aspektid

Õppematerjali väljatöötamine eeldab põhjalikku pedagoogilist tausta. Kuna autori varasemad kogemused õppematerjalide koostamisega olid väga piiratud, algab töö kirjanduse ülevaatega, et tuvastada aspekte, mis toetavad kvaliteetsete õppematerjalide väljatöötamist. Peatüki võtmetulemused on esitatud lühikokkuvõttena lisas 1, et hõlbustada nende kiiret kasutamist.

2.1 Kodutööde tõhusus

Kodutöö võib avaldada positiivset mõju õpilaste akadeemilisele saavutusele, kuid selle tõhusus sõltub paljudest teguritest. Terada (2015) rõhutab, et oluline pole mitte ainult kodutööde hulk, vaid nende autentsus ja tähenduslikkus: probleemilahendus, loominguline rakendus ja seoste loomine reaaleluga toetavad õppimist enam kui drill-tüüpi ülesanded. Sarnaseid järeldusi teevad ka Cooper *et al.* (2006) metaanalüüsi põhjal, kus efektiivsemad olid just ettevalmistavad ja laiendavad tööd, mis aitasid uue materjali omandamisele kaasa.

Kodutööde mahu osas on uuringute tulemused erinevad. Fernández-Alonso *et al.* (2017) leidsid, et mahukam kodutöö annab paremaid tulemusi, ent individuaalsel tasandil on seos vastupidine — õppijad, kes kulutasid keskmisest enam aega, saavutasid hoopis madalamaid punkte. Autorid järeldasid, et kõige enam võidavad need õppijad, kus igapäevane kodutöö maht jääb 60 – 90 minuti piiresse; sellest rohkem tekitab pigem ebavõrdsust ja väsimust.

2.2 Programmeerimisülesanded

Programmeerimisülesanded väljatöötamisel rõhutab Vatterott (2010), Darling-Hammond ja Ifill-Lynch (2006), et igal ülesandel peab olema üheselt tuvastatav akadeemiline eesmärk (nt harjutamine, mõistmise kontroll, rakendus) ja et see eesmärk tuleb õppijal selgelt välja tuua. Paulu ja Darby (1998) lisavad, et eesmärgi nähtavaks tegemine vähendab segadust ja tõstab soorituse kvaliteeti, sest õppijad teavad, kuidas töö seostub järgmiste tundide sisuga. Bench ja Lench (2013) selgitavad emotsiooniteooriale tuginedes, et igavus tekib just siis, kui tegevuse eesmärk muutub ebamääraseks.

Hundley ja Britt (2009) rõhutavad, et programmeerimisülesanded, mis lähtuvad õppijatele tähenduslikust teemast või projektist, soodustavad õppija isiklikku pühendumust ning suurendavad püsivust keerukamate eesmärkide poole liikumisel. Sama kinnitavad

Epstein ja Van Voorhis (2001): ajamahu suurendamine toob kasu vaid siis, kui ülesanne on loogilisest üles ehitatud ja toetatud konstruktiivse tagasisidega. Craig *et al*. (2017) rõhutasid, et tuttav teema ei pruugi programmeerimisülesannet lihtsamaks muuta, sest keerulised mõisted ja pikad juhised võivad selle eelise ära võtta.

Ülesannete osadeks jaotamine (ingl *task decomposition*) on oluliseks osaks nii objektorienteeritud programmeerimise kui ka laiemalt kogu tarkvaraarenduse õpetamisel. See on eriti oluline üliõpilaste jaoks, kes läbivad uut kursust, mis loodi käesoleva bakalaureusetöö raames, kuna üheks võimalikuks raskuseks võib olla keerukate programmide struktuuri loomise oskuse puudus. Uuringud näitavad, et keeruka probleemi tükeldamine vähendab nii kognitiivset koormust kui ka programmi arendamise keerukust (Keen & Mammen, 2015). Keen ja Mammen (2015) näitasid oma uuringus, et dekompositsioonioskuste järkjärguline arendamine võib vähendada koodi keerukust ning suurendada selle modulaarsust.

Haldeman *et al*. (2025) pakuvad kolme tõhusat dekomponeerimise põhimõtet

1. ühe vastutuse printsiip — iga funktsioon lahendab ühte selgelt piiritletud alamülesannet;
2. sidususe maksimeerimine ja tehisliku haakumise vähendamine — minimeeritakse üleliigne andmeedastus funktsioonide vahel;
3. korduskoodi elimineerimine ja taaskasutatavus — üleliigsed või identsed koodilõigud koondatakse eraldi funktsioonidesse.

Loetletud aspktide järgitakse hoolikalt bakalaureusetöö raames arendavas OOP-kursuses programmeerimisülesannete väljatöötamisel.

2.3 Automaatkontrollid

Automaatkontroll (AK) on protsess, mille käigus analüüsitakse õppijate poolt esitatud lahendusi programmikoodi kujul spetsiaalse süsteemi abil, mis genereerib ka kohese ja struktureeritud tagasiside. AK mõju on põhjalikult uuritud ka suurtes avatud veebikursustes (MOOC). Gabbay ja Cohen (2022) uurisid Pythoni programmeerimiskeelt käsitlevat MOOC-i, kus õppijad võisid lahendada ülesandeid piiramatu arv kordi ja said automaatset tagasisidet. Tulemused näitasid suuremat õppijate kaasatust ja paremaid testisooritusi võrreldes kontrollrühmaga (Gabbay & Cohen, 2022). Süstemaatiline ülevaade, mille koostasid Cavalcanti *et al*. (2021) näitavad, et

ligikaudu kaks kolmandikku analüüsitud uuringutest raporteerib AK positiivset mõju üliõpilaste akadeemilisele sooritusele.

Cavalcanti *et al*. (2021) uuring osutab, et tõendid õppejõu töökoormuse vähenemisest on ebaühtlased — osa uuringuid leidis selge leevenduse, teised aga mitte. Autorid rõhutavad, et õppejõu roll ei kao isegi siis, kui hindamine on suures osas automatiseeritud, sest testide loomine ja tagasiside rikastamine nõuavad jätkuvalt pedagoogilist läbimõtlemist (Cavalcanti *et al*., 2021).

2.4 Enesekontrollitestid

Enesekontrolltestid on vabatahtlikud, mis soodustab õppija sisemine motivatsiooni säilitamist (Darling-Hammond & Ifill-Lynch, 2006), vältides liigset survekäsitlust. Enesekontrollitestide pedagpogiline väärtus põhineb eneseregulatsiooni ja motivatsiooni suurendamisel (Vatterott, 2010). Kohene tagasiside aitab üliõpilasel kiiresti tuvastada kontseptsioonid, mis vajavad täiendavat läbivõtmist (Vatterott, 2010).

3. Õppematerjalide arendamise protsess

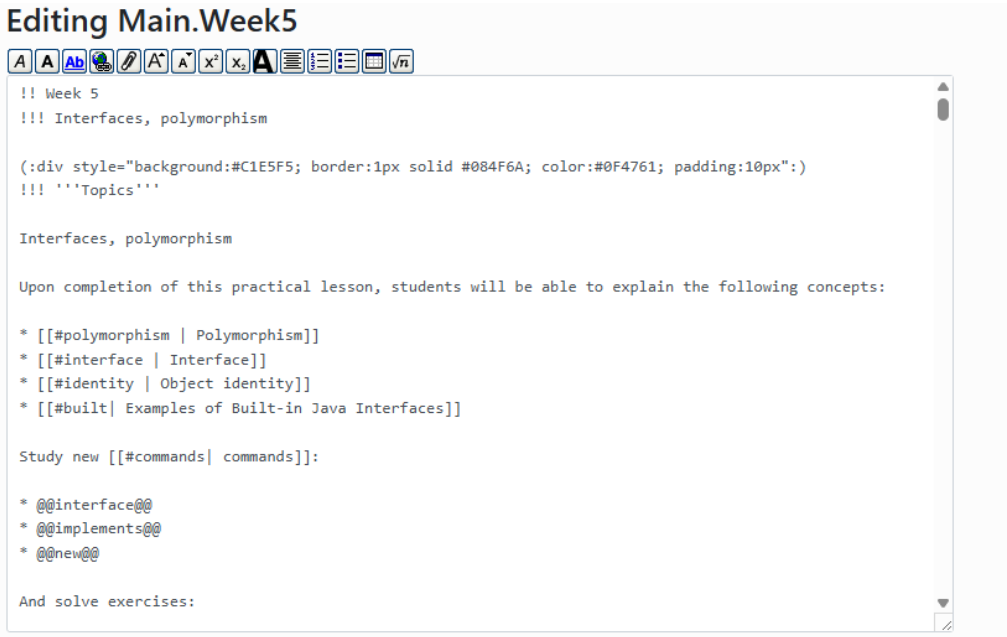
Kuna eelnevas peatükis määratleti õppematerjalide kvaliteedi parimad praktikad, keskendutakse selles peatükis materjalide väljatöötamise etappide detailsusele ja loogilisele järjestusele.

3.1 Üldine protsess

Enne uute materjalide väljatöötamist, tutvuti samanimelise eestikeelse objekt orienteeritud programmeerimise kursusega (Tartu Ülikooli õppeinfosüsteem, 2024). Pöörati tähelepanu kursuse ülesehitusele, kujundusele ja sisule. Lähtudes tähelepanekutest ja eelmisest peatükist loetelud heade praktikatest koostati uute õppemoodulite kava ja struktuur.

Järgmiseks algas õppematerjalide sisuline väljatöötamine - koostati esmaversioon tekstist. Teksti koostamisega paralleelselt töötati välja selged näited definitsioonide illustreerimiseks. Samal etapil töötati välja ka praktilised harjutused.

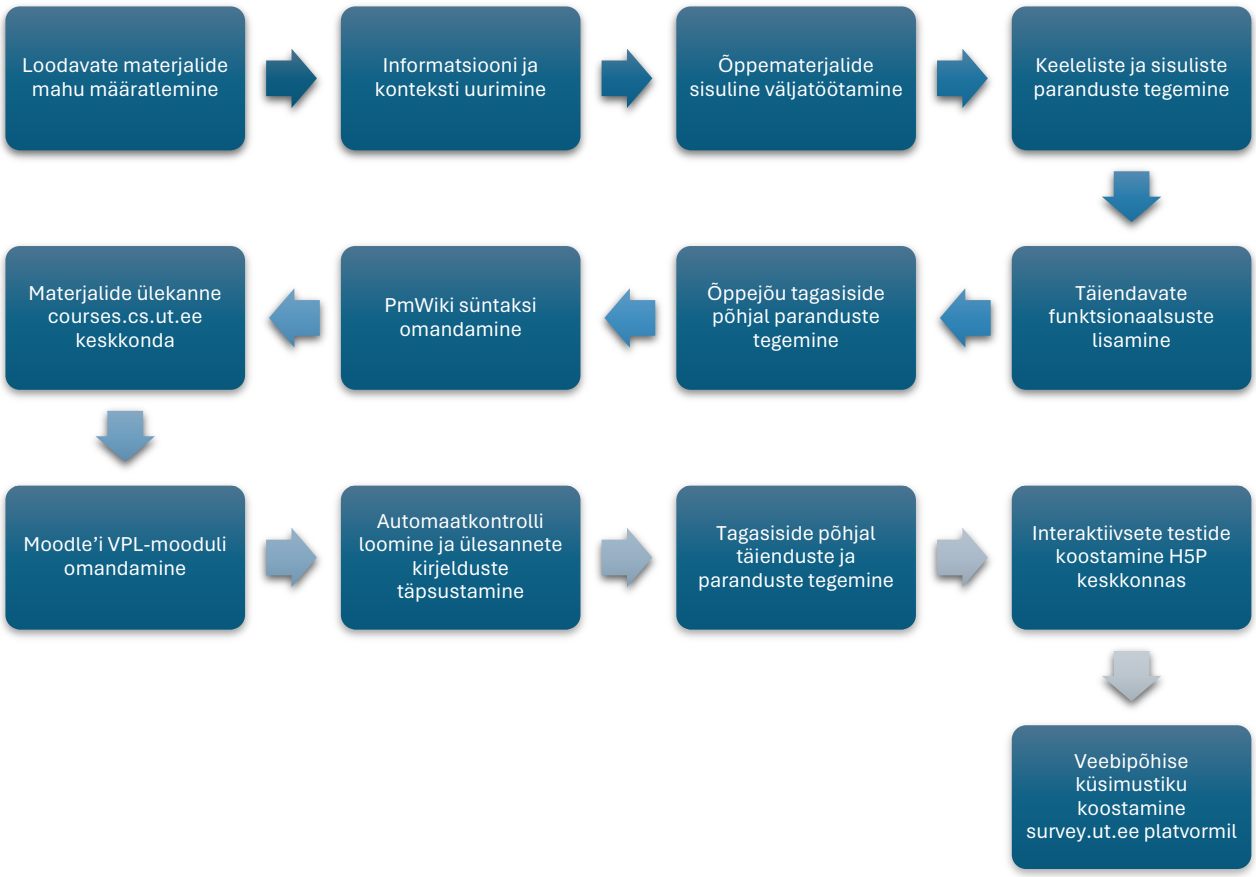
Kui materjalide esimene versioon oli valminud ja need olid juhendaja antud tagasiside põhjal toimetatud, viidi õppematerjalid üle kursuse veebikeskkonda (Lisa 2). Antud keskkonnas toimub tekstiredigeerimine PmWiki sisukirjelduskeeles (Joonis 1), mida autor pidi selgeks tegema, kuna tal puudus varasem kogemus selle keelega. Ülekanne õnnestus ning kogu planeeritud funktsioonaalsus, sh hüperlingid ja vormindus-lahendused säilisid.



Joonis 1. PmWiki sisukirjeldusekeele näide

Järgmisena prooviti koostada automaatkontrollid väljatöötatud programmeerimisülesannetele Tartu Ülikooli Moodle-s kuna seal on juba paigaldatud Virtual Programming Lab (VPL) moodul. Varasemast puudub autoril kogemus automaatkontrollide testide koostamises, seega oli vaja tutuvuda *JUnit* ja VPL-ga. Testitede koostamisel selgus, et programmeerimisülesanndeid oli vaja natuke kohandata, et testide kontroll oleks võimalikult sujuv. Seejuures ei muutunud harjutuste keerukus. Saadud juhendaja tagasiside põhjal viidi sisse vajalikud täiendused ja parandused, mis kõrvaldasid varasemad puudujäägid.

Lõpuks koostati üliõpilaste teoreetiliste teadmiste kontrollimiseks interaktiivsed enesekontrollitestid H5P mallide abil (Lisa 2) ning lisati need kursuse veebilehele. Koostatud materjalide hindamiseks töötati välja tagasiside küsimustiku survey.ut.ee platvormil (Lisa 2) ning kursuse veebilehele lisati link. Kirjeldatud protsessi (Joonis 2) kasutati mõlema nädala materjalide loomiseks.



Joonis 2. Ühe nädala jaoks koduülesannete koostamise protsess

3.2 Automaatkontrolli testide koostamise detailid

Käesolevas töös arendati iga programmeerimisülesande jaoks automaatkontroll, et üliõpilased saaksid tagasisidet kiiremini ning õppejõudude hindamistöö maht väheneks.

Automaatkontrolli testid on realiseeritud Moodle-i õpikeskkonnas Virtual Programming Lab (VPL) mooduli abil, kuna üliõpilased esitavad lahendused otse selles keskkonnas ning on varasemalt edukalt kasutatud programmeerimiskursustel (Tartu Ülikooli õppeinfosüsteem, 2024). VPL võimaldab esitatud programmi käivitada ja tulemusi automaatselt hinnata ilma täiendava platvormi juurutamiseta.

VPL-i standardpakett sisaldab juba *junit4.jar*, mistõttu testide ja skriptide ühtsus nõudis *JUnit* 4 kasutamist. Kõik testid ja testide käivitamise loogika kirjutati Java keeles. Kõik kontrollimeetodid on defineeritud *MyTest.java* failis, mis sisaldab nii lihtsaid `assert` funktsioone kui ka väljundipõhist kontrolli (`ByteArrayOutputStream`) erandite ja ebaõnnestumiste aruandluseks.

Testide käivitamise programmid on VPL-iga kohandatud *shell*-skriptid, mis laadivad sisse ühised funktsioonid (*common_script.sh*), kontrollivad Java tööriistu, koguvad lähtefailid, kompileerivad need ja genereerivad täitmise käivitaja (*vpl_execution.sh*). Näiteks *vpl_run.sh* lisab klassirajale *JUnit* 4 ja konfigureerib `org.junit.runner.JUnitCore MyTest` käsu; samas *vpl_evaluate.sh* kasutab testi käivitajana `java … Main vpl_run`. Kõik programmid on lisatud ka GitHubi repositooriumist “jimburton/vpl-unit-test” (Lisa 2), mis pakub mudeli *shell*-põhiste autotestide lihtsaks juurutamiseks Moodle-i VPL-is.

Põhiline juhend testide kirjutamiseks pärineb James Smith-i (2020) YouTube-i videost “U VPL Java + JUnit Assignment setup in Moodle.” (Lisa 2), mille alusel kohandati testide struktuuri ja skripte.

3.2 Automaatkontrolli koostamine

Käesolevas töös arendati iga programmeerimisülesande jaoks automaatkontroll, et üliõpilased saaksid tagasisidet kiiremini ning õppejõudude hindamistöö maht väheneks.

Kõik testid ja automaatkontrolli mehhanismid on realiseeritud Moodle-i õpikeskkonna Virtual Programming Lab (VPL) mooduli abil. Tegemist on Tartu Ülikoolis varasemalt edukalt rakendatud lahendusega, mis võimaldab üliõpilastel esitada lahendusi otse keskkonnas ning hinnata nende korrektsust automaatselt (Tartu Ülikooli õppeinfosüsteem, 2024).

Automaatkontrolli testid on loodud Java programmeerimiskeeles kasutades *JUnit* 4 raamistikku, mis on VPL-i vaikimisi paketis olemas. Testid on koondatud klassi *MyTest.java*, kus on defineeritud nii lihtsad `assertEquals()` tüüpi kontrollid kui ka tekstilise väljundi testid, mille puhul kasutatakse `ByteArrayOutputStream` lahendust programmi väljundi kontrollimiseks.

Testide täitmine on korraldatud *shell*-skriptide abil, mis on kohandatud VPL-i keskkonnale. Näiteks skript *vpl_run.sh* lisab klassirajale vajalikud *JUnit* teegid ning käivitab testid `org.junit.runner.JUnitCore` *MyTest* käsuga. Skript *vpl_evaluate.sh* kogub kokku tulemused, arvutab punktid ja vormindab tagasiside. Kõik skriptid põhinevad GitHubi repositooriumis “jimburton/vpl-unit-test” avaldatud mallil (Lisa 2), mille abil oli võimalik kiiresti ja tõhusalt üles seada toimiv testimissüsteem Moodle’i sees.

Automaatkontrolli süsteem on üles ehitatud nii, et iga testimeetod käivitatakse iseseisvalt ning selle tulemus põhineb konkreetsele sisendile antud väljundil. Iga testi käivitamine toimub `Main.java` klassist *try–catch* plokis, kus testi edukas läbimine lisab punkte lõpptulemusele ning tagastab märksõna “success”. Juhul kui test ebaõnnestub, püütakse kinni `AssertionError` erand ning väljastatakse täpne veateade, mis aitab üliõpilasel mõista, milline osa lahendusest vajab parandamist.

Näide try–catch ploki:

```
try {
     test.test_Exercise_1_Bus_canMakeTrip(1);
     System.out.println(formatOutput("Test Exercise 1 Bus
canMakeTrip() 1", "2.5", null));
     grade += 2.5;
} catch (AssertionError e) {
     System.out.println(formatOutput("Test Exercise 1 Bus
canMakeTrip() 1", "2.5", e));
}
```

Peamisteks kontrollimeetoditeks on `assertEquals()` kontrollid ja väljundi sobivuse võrdlus, mis on realiseeritud suunates `System.out` voog ajutiselt mällu (`ByteArrayOutputStream`) ning kontrollides seal sisalduva teksti vastavust oodatud mustrile meetodiga `contains()`.

Lisaks on ülesannete struktuuris kasutatud *switch*-lauseid, et juhtida testide järjestust ning vähendada koodikordust. Igale ülesandele on määratud 2–4 erinevat testi, mis katavad põhifunktsionaalsused. Mõne ülesande puhul, näiteks liideste kasutamisel (nädal 5), kontrolliti lisaks ka nõutavate meetodite olemasolu. Kui vajalikud meetodid puuduvad, tekib kas kompileerimisviga või kuvatakse automaatkontrolli käigus informatiivne teade puuduva meetodi kohta.

Näide ühest testist, milles testitakse sõnalisi väljundi. Koodi loetavuse parandamiseks vähendati teksti suurust ja reavaheid.

```
@Test
public void test_Exercise_4_toString(int x) {
	ByteArrayOutputStream outContent = new ByteArrayOutputStream();
	PrintStream originalOut = System.out;
	System.setOut(new PrintStream(outContent));
	String output;
	switch (x) {
		case 1:
			FullTimeEmployee fullTime1 = new
			FullTimeEmployee("Alice",
				LocalDate.now().minusYears(5), 60000, 0.02);
			System.out.println(fullTime1);
			System.setOut(originalOut);
			output = outContent.toString().toLowerCase();
			String[] outputToString = output.split("\n");
			assertEquals("With values FullTimeEmployee(\"Alice\",
					LocalDate.now().minusYears(5),
			60000,		0.02) the output should
			contain the		following words: " +
			"\"Alice\",		\"2020-04-
			11\",\"60000\", \"5500\""
			,true,
			outputToString[0].contains("alice")&&
			outputToString[0].contains(LocalDate.now().minusYears(5).
			toString()) && outputToString[0].contains("60000")&&
			outputToString[0].contains("5500")
		);
		break;
	…
	}
}
```

Testide kirjutamisel ja süsteemi seadistamisel oli oluliseks abimaterjaliks James Smithi (2020) YouTube-i video *"U VPL Java + JUnit Assignment setup in Moodle"*, mille põhjal kohandati testide struktuuri ning skripte kursuse vajadustele vastavaks.

3.3 Enesekontrolli testid H5P keskkonnas

Selle peatükki eesmärk on kirjeldada kursuse viienda ja kuuenda nädala enesekontrolliteste, mis loodi H5P vahenditega.

Uute õppematerjalide loomise protsessis jäi üks viimaseid samme just interaktiivsete testide väljatöötamine H5P mallide abil. See samm viidi ellu paralleelselt materjalide põhiversiooni valmimisega ja automaatkontrolli integreerimisega Moodle-i VPL-i kaudu. Enesekontrollitestid on lisatud kursuse veebilehele. Testide struktuur on järgmine: mõlemas testikomplektis (nädalad 5 ja 6) on kaheksa mitmikvaliku küsimust, millest igaühes on üks korrektne vastus. Küsimuste teemadering katab olulisemad objektorienteeritud programmeerimise kontseptsioone, mis kursuse viienda ja kuuenda nädalal käsitletakse:

- Nädal 5: polümorfism, liidesed, meetodite ülekatmine (overloading) ja dünaamiline sidumine;
- Nädal 6: pärilus (*extends*), abstraktsed klassid, meetodite ülekirjutamine (*overriding*), konstruktorite valik ja java.lang.*Object* klassi roll.

H5P toel oli igale küsimusele lisatud ka lühitekstiline selgitus. Vale vastuse puhul selgitatakse, miks vastus ei ole korrektne ja õige vastuse puhul tuuakse välja lisa selgitus.

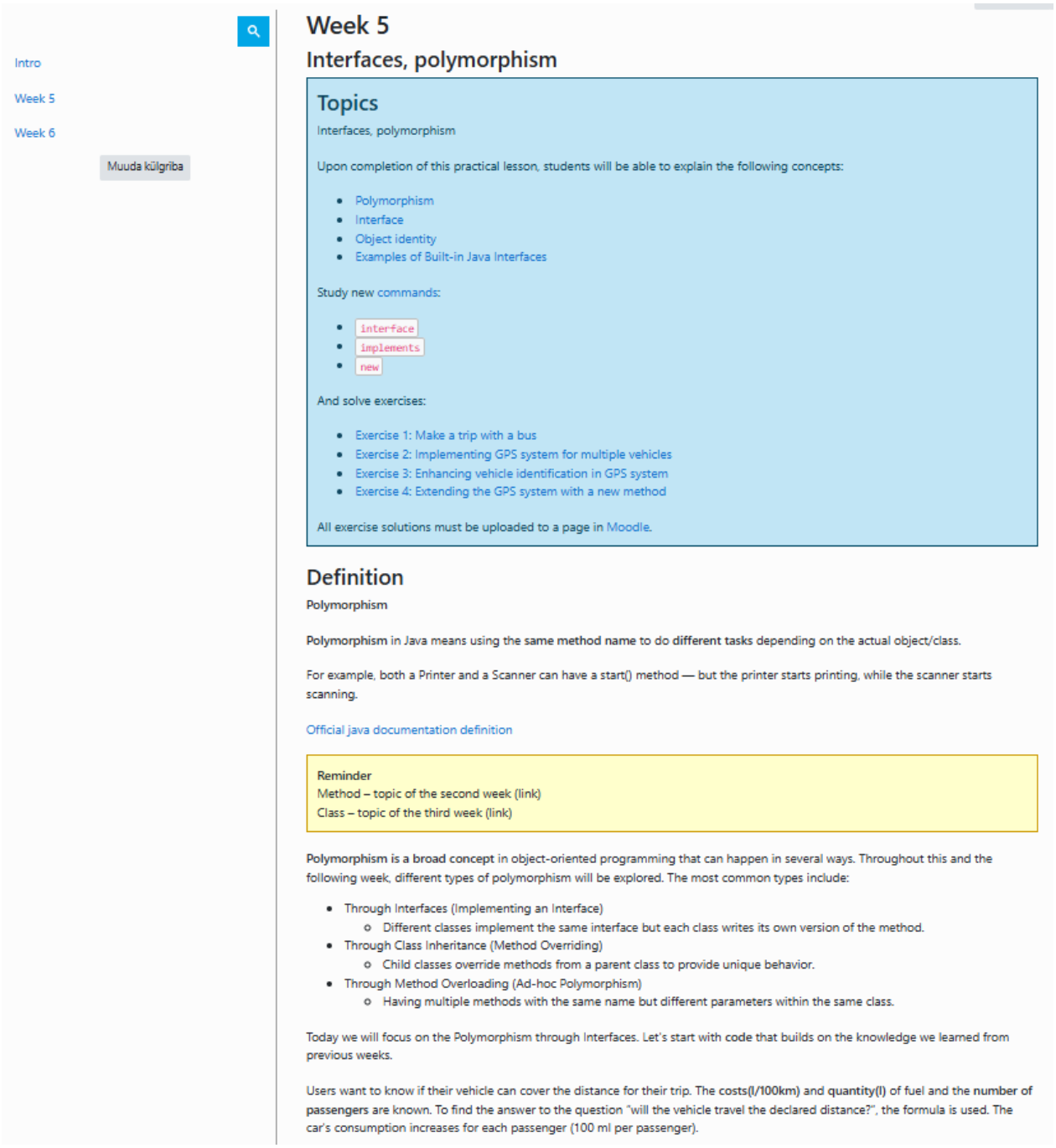
4. Valmis materjalid ja nende tagasiside

Käesolevas peatükis antakse ülevaade bakalaureusetöö käigus valminud õppematerjalidest ning nende alusel kogutud kasutajate tagasisidest. Esmalt kirjeldatakse materjalide struktuuri, sisulisi ja vormistuslikke lahendusi, mis loodi viienda ja kuuenda nädala teemade toetamiseks. Seejärel analüüsitakse küsimustiku kaudu saadud hinnanguid, tuues esile nii tugevused kui ka arenguvõimalused. Lõpuks seotakse saadud tulemused edasiste uurimissuunade ja parendusettepanekutega.

4.1 Valminud materjalid

Käesoleva bakalaureusetöö käigus loodi terviklikud interaktiivsed õppematerjalid. Kokku valmisid materjalid kahe nädala jaoks ning sisaldasid lugemismaterjale, 8 programmeerimisülesannet ja 2 enesekontrolli testi. Materjalid olid üles ehitatud nii, et need toetasid eraldi vaatluse alla võetud teemade süvendamist ja omandamist, võimaldades üliõpilasel ise harjutada ning kohe oma teadmiste taset kontrollida.

Iga nädala materjalid käsitlesid mitut teemat korraga – viienda nädala materjalides käsitleti järgmised teemasid polümorfismi, liidest, objektide identiteeti ja Java sisseehitatud liideste näited. Kuuenda nädala materjalides keskenduti pärilikkusele, klassile *Object*, meetodite ülekirjutamisele, dünaamilisele sidumisele ja abstraktsele klassile. Iga õppemooduli alguses on antud lühike kokkuvõte, eesmärgid ning lingid käsitletavatele teemadele ja programmeermisülesannetele (Joonis 3).

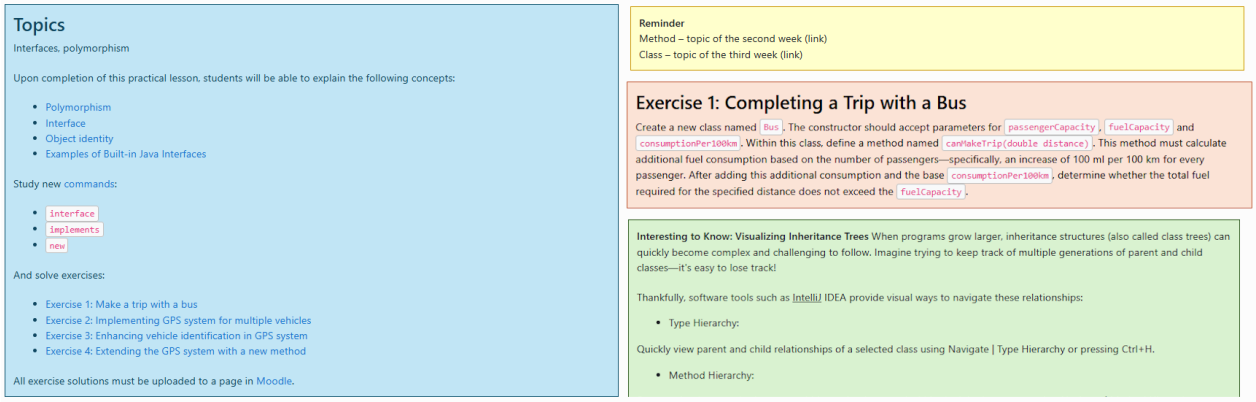


Joonis 3. Viienda nädala materjali algus, milles on nähtav ka sisukord

Joonisel 3 on ka näha, et sisukord on siniseks värvitud. See aitab vähendada kognitiivset koormust ja võimaldab tekstis kiiremini orienteeruda. Õppemoodulites oli kasutatud mitu erinevat värvi ja igal värvil on oma tähendus (kõik värvid on toodud joonisel 4):

- sinine – nädala sisu

- oranž – praktiline ülesanne
- kollane – meeldetuletus, millise nädala materjali oleks kasulik selles peatükis
- roheline – huvipakkuv lõik, kuid mitte kohustuslik õppetööks



Joonis 4. Tekstide erinevate värvide näidis

Tekstis kiiremaks ja lihtsamaks navigeerimiseks loodi ka hüperlingid, mis suunavad kasutaja vastavasse jaotisesse. Lisaks tekstisisestele hüperlinkidele kasutati linke ka konkreetsetele veebilehtedele, näiteks ametlikule Java dokumentatsioonile, mis võimaldab materjali süvendatult õppida. „*Reminder*“ osas kavandati linkide lisamine teiste nädalate materjalidele, et üliõpilane saaks kiiresti meenutada varem käsitletud teemasid.

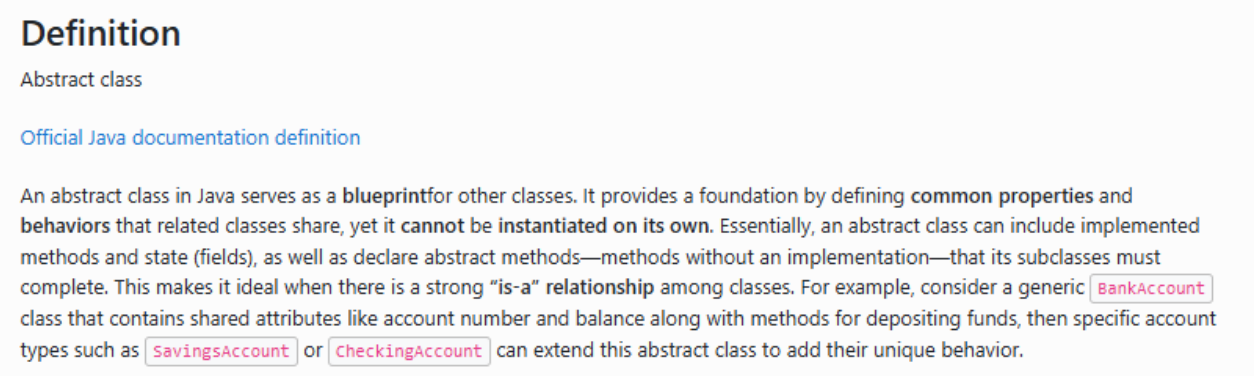
Kuna OOP kontseptsioonide juures on algajatele programmeerijatele suurimaks väljakutseks kõrge abstraktsioonitase, lisati peaaegu iga teema juurde koodinäiede, mis peaks andma üliõpilasele visuaalset tuge. Kõik koodilõigud olid vormistatud ühtemoodi, kasutades kursuse keskkonna PmWiki süntaksi: `:codestart java:` ja `(:codeend java:)` (Lisa 2). Joonis 5 illustreerib koodiploki paigutust.

```java
public class Car {
    private int passengerCapacity;
    private double fuelCapacity;
    private double consumptionPer100km;
    public Car(int passengerCapacity, double fuelCapacity, double consumptionPer100km) {
        this.passengerCapacity = passengerCapacity;
        this.fuelCapacity = fuelCapacity;
        this.consumptionPer100km = consumptionPer100km;
    }
    public boolean canMakeTrip(double distance) {
        double extraConsumption = 0.1 * passengerCapacity;
        double totalConsumptionPer100 = consumptionPer100km + extraConsumption;
        double fuelNeeded = (distance / 100.0) * totalConsumptionPer100;
        return fuelNeeded <= fuelCapacity;
    }
}
```

Joonis 5. Materjalide koodiploki näide

Teksti vormindus nõudis täiendavat pingutust. Visuaalse selguse ja lihtsuse tagamiseks rakendati järgmisi lahendusi (kõik lahendused on toodud joonistel 6, 7 ja 8):

- Tähtsate osade esiletõstmine rasvases kirjas
- Pealkirjade kasutamine suuremas fondis
- Kõigi võtmesõnade ja koodiga seotud sõnade kujundamine *monospaced* stiilis
- Kõik loetelusid olid nummerdatud või täppidega märgitud



Joonis 6. Teksti vormistuse lahendused: rasvas kirjas, suurema kirjas, monospaced stiilis teksti vormistuse

Thankfully, software tools such as IntelliJ IDEA provide visual ways to navigate these relationships:

- Type Hierarchy:

Quickly view parent and child relationships of a selected class using Navigate | Type Hierarchy or pressing Ctrl+H.

- Method Hierarchy:

See exactly where methods are overridden or implemented within the inheritance tree using Navigate | Method Hierarchy or pressing Ctrl+Shift+H.

- Call Hierarchy:

Analyze which methods invoke specific methods by selecting Navigate | Call Hierarchy or pressing Ctrl+Alt+H.

- Diagrams:

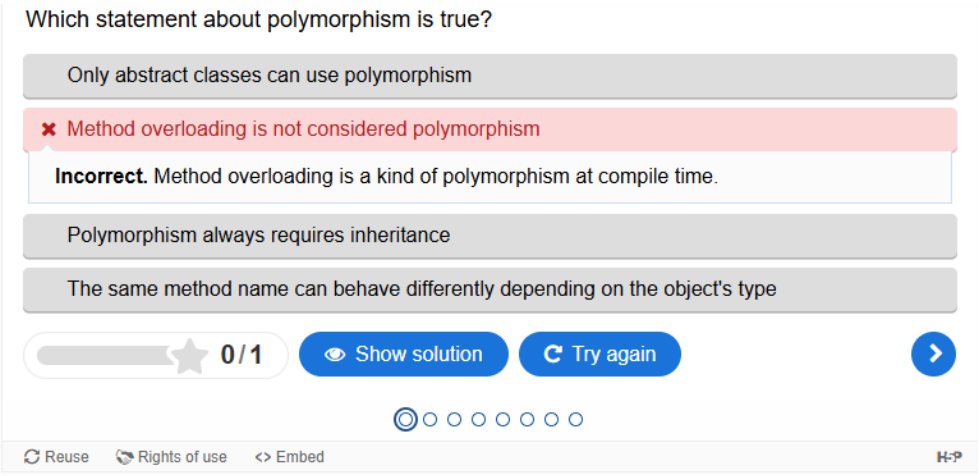
Joonis 7. Teksti vormistuse lahendused: loetelu täppidega märgitud

Remember:

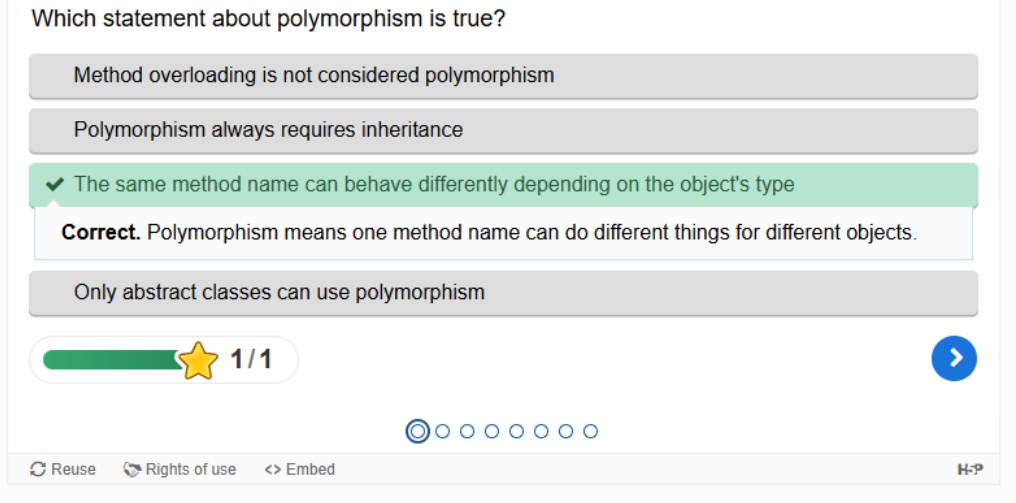
1. **Absence of Method Bodies:** An interface typically contains only method signatures—comprising method names, parameters, and return types—without any detailed implementation or method bodies.
2. **Contracts for Classes:** When a class implements an interface, it promises to write the methods (both the definition and the body) described in the interface.
3. **Grouping of Similar Methods:** Interfaces commonly group related methods, enabling different classes to adhere to a standardized set of method declarations. This ensures consistency in how these classes can be utilized.

Joonis 8. Teksti vormistuse lahendused: nummerdatud loetelud

Pärast materjalide läbimist pakuti üliõpilasele H5P-keskkonnas loodud enesekontrollitesti teadmiste kontrollimiseks (Lisa 2). Iga küsimuse vastuse juures anti selgitus: vale vastuse korral põhjendati, miks see vastus ei ole õige, õige vastuse korral täiendati teavet. Testi funktsionaalsus on näidatud joonistel 9 ja 10.



Joonis 9. Enesekontrolli testid H5P keskkonnas – vale vastus



Joonis 10. Enesekontrolli testid H5P keskkonnas – õige vastus

Näidete ja jooniste kaudu on selgelt nähtav teemade ülesehitus, navigeerimisvõimalused ning enesekontrolli testide toimimine. Autori hinnangul valmisid materjalid edukalt, arvestades teoreetilist alust teisest peatükist. Viide valmis materjalidele on lisas 3.

4.2 Tagasiside

Tagasiside kogumine toimus Tartu Ülikooli veebipõhise küsitlusvahendi LimeSurvey abil (Lisa 2), kuhu osalejad jõudsid kursuse veebilehele lisatud lingi kaudu. Kuu aja jooksul vastas küsimustikule viis inimest viienda nädala kohta ja neli inimest kuuenda nädala kohta.

Kõik vastajad andsid positiivse hinnangu loodud materjalidele: hinnati kõrgelt õppevara sisu, materjali esitust ja harjutuste selgust. Samas tõid nad välja vajaduse lisada rohkem süvitsi käsitletavaid näiteid ja täiendavaid juhendmaterjale, et toetada iseseisvat õppimist. Täpsemad vastuste ülevaated on esitatud lisas 4.

Mõlemas, nii viienda kui ka kuuenda nädala, küsitluses esitati 12 kvantitatiivset küsimust ja üks avatud küsimus täiendavate märkuste jaoks. Avatud küsimuse kaudu laekunud kommentaarid olid järgmised:

Viienda nädala avatud küsimuse kommentaarid

- kiideti materjali sisu
- soovitati rõhutada, et == kontrollib viitevõrdsust ja `.equals()` sisuvõrdsust,

- soovitati selgitada *for*-tsükli mehhanismi (massiivid vs *Iterable*)
- soovitati kaaluda objektide identiteedi käsitlemist varasemates nädalates ning liideste teemat enne polümorfismi
- soovitati ühtlustada koodiplokkide vormistust.

Kuuenda nädala avatud küsimuse kommentaarid

- toetasid ülevaatlikult IntelliJ klassihierarhia kuvamise funktsiooni kui väärtuslikku täiustust
- kiideti materjalide terviklikku ülesehitust heaks
- soovitati koodinäidetes järgida ühtseid nimetamisstandardeid (*camelCase* meetodite, *PascalCase* klasside jaoks)
- soovitati kasutada tühikuid loetavuse parandamiseks.

4.3 Edasised uurimissuunad

Pärast materjalide loomise edukat lõpetamist ja saadud tagasiside analüüsi pakutakse edasisteks uurimissuunadeks mitmeid võimalusi. Esmalt võiks õppematerjalide mahtu laiendada, lisades ingliskeelseid materjale ka teiste nädalate jaoks (nt objektid, klassid, sõned, failid, listid). Teisalt koodidemosüsteemei integreerimine kursuse veebilehele võimaldaks reaalajas koodi jooksutamist veebilehitsejas ning annaks kohese tagasiside. Kolmandaks võiks pakkuda täiendavaid raskusastme valikuid, jagades praktilisi ülesandeid hierarhiliselt „algaja“, „kesktaseme“ ja „väljakutsete“ rühmadesse ning lisades dünaamilised vihjesüsteemid. Neljandaks tasub arendada tagasiside analüütikat, luues automaatkontrolli süsteemi andmete põhjal interaktiivseid aruandeid õppejõududele ning visualiseerides õppijate sooritusi. Lõpuks on soovitatav läbi viia pikemaajalisi empiirilisi kasutajauuringuid, et hinnata õppijate õpitulemusi ja motivatsiooni interaktiivsete materjalide kasutamisel. Nende suundade rakendamine võimaldab veelgi suurendada õppematerjalide kvaliteeti, kohandatavust ja mõju õppimistulemustele ning tugevdada autori algset panust ingliskeelsete objekorienteeritud programmeerimise materjalide arendamisse.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli koostada ingliskeelsed õppematerjalid Tartu Ülikooli objektorienteeritud programmeerimise (OOP) kursuse jaoks. Töö keskendus kahe kursuse õppenädala (viies ja kuues nädal) sisu arendamisele, hõlmates olulisi teemasid nagu klassid, objektid, pärilus, polümorfism ja liidesed.

Materjalide loomisel järgiti ümberpööratud klassiruumi põhimõtteid. Koostati selged ja näidetega täiendatud teoreetilised ülevaated ning praktilised ülesanded ja enesekontrollitestid. Töö oluline osa oli automaatkontrolli testide loomine Moodle-i VPL-mooduli abil, mis kasutab *JUnit*-teste ja shell-skripte üliõpilaste esitatud lahenduste automaatseks kontrollimiseks ja kohese tagasiside andmiseks. Lisaks loodi H5P-keskkonnas interaktiivsed enesekontrollitestid koos selgitustega, mis toetavad õppijate reflektiivset õppimist ja eneseregulatsiooni.

Kuigi üliõpilaste tagasiside kogumine küsitluse kaudu ei õnnestunud soovitud mahus, selgus antud hinnangutest ja kommentaaridest, et valmis terviklik õppematerjalide komplekt, mis toetab iseseisvat õppimist, võimaldab automaatset kontrolli ning kinnistab teadmisi testide ja praktiliste harjutuste kaudu. Kõik planeeritud eesmärgid täideti edukalt.

Tulevikus võiks õppematerjale laiendada teiste kursuse teemadega, lisada erineva raskusastmega ülesandeid, kasutada veebilehitsejapõhiseid koodidemo platvorme ning rakendada automaatset analüütikat ja empiirilisi uuringuid õppimise mõju hindamiseks. Need sammud aitaksid veelgi parandada õppematerjalide kvaliteeti ja kasutajasõbralikkust.

Kasutatud kirjandus

Bench, S. W., & Lench, H. C. (2013). On the Function of Boredom. Behavioral Sciences, 3(3), 459-472. https://doi.org/10.3390/bs3030459

Bishop, J., & Verleger, M. A. (2013, June). The flipped classroom: A survey of the research. In 2013 ASEE annual conference & exposition (pp. 23-1200). https://doi.org/10.18260/1-2--22585

Cavalcanti, A. P., Barbosa, A., Carvalho, R., Freitas, F., Tsai, Y. S., Gašević, D., & Mello, R. F. (2021). Automatic feedback in online learning environments: A systematic literature review. Computers and Education: Artificial Intelligence, 2, 100027. https://doi.org/10.1016/j.caeai.2021.100027

Chen, H. R., & Hsu, W. C. (2022). Do flipped learning and adaptive instruction improve student learning outcome? A case study of a computer programming course in Taiwan. Frontiers in Psychology, 12, 768183. https://doi.org/10.3389/fpsyg.2021.768183

Codecademy. (n.d.). Search results: Object-oriented programming. Retrieved May 12, 2025, from https://www.codecademy.com/search?query=object-oriented%20programming

Cooper, H., Robinson, J. C., & Patall, E. A. (2006). Does homework improve academic achievement? A synthesis of research, 1987–2003. Review of educational research, 76(1), 1-62. https://doi.org/10.3102/00346543076001001

Coursera. (n.d.). Search results: Object-oriented programming. Retrieved May 12, 2025, from https://www.coursera.org/search?query=object-oriented%20programming%20&sortBy=BEST_MATCH

Craig, M., Smith, J., & Petersen, A. (2017, November). Familiar contexts and the difficulty of programming problems. In Proceedings of the 17th Koli calling international conference on computing education research (pp. 123-127). https://doi.org/10.1145/3141880.3141898

Darling-Hammond, L., & Ifill-Lynch, O. (2006). If They'd Only Do Their Work!. Educational Leadership, 63(5), 8-13. https://www.ascd.org/el/articles/if-theyd-only-do-their-work

Engel, D., & Kapp, C. (2017). Teaching the fundamentals of computer programming: A flipped classroom approach. Department of Computer Science, New York University. https://cs.nyu.edu/~deena/docs/Engel_Kapp_FlippedClassroomArticle_2017.pdf

Epstein, J. L., & Van Voorhis, F. L. (2001). More than minutes: Teachers' roles in designing homework. Educational psychologist, 36(3), 181-193. https://doi.org/10.1207/S15326985EP3603_4

Fernández-Alonso, R., Álvarez-Díaz, M., Suárez-Álvarez, J., & Muñiz, J. (2017). Students' achievement and homework assignment strategies. Frontiers in psychology, 8, 286. https://doi.org/10.3389/fpsyg.2017.00286

Gabbay, H., & Cohen, A. (2022). Investigating the Effect of Automated Feedback on Learning Behavior in MOOCs for Programming. International Educational Data Mining Society. https://eric.ed.gov/?id=ED624080

Haldeman, G., Robbins Bernal, J., Wydra, A., & Denny, P. (2025, February). Teaching Program Decomposition in CS1: A Conceptual Framework for Improved Code Quality. In Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (pp. 443-449). https://doi.org/10.1145/3641554.3701880

Hundley, J., & Britt, W. (2009, April). Engaging students in software development course projects. In the Fifth Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations (pp. 87-92). https://doi.org/10.1145/1565799.1565820

Keen, A., & Mammen, K. (2015, February). Program decomposition and complexity in CS1. In Proceedings of the 46th ACM technical symposium on computer science education (pp. 48-53). https://doi.org/10.1145/2676723.2677219

Lynch, R., Hurley, A., Cumiskey, O., Nolan, B., & McGlynn, B. (2019). Exploring the relationship between homework task difficulty, student engagement and performance. Irish Educational Studies, 38(1), 89-103. https://doi.org/10.1080/03323315.2018.1512889

Paulu, N. (1998). Helping your students with homework: A guide for teachers. US Department of Education, Office of Educational Research and Improvement. https://books.google.com/books?hl=en&lr=&id=M2MXajFNx2kC&oi=fnd&pg=PA1&dq=Helping+Your+Students+With+Homework&ots=1XF6wiBh_h&sig=z1dckMMuVsy5AAUhq98t4lZWmaQ

Terada, Y. (2015). Research Trends: Why homework should be balanced. Online),(https://www.edutopia.org/blog/research-trends-is-homework-effective-youki-terada), accessed on December, 8, 2016.

University of Tartu. (n.d.). LTAT.03.003: Object-oriented programming [Course page]. Retrieved May 12, 2025, from https://ois2.ut.ee/#/courses/LTAT.03.003

University of Tartu. (2024/25). LTAT.03.003: Object-oriented programming [Course page]. Retrieved May 12, 2025, from https://courses.cs.ut.ee/2025/OOP/spring

Vatterott, C. (2010). Five hallmarks of good homework. Educational leadership, 68(1), 10-15. https://www.ascd.org/el/articles/five-hallmarks-of-good-homework

Lisad

1. Peatüki 2 võtmetulemused

Kodutööde tõhusus

- Päevase kodutöömahuna on optimaalne 60–90 minutit; sellest enam kulutatud aeg toob kaasa väsimust ja ebavõrdsuse süvenemist.
- Kodutööde tõhusus sõltub ülesannete autentsusest ja tähenduslikkusest: probleemilahendus- või loovülesanded toetavad õppimist paremini kui drill-tüüpi harjutused.

Ülesande raskusaste, motivatsioon ja tulemuslikkus

- Raskusaste peab vastama õppija lähima arengu alale: liiga keerulised ülesanded suurendavad kognitiivset koormust ja vähendavad püsivust, lihtsad aga ei loo piisavat väljakutset.
- Õppija motivatsioon suureneb siis, kui ülesande kontekst on tema jaoks isiklikult relevantne ja tähenduslik, mitte lihtsalt lihtsustatud.
- Tõhusaks osutub ülesannete – selged vahesammud, realistlikud ajakavad ja juhised, mis vähendavad lootusetust ja toetavad tulemuslikkust.
- Ülesande selgus ja struktuur, sealhulgas täpne töömahu jaotamine, vähendavad kontrollivat survet ja toetavad autonoomset õppimist.

Ülesande eesmärkide ja tulemuste selgus

- Iga ülesanne peab sisaldama selget akadeemilist eesmärki (nt harjutamine, mõistmise kontroll, rakendus) ja see tuleb õppijale selgelt esitada.
- Eesmärkide nähtavaks tegemine (nt kontrollnimekirjad ja rubriigid) aitab õppijal mõista, kuidas töö seostub kursuse järgnevate teemadega.
- Ülesande kirjelduste mahu piiramine ning näidete ja visuaalide kasutamine aitab õppijal fokusseerida olulisele ja vähendab segadust.
- Õppejõu poolt esitletud selged väljundnäited suurendavad õppija „buy-in'i" ja püsivust keerukamate eesmärkide saavutamisel.

Ülesannete osadeks jaotamise tähtsus programmeerimisel

- Probleemi dekomponeerimine vähendab kognitiivset koormust.
- Dekompositsioon peab olema õpieesmärk, kus õppijad mõtestavad iga alamfunktsiooni semantilist rolli, mitte ainult lisavad uusi meetodeid.
- Hea praktikaks on ühe vastutuse printsiip, sidususe maksimeerimine ja korduskoodi elimineerimine, et edendada taaskasutust.
- Graafilised sõltuvuskaardid või mudelid toetavad metakognitiivset teadlikkust ja aitavad visualiseerida, kus funktsioonid peaksid hargnema või kokku voolama.

Automaatkontrollid

- Automaatne tagasiside (AT) võimaldab skaleeritavat ja kiiret hindamist programmeerimiskursustel ning toetab õppija eneseregulatsiooni.
- Suurem osa uuringuid näitab AT positiivset mõju õpilaste akadeemilisele sooritusele.
- Õppejõu töökoormus ei vähene: testide ülesehitus ja tagasiside sisuline rikastamine nõuavad jätkuvalt pedagoogilist läbimõtlemist.
- Populaarsed tööriistad (nt *JUnit*++ koos VPL-i *shell*-skriptidega) lihtsustavad testide loomist ja ühtlustavad tagasiside vormi.
- Õppejõu- ja õppijavaates on tagasiside kvaliteet määrava tähtsusega: põhjalik selgitav tagasiside soodustab sügavamat õppimist rohkem kui pelgalt korrektne/ebakorrektne märge.

Enesekontrollitestide roll eneseregulatsioonis ja motivatsioonis

- Enesekontrollitestid toetavad õppija eneseregulatsiooni ja aitavad kiiresti tuvastada nõrgad kohad.
- Kohene tagasiside ja selged eesmärgid suurendavad õppimise fokusseeritust ja vähendavad segadust.
- Vabatahtlikkus säilitab sisemise motivatsiooni ja soodustab iseseisvat õppimist.

2. Viidatud õppematerjalide koond

Kursuse materjalide veebikeskkond: https://courses.cs.ut.ee/

Interaktiivsed testid H5P keskkonnas: https://h5p.org/

Kõige uuem kättesaadav versioon JUnit 5.13.0-M3 07.05.2025 seisuga: https://junit.org/junit5/

GitHubi repositoorium “jimburton/vpl-unit-test”: https://github.com/jimburton/vpl-unit-test/

Smith, J. A. (2020, September 20). VPL Java + JUnit assignment setup in Moodle. [Video]. YouTube. https://www.youtube.com/watch?v=69Ma3BJYHbo

Tartu Ülikooli veebipõhiste küsimustike koostamise keskkond: https://survey.ut.ee/

3. Viited väljatöötatud materjalidele

Loodud objektorienteeritud programmeerimise kursus inglise keeles courses.cs.ut.ee keskkonnas: https://courses.cs.ut.ee/2025/oopn/spring

Loodud automaat kontrool objektorienteeritud programmeerimise kursuse jaoks: https://github.com/nikstormCh/Automaat-kontroll-Tartu-Ulikooli-OOP-kursuse-jaoks-

Tegemist on privaatse koodihoidlaga, kuna see sisaldab koostatud automaatkontrolli loogikat. Loogika avalik levitamine ei ole kursuse õppejõudude huvides. Ligipääs koodihoidlale on olemas käesoleva lõputöö autoril, retsensendil ja kursuse vastutaval õppejõul.

Loodud enesekontroll H5P keskkonnas viienda ja kuuenda nädala jaoks:

Viies nädal: https://h5p.org/h5p/embed/1527313

Kuues nädal: https://h5p.org/h5p/embed/1527329

Loodud veebipõhised küsimustikud:

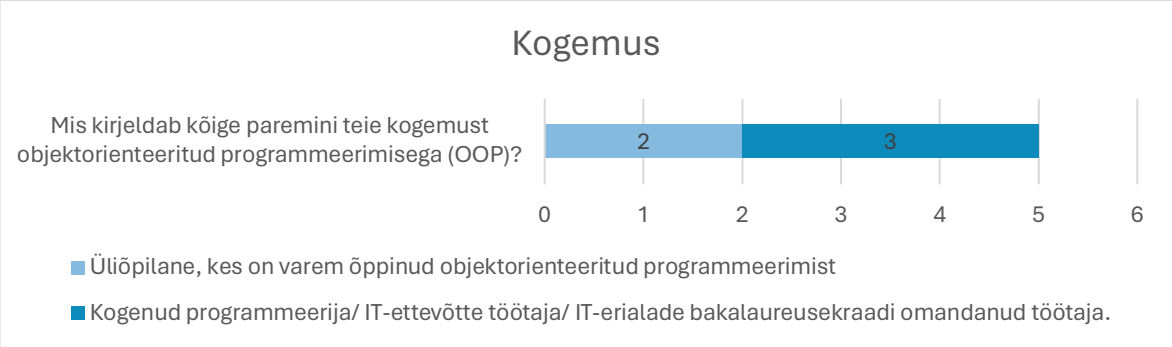
Viies nädal: https://survey.ut.ee/index.php/546817?lang=en

Kuues nädal: https://survey.ut.ee/index.php/816361?lang=en

4. Tagasiside

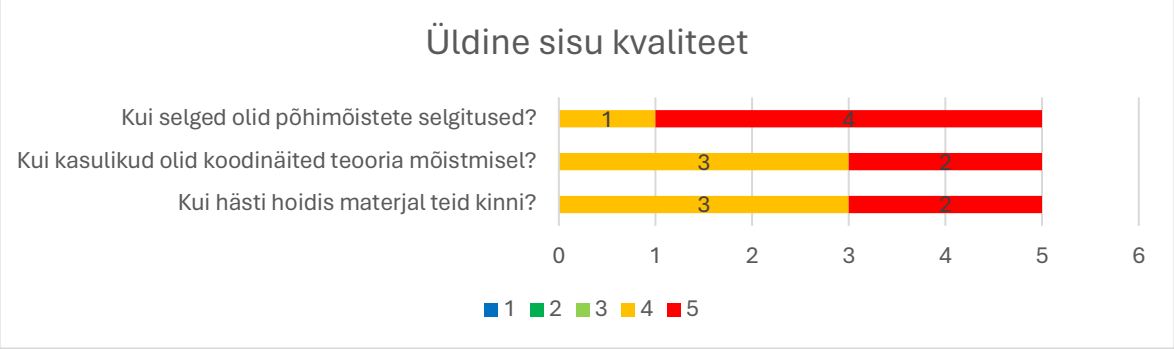
Tagasiside viienda nädala kohta

Käesolevas peatükis analüüsiti viie vastaja tagasisidet. Joonis 11 näitab, et 2 vastajat olid varasemalt läbinud OOP-õppematerjale ja 3 olid kogenud programmeerijad või IT-spetsialistid, mis viitab sellele, et õppematerjalid jõudsid nii algajate kui edasijõudnute sihtrühmani, kuid sihtrühma ühtlustamiseks võiks edaspidi kutsuda lisaks uuringusse gümnaasiumi- ja magistriõppijaid.



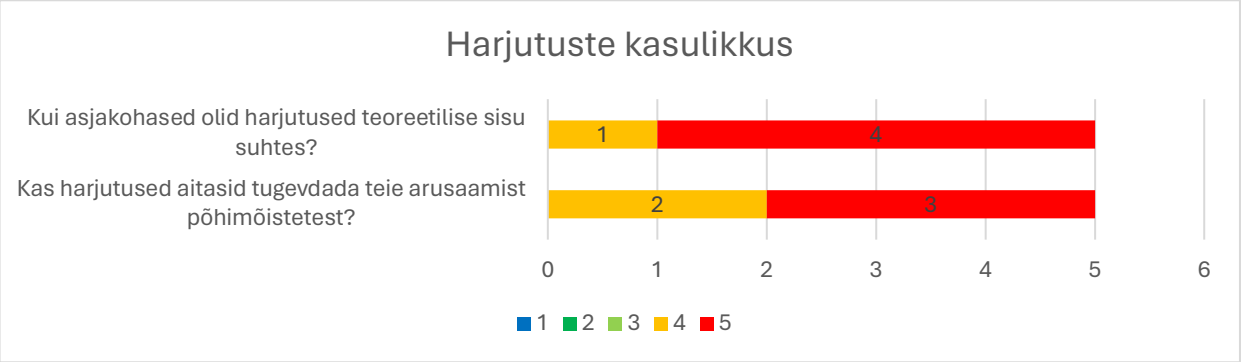
Joonis 11. Viienda nädala küsimustiku vastajate kogemus programmeerimisega

Joonis 12 kajastab üldist sisu kvaliteeti. Enamiku vastajate hinnangud haaravusele ja koodinäidete kasulikkusele olid vahemikus 4–5, mis näitab rahulolu esitletud materjaliga. Selle parandamiseks tasub lisada interaktiivseid koodidemosid (nt *live*-koodi jooksutamine veebilehitsejas), süvendada koodinäiteid ülesandepõhiste praktikumidega ning ühtlustada definitsioonide vorming, rõhutades võtmekontseptsioone diagrammide abil.

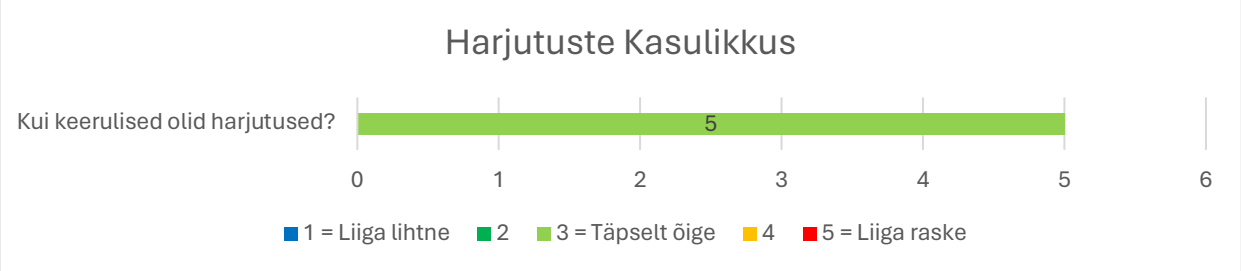


Joonis 12. Vastused viienda nädala küsimustikule materjalide haaravuse, koodinäidete kasulikkuse ja põhimõistete selguse kohta

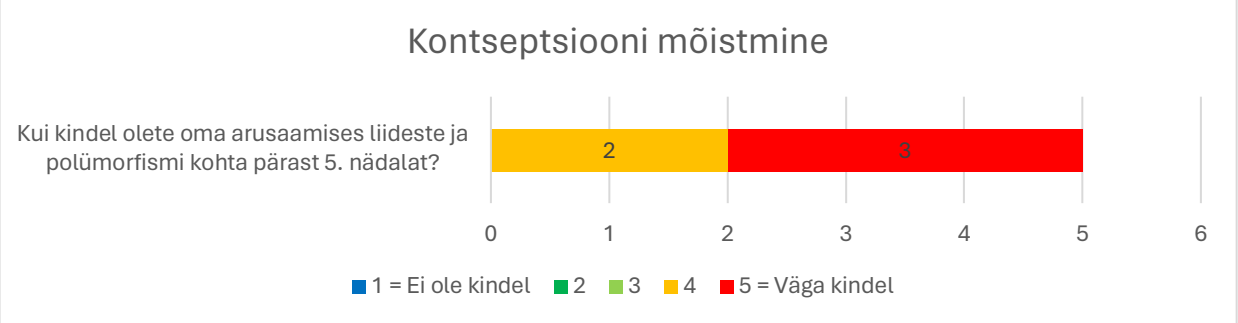
Joonis 13 näitab, et kõik viis vastajat pidasid harjutusi teooriaga kooskõlas kasulikeks (hinnang 4–5), kuid Joonis 14 kinnitab, et keerukust tajuti valdavalt „just paras“ (5 vastust): edasijõudnute motiveerimiseks võiks lisada vabatahtlikke täiendavaid väljakutseid ja algajatele lühikesi näpunäiteid. Joonis 15 andmetel tundsid enamik õppijaid end liideste ja polümorfismi osas kindlalt (hinnang 4–5), kuid Joonis 16 näitab, et polümorfismi pidas raskemaks 2 vastajat, ülejäänud ei tuvastanud erilist raskusmõistet; polümorfismi selgitamise toetamiseks võiks lisada visuaalseid skeeme.



Joonis 13. Vastused viienda nädala küsimustikule harjutuste teooriaga kooskõla kohta



Joonis 14. Vastused viienda nädala küsimustikule harjutuste keerukuse tajumise kohta

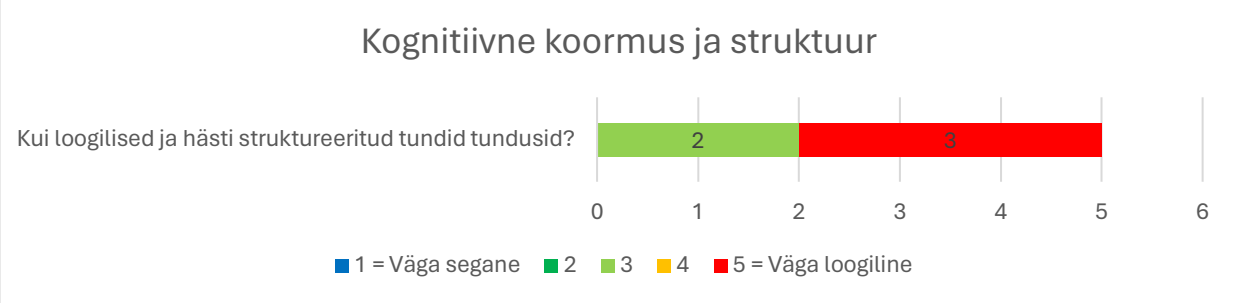


Joonis 15. Vastused viienda nädala küsimustikule liideste ja polümorfismi mõistmise enesekindluse kohta

Joonis 16. Vastused viienda nädala küsimustikule kõige raskema mõiste tuvastamise kohta

Joonis 17 näitab, et 4 vastajat pidasid uue teabe mahtu hallatavaks ja 1 mõõdukalt hallatavaks, mistõttu materjali saab paremini struktureerida – näiteks jagada lühemateks alajaotusteks ja lisada igasse kokkuvõtte või kontrollküsimused. Joonis 18 kajastab loogilist ülesehitust, mida 3 vastajat pidasid väga loogiliseks ja 2 mõõdukalt loogiliseks; selgem sisu juhtnavigatsioon (sisukord, läbiviidavad näited) aitaks nii seoseid rõhutada kui osalejat juhendada.

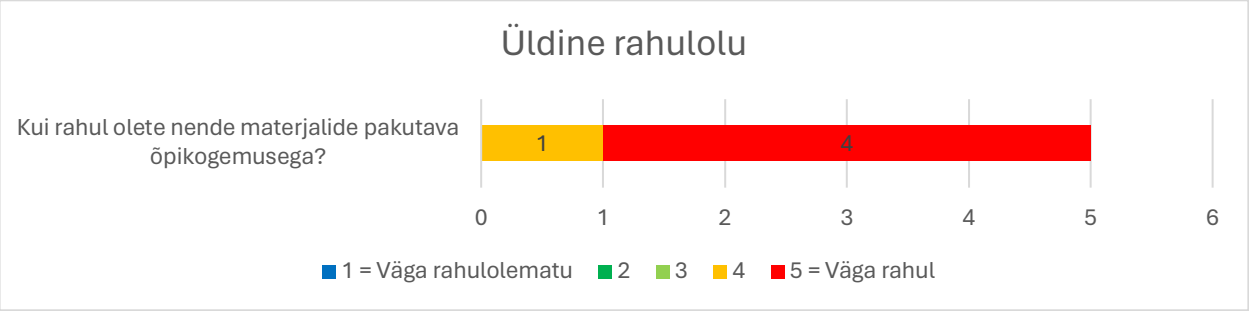
Joonis 17. Vastused viienda nädala küsimustikule uue teabe mahu tajumise kohta



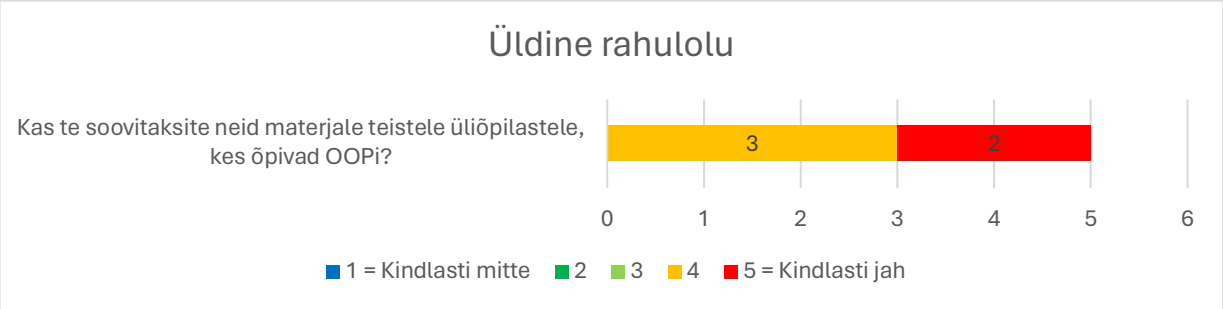
Joonis 18. Vastused viienda nädala küsimustikule materjalide loogilise struktuuri ja ülesehituse kohta

Joonis 19 järgi olid 4 vastajat materjalidega väga rahul ja 1 rahuldavaks, mis kinnitab positiivset üldmuljet; rahulolu tugevdamiseks võib lisada õppijate refleksioonivormi. Joonis 20 näitab, et 3

vastajat soovitaksid materjale kindlasti ja 2 kaaluksid proovimist enne soovitamist; soovitusaktiivsuse tõstmiseks võiks luua tunnustussüsteemi aktiivsetele tagasisideandjatele.



Joonis 19. Vastused viienda. nädala küsimustikule üldise rahulolu kohta

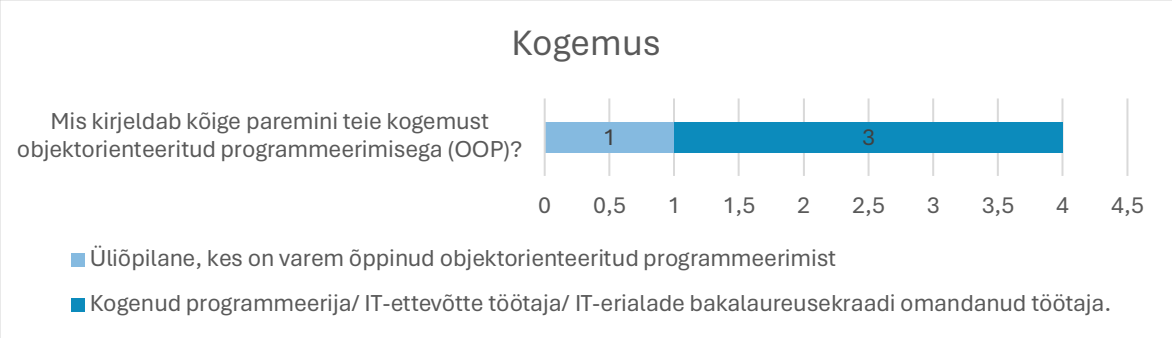


Joonis 20. Vastused viienda. nädala küsimustikule soovituse andmise valmisoleku kohta

Tagasiside kuuenda nädala kohta

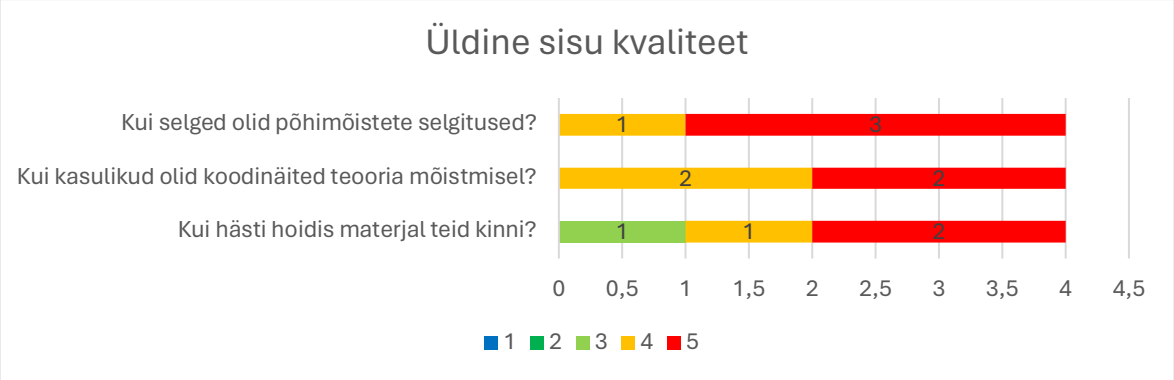
Käesolevas alapeatükis analüüsitakse kuuenda nädala õppematerjalide tagasisidet. Vastajate arv igas küsimuses oli neli, mis annab esmase, kuid mitte statistiliselt üldistatava ülevaate materjalide tugevustest ja parenduskohtadest.

Joonis 21 näitab, et üks vastaja oli üliõpilane, kellel oli varem OOP-kogemus, ja kolm vastajat olid kogenud programmeerijad või IT-spetsialistid, mis viitab sellele, et kuuenda nädala materjalid jõudsid eri tasemega õppijateni. Sihtrühma ühtlustamiseks võiks edaspidi pakkuda õppematerjale või lühikursust neile, kellel varasemat OOP-teadmisi napib, ning kutsuda osalejaid ka magistriõppest.



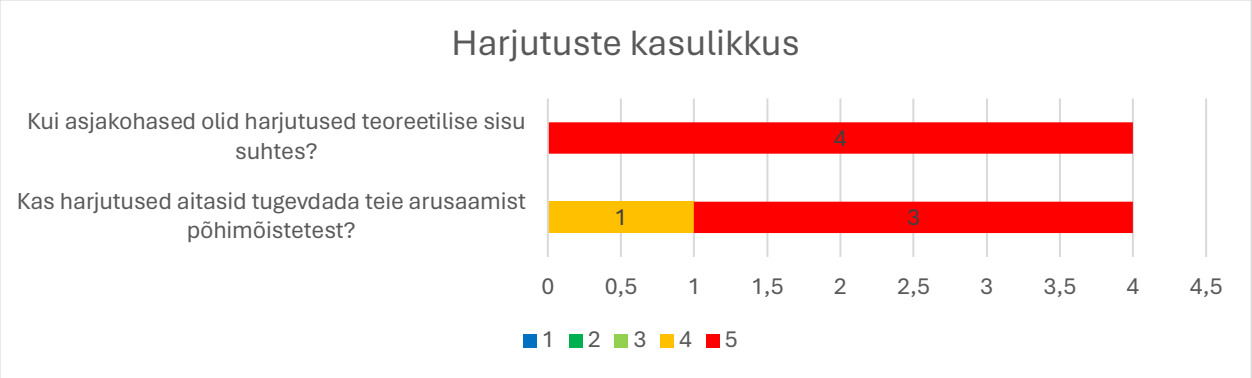
Joonis 21. Kuuenda nädala küsimustiku vastajate varasem OOP-kogemus

Joonise 22 põhjal hindasid vastajad materjali haaravust ühe hinnangu 3, ühe hinnangu 4 ja kahe hinnangu 5, koodinäiteid väärtustati hindega 4 (kaks vastust) ja 5 (kaks vastust) ning põhimõistete selgitusi hindas üks vastaja 4 ja kolm vastajat 5. Selle põhjal võiks materjali kaasahaaravust tugevdada interaktiivsete demosüsteemide ja ülesandepõhiste praktikumide kaudu ning ühtlustada definitsioonide esitlust visuaalsete skeemidega.

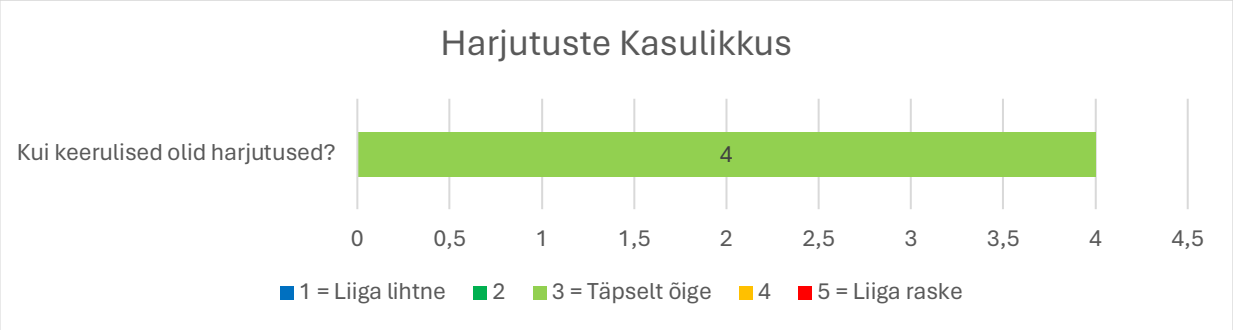


Joonis 22. Kuuenda nädala küsimustikule antud hinnangud materjalide haaravusele, koodinäidete kasulikkusele ja põhimõistete selgituste selgusele

Joonis 23 näitab, et harjutused aitasid tugevdada põhimõistete arusaamist (üks hinnang 4, kolm hinnangut 5) ja joonis 24 kinnitab, et harjutuste keerukus oli nelja vastaja arvates paras (kõik hinnang 3). Harjutuste taseme valikuks võiks luua eraldi raskusastmed (algaja, kesktase, väljakutse), mis võimaldaks õppijal valida vastavalt oma pädevusele, ning lisada lühikesed näpunäited või vihjed, et toetada vähem kogenud kasutajaid.

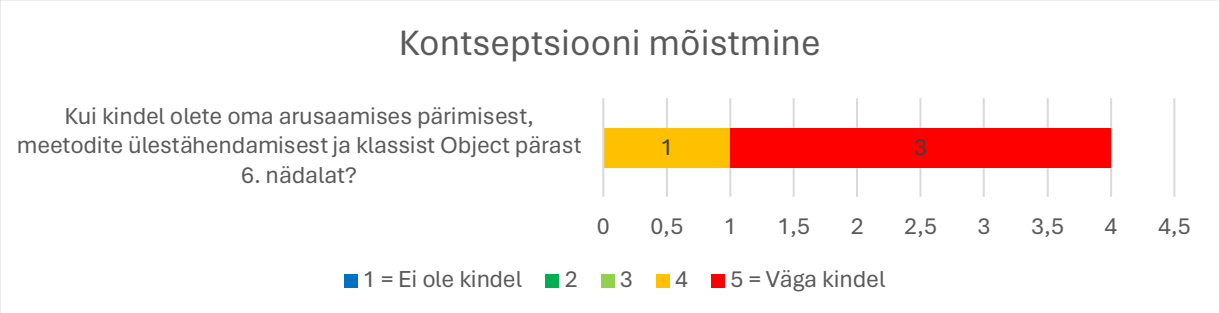


Joonis 23. Kuuenda nädala küsitluse vastajate hinnangud harjutuste teooriaga kooskõla kohta

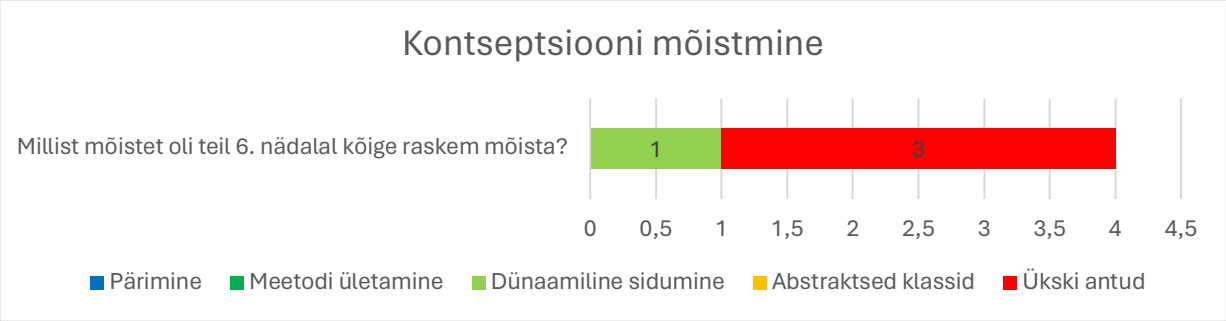


Joonis 24. Kuuenda nädala küsitluse vastajate tajutud harjutuste keerukus

Joonis 25 kajastab kontseptsioonide – pärimise, meetodite ülekirjutamise ja klassi *Object* – mõistmise enesekindlust, mida hinnati hindega 4 (üks vastus) ja 5 (kolm vastust). Joonis 26 näitab, et üks vastaja pidas kuuendal nädalal kõige keerulisemaks dünaamilist sidumist, kolm vastajat ei tuvastanud mingeid eriliselt raskusi. Päringu selgitamiseks võiks lisada visuaalseid skeeme ja interaktiivseid viktoriine, mis aidaksid kinnistada dünaamilise sidumise loogikat.

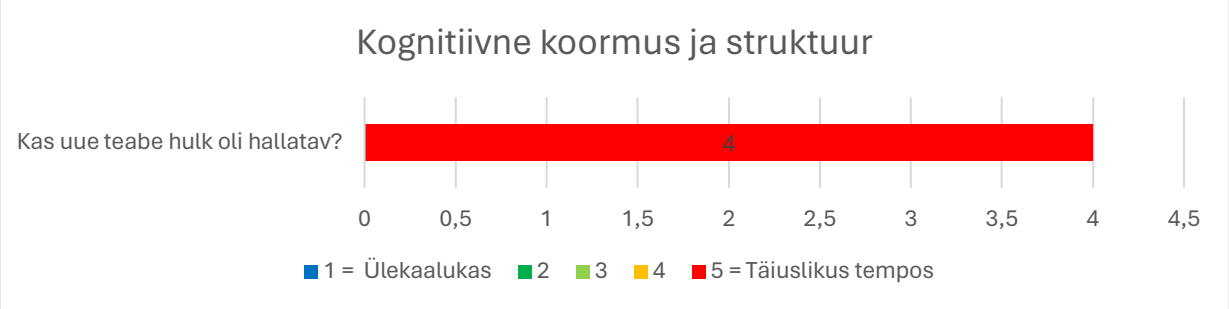


Joonis 25. Kuuenda nädala küsitluse vastajate enesekindlus pärimise, meetodite ülekirjutamise ja klassi Object kontseptsioonide mõistmisel

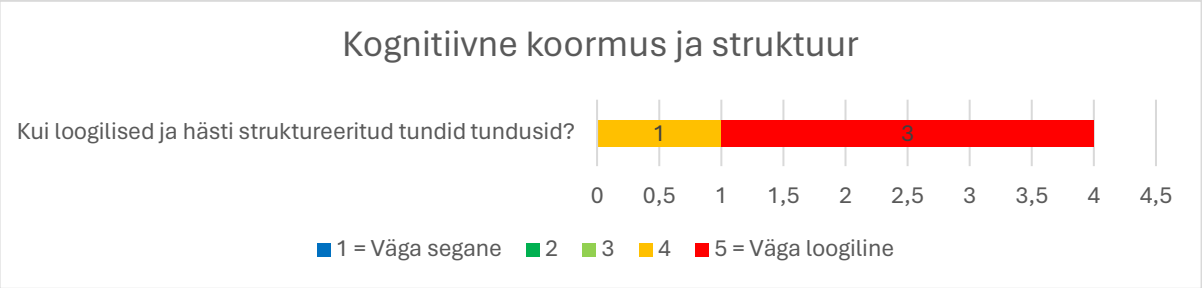


Joonis 26. Kuuenda nädala küsimustikule vastajate arvamine kõige keerulisemate kontseptsioonide kohta (dünaamiline sidumine)

Joonis 27 alusel pidasid kõik neli vastajat uue teabe mahu haldavalt piisavaks (hinnang 5), ent materjali struktuuri võiks veelgi selgemaks muuta, jagades õpiüksuse lühemateks alamsektsioonideks ja lisades igasse kokkuvõtte või kontrollküsimused. Joonis 28 näitab, et üks vastaja leidis ülesehitust mõõdukalt loogilisena (hinnang 4) ja kolm vastajat väga loogilisena (hinnang 5), mistõttu navigeerimisjuhte, näiteks läbivaid näiteid ja sisukorda, võiks selgemini esile tõsta.

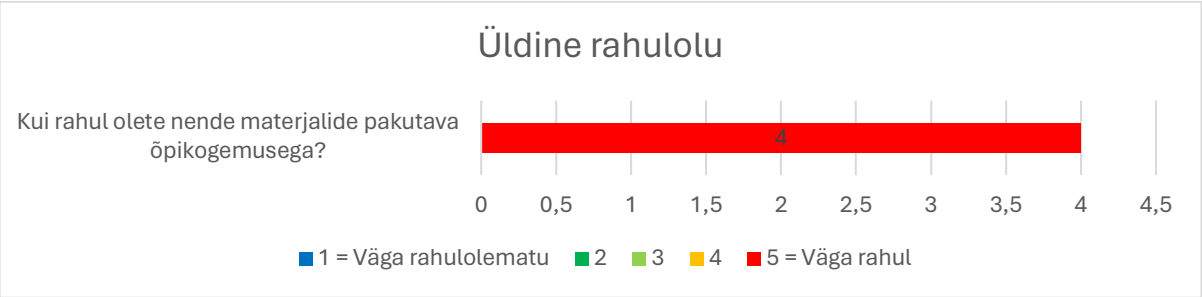


Joonis 27. Kuuenda nädala küsitluse vastajate hinnangud uue teabe mahu hallatavusele

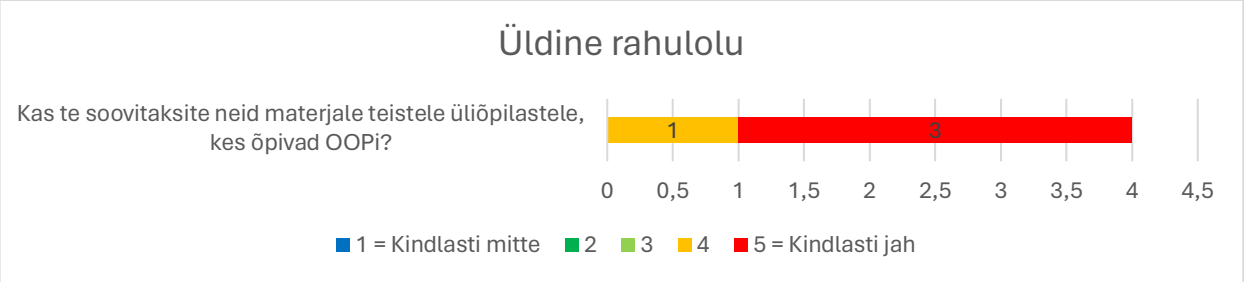


Joonis 28. Kuuenda nädala küsitluse vastajate hinnangud materjalide loogilisele struktuurile ja ülesehitusele

Joonis 29 kinnitab, et neli vastajat olid materjalidega väga rahul (hinnang 5), ja joonis 30 näitab, et üks vastaja soovitaks materjale teistele õppijatele hindega 4 ning kolm vastajat hindega 5. Soovitusteaktiivsuse tõstmiseks võib uurida tunnustussüsteemi rakendamist, näiteks virtuaalsete märgiste või lühikursuse lõpetamise tõendite kaudu.



Joonis 29. Kuuenda nädala küsitluse vastajate üldine rahulolu materjalidega



Joonis 30. Kuuenda nädala küsitluse vastajate valmisolek soovitada materjale teistele üliõpilastele

5. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Nikita Chernov**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **"Ingliskeelsete õppematerjalide koostamine ainele "Objektorienteeritud Programmeerimine""**, mille juhendaja on **Ljubov Jaanuska**, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;
2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi 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.

Nikita Chernov
15.05.2025