TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Andra Laura Meeksa

Õppevideote rakendamine programmeerimise kursustel

Bakalaureusetöö (9 EAP)

Juhendaja: Eno Tõnisson, PhD

Tartu 2018

Õppevideote rakendamine programmeerimise kursustel

Lühikokkuvõte:

Antud bakalaureusetöös keskenduti õppevideote rakendamisele programmeerimise kursustel, täpsemalt õppevideote kasutamise eesmärkidele ning õppevideote struktuurile ja informatsioonidisainile. Samuti toodi välja õppevideote võrdlus traditsiooniliste programmeerimise õpetamise ja õppimise vahendite ning materjalidega. Töös uuriti programmeerimise õppejõudude kokkupuudet õppevideotega ning nende suhtumist õppevideotesse. Käesoleva töö raames valmisid *Objektorienteeritud programmeerimise* kursuse jaoks Java andmestruktuuride ning geneeriliste meetodite, klasside ja liideste teemal kaks videonäidet. Pärast õppevideote loomist uuriti ka üliõpilaste tagasisidet loodud videonäidetele ning nende suhtumist õppevideote kasutamisse.

Võtmesõnad:

Programmeerimise õppevideod, Objektorienteeritud programmeerimine, informatsioonidisain

CERCS: P175, Informaatika, süsteemiteooria

Application of Videos in Programming Courses

Abstract:

The purpose of this thesis is to focus on applying videos in programming courses, specifically the various purposes of using videos, including their structure and information design. Additionally, videos as study materials are compared to traditional programming teaching tools and materials. Programming lecturers' experience with videos and their opinion regarding the application of videos as study aids are subsequently analyzed. As an integral part of the thesis, two videos were created for the course of Object-oriented Programming, covering Java data structures and generic methods, classes and interfaces. Eventually, feedback from the students regarding these videos and their attitude towards the application of videos in the study process were collected and analyzed.

Keywords:

Videos in programming, Object-oriented Programming, information design

CERCS: P175, Informatics, systems theory

Sisukord

1. Sissejuhatus

Üha enam kasutatakse programmeerimise õpetamiseks ja õppimiseks tavakursustel ning vaba juurdepääsuga e-kursustel erinevaid õppevideoid. Ka Tartu Ülikooli programmeerimise kursustel kasutatakse õppevideoid, milleks on üldjuhul loengute videod ehk loengute salvestised. Tartu Ülikoolis on programmeerimise praktikumides levinud peamiselt tekstina esitatud juhendid, kuid teatud määral kasutatakse ka õppevideoid. Kõige rohkem on Tartu Ülikoolis programmeerimise õppevideod kasutusel MOOCidel ehk vaba juurdepääsuga e-kursustel [1] ja nende materjalidel põhinevatel tavakursustel. Varasemalt on õppevideoid loodud ka lõputöö raames, kui 2013. aastal kirjutas M. Gaiduk Tartu Ülikooli arvutiteaduse instituudis magistritöö „Videojuhised programmeerimise aluste kursustel“, kus keskenduti eelkõige videojuhiste tehnilisele keerukusele ja ajakulule ning loodi 15 videojuhist *Programmeerimise alused* kursusele [2].

J. Bennedsen ja M. E. Caspersen väidavad oma uuringus, et traditsioonilised õpetamise vahendid ja materjalid, näiteks tekstimaterjalid, tahvlil programmeerimine, slaidide näitamine jne, ei ole piisavad programmeerimise õpetamiseks ja õppimiseks. Sellised meetodid on kasulikud protsessi tulemuse näitamiseks, kuid protsessi ennast on selliste meetodite abil keeruline näidata [3]. Õppevideod on üliõpilasele hea võimalus materjali läbi töötada endale sobival ajal endale sobivas kohas [4]. Teisalt on õppevideote koostamine video ettevalmistamise, filmimise ja monteerimise mahu tõttu aeganõudev töö. Õppevideote loomisel ja kasutamisel tuleb arvestada mitmesuguseid aspekte, mis võivad sõltuvalt allikast märkimisväärselt erineda.

Käesoleva bakalaureusetöö eesmärk on välja selgitada õppevideote loomise ja kasutamise võimalused programmeerimise õpetamisel ja õppimisel, uurida programmeerimise õppejõudude ja üliõpilaste suhtumist õppevideotesse ning luua kaks õppevideot, et saada isiklik kogemus programmeerimise õppevideo loomise keerukuse ning ajakulu kohta.

Bakalaureusetöö sissejuhatusele järgnevas teises peatükis antakse kirjanduse põhjal ülevaade õppevideote liigitusest (loengu video, videoloeng, videojuhis ja videonäide). Samuti käsitletakse õppevideote võimalikke eesmärke programmeerimise õpetamisel ja õppimisel ning võrreldakse traditsioonilisi programmeerimise vahendeid ja materjale õppevideotega. Kolmandas peatükis on välja toodud õppevideote struktuur ning informatsioonidisaini elemendid, mida on kasulik õppevideote loomisel kasutada.

Antud töö neljandas peatükis antakse küsitluse põhjal ülevaade programmeerimise õppejõudude kokkupuudetest õppevideotega ning nende suhtumisest õppevideote rakendamisse. Viiendas peatükis käsitletakse käesoleva töö käigus valminud õppevideote loomise protsessi. Lõpetuseks antakse kuuendas peatükis ülevaade üliõpilastelt saadud tagasisidest loodud videonäidetele ning nende suhtumisest õppevideote kasutamisse. Lisades on välja toodud küsimustikud, mille abil saadi teada õppejõudude ja üliõpilaste arvamus ning õppevideote loomiseks kasutatud käsikirjad.

2. Õppevideod programmeerimise kursustel

Õppevideoid kasutatakse programmeerimise kursustel suhteliselt palju. Õppevideod on kasutusel ka Tartu Ülikooli arvutiteaduse instituudis, kuid seda peamiselt loengute salvestiste näol. Programmeerimise õpetamiseks ja õppimiseks on võimalik kasutada erinevaid õppevideote liike ning õppevideoid saab kasutada erinevate eesmärkide tarvis. Käesolevas peatükis on välja toodud erinevad õppevideote liigid, programmeerimise õppevideote kasutamise eesmärgid ning õppevideote võrdlus traditsiooniliste programmeerimise õpetamise vahendite ja materjalidega.

2.1 Õppevideote liigid

Õppevideoid on võimalik liigitada erinevate omaduste põhjal. Üheks võimaluseks on jaotada õppevideod vastavalt salvestamise viisile ning õppevideo eesmärgile. Selline jaotus on kasutusel ka käesolevas töös. Tulenevalt ühise allika kasutamisest on käesolev jaotus väga sarnane M. Gaiduki magistritöö „Videojuhised programmeerimise aluste kursustel“ jaotusega. Salvestamise viisi ning eesmärgi põhjal jaotatuna on antud töö kontekstis olulised õppevideote liigid järgmised [5]:

- loengu video - videosalvestis sündmuse arhiveerimiseks lisamaterjalideta või lisamaterjalidega (näiteks loenguslaidid). Loengut ei peeta video salvestamise eesmärgil;
- videoloeng - ettekande salvestis erinevate lisamaterjalidega (näiteks loenguslaidid). Videoloengu eesmärgiks on õppematerjali iseseisev omandamine;
- videojuhis - instruktsioon tegevuse jäljendamiseks, mis on harilikult ekraanisalvestise vormis;
- videonäide - õppematerjali sees olev videolõik, mis toetab ja visualiseerib tekstimaterjali (harilikult ekraanisalvestise vormis).

Loengu video pikkuseks on tavaliselt ühe loengu kestus (Tartu Ülikoolis ~1,5 tundi). Videoloengu, videojuhise ning videonäite kestus sõltub vastava video materjali mahust. Õppevideote loomisele kuluv aeg on samuti varieeruv. Loengu video loomine võtab üldiselt sama palju aega kui kaua kestab loeng, mida salvestatakse, kuid sinna võib lisanduda ka monteerimisele kuluvat aega (näiteks pauside eemaldamine). Loengu video ettevalmistusele kulub samuti aega, kuid loengu ettevalmistus toimuks ka siis, kui loengust salvestist ei tehtaks. Videoloengu, videojuhise ja videonäite puhul sõltub video loomisele kuluv aeg

ettevalmistatava ja filmitava materjali mahust ning monteerimise kiirusest ja vilumusest, kuid ka lühema õppevideo loomisele kulub minimaalselt paar tundi.

2.2 Õppevideote eesmärgid programmeerimise õpetamisel ja õppimisel

Õppevideoid on võimalik kasutada erinevatel eesmärkidel. Vastavalt video eesmärgile võib neid erinevalt struktureerida ning video pikkus varieerub sõltuvalt eesmärgist ning materjali mahust. Üldiselt võimaldab õppevideo üliõpilastele kiirelt teemat selgitada olukorras, kus paberil seletamiseks võib vaja minna mitut lehekülge [4]. Õppevideo loob üliõpilasele võimaluse videot korduvalt vaadata, kui üks kord vaadates midagi segaseks jääb. Programmeerimises on samuti mitmeid erinevaid eesmärke õppevideote kasutamiseks õppetöös.

Levinud on olukorrad, kus programmeerimise algtaseme kursusel osalevad väga erineva tasemega üliõpilased. Õppevideod pakuvad võimaluse taset ühtlustada, kui enne auditoorset õppetööd on üliõpilastel vaja vaadata ühte või mitut õppevideot. Kui üliõpilased vaatavad videot enne auditoorsel õppetööl osalemist, saab õppejõud kindlustada olukorra, kus kõikidel üliõpilastel on olemas vähemalt olulisemad eelteadmised kursusega alustamiseks. Samuti välistab see üliõpilase jaoks olukorra, kus üliõpilane osaleb auditoorsel tööl ning tema motivatsiooni kahandavad puuduvad eelteadmised [4].

Õppevideote abil on üliõpilastel ka võimalik enne loengut või praktikumi lihtsamad põhitõed selgemaks saada. Selline õppevideote rakendamine annab õppejõule võimaluse käsitleda loengus või praktikumis juba algusest peale keerulisemaid ning väljakutsuvamaid teemasid ning ülesandeid, toetudes video abil üliõpilastele juba eelnevalt õpetatud põhitõdedele. Tulemusena on ka loeng või praktikum üliõpilaste jaoks tõenäoliselt põnevam [6].

Levinuim õppevideote rakendusviis Tartu Ülikoolis on üliõpilastele alternatiivse õppevõimaluse pakkumine või auditoorselt õppetöölt puudumise mõju vähendamine. Sellisel juhul kasutatakse üldiselt videoloengut või loengu videot, kuid on võimalik kasutada ka videoloengu asemel ühte või mitut videonäidet või videojuhist. Videoloeng katab toimunud või ära jäänud loengus käsitletud teemasid. Loengu video võimaldab erinevatel põhjustel puuduvatel üliõpilastel loengust ikkagi osa saada. Juhul, kui õppevideod jäävad semestri lõpuni kättesaadavaks, annab see üliõpilastele ka hea võimaluse eksamiks või kontrolltööks õppides materjali kinnistamiseks õppevideoid vaadata [6] [7].

MOOCidel (ingl k *massive online open course*) ehk interneti vahendusel piiramatule arvule osavõtjatele kättesaadavatel e-kursustel õpetamiseks ja õppimiseks kasutatakse enamjaolt õppevideoid, kus tihtipeale on terve kursus õppevideote vormis. Tuntuimad MOOCide

pakkujad on Coursera, Udacity, edX, Khan Academy jne [8]. Tartu Ülikool pakub kolme programmeerimise teemalist MOOCi – *Programmeerimise alused*, *Programmeerimise alused II* ning *Programmeerimisest maalähedaselt*. Tartu Ülikooli programmeerimise MOOCidel on õppevideod kasutusel, kuid kursused on selliselt üles ehitatud, et neid on võimalik ka õppevideoid vaatamata läbida [1].

2.3 Traditsiooniliste programmeerimise õpetamise ja õppimise vahendite ja materjalide võrdlus õppevideotega

Üliõpilastele programmeerimise õpetamiseks on erinevaid vahendeid ja materjale. Bennedsen ja Caspersen on traditsiooniliste programmeerimise õpetamise ja õppimise vahendite ja materjalidena välja toonud tahvlil programmeerimise, eelnevalt valmiskirjutatud programmilõikude näitamise loenguslaididel ja praktikumis või loengus programmeerimise [3]. Kõikidel traditsioonilistel vahenditel ja materjalidel ning õppe-videotel on omad positiivsed ja negatiivsed küljed, mis on järgnevalt välja toodud Bennedseni ja Casperseni loodud materjalide põhjal.

Tahvlil programmeerimist kasutatakse üldiselt tahvlipraktikumides ning veidi ka programmeerimise praktikumides, kujutades endast õppejõu poolt käsitsi programmikoodi tahvlile kirjutamist (kriiti või tahvlimarkerit kasutades). Sellise meetodi eeliseks on võimalus üliõpilastega samaaegselt arutelu pidada ning vastavalt olukorrale erinevalt koodi kirjutada. Samuti ei ole sellisel juhul tempo üliõpilastele liiga kiire. Antud lähenemise negatiivseteks külgedeks on piiratus ehk kirjutatud koodi ei saa jooksutada ning see ei anna üliõpilastele võimalust kogeda arenduskeskkonna kasutamist.

Loengutes kasutatakse sagedaselt loenguslaide ning programmeerimise kursustel sisaldavad loenguslaidid tihtipeale koodilõike. Eelnevalt valmiskirjutatud programmilõikude kuvamine loenguslaididel on võimalus näidata üliõpilastele suuremaid ja keerulisemaid programme. Sellisel juhul on õppejõul võimalik programmikoodi kirjeldada ning ka väljundit ennustada. Negatiivse küljena võib tekkida aga olukord, kus õppejõud kipub kiirustama ning samuti ei saa üliõpilased osa programmikoodi kirjutamise protsessist.

Auditoorse töö käigus programmeerimine arvutit ja projektorit kasutades on sümbioos tahvlil programmeerimisest ja loenguslaididest, mille puhul lisandub võimalus koodi jooksutada ja testida ning selline meetod näitab üliõpilastele ka programmeerimisvahendite, näiteks IDE (ingl k *integrated development environment*) ehk arenduskeskkonna, kasutamist. Antud variant on kõige sarnasem programmeerimise protsessile. Ka sellisel

meetodil on omad tagasilöögid – auditoorsel tööl on aeg piiratud ning see piirab omakorda demonstreeritava koodi keerukust ning koodi kirjutamise protsessi näeb vaid auditoorsel tööl osaledes. Puudub võimalus koodikirjutamise protsessi hiljem uuesti näha, näeb vaid tulemust ning lisaks enda kirjutatud kommentaare.

Eelnevalt selgitatud traditsioonilised programmeerimise meetodid on kõik ühekordsed sündmused, mis ei anna üliõpilasele võimalust läbitud protsessi uuesti näha, toetuda saab mälupildile ning kirjapandud kommentaaridele. Juba korra nähtud protsessi uuesti nägemiseks annab hea võimaluse õppevideote kasutamine õppetöös. Video kasutamine programmeerimise õpetamise materjalina on väga sarnane auditoorsel tööl programmmeerimisega. Erinevus seisneb programmeerimise auditoorsel tööl peamiste negatiivsete külgede puudumises. Videos saab kasutada nii palju aega kui vaja ning võib käsitleda ka keerulisemat teemat. Üliõpilasel on võimalus videot mitu korda üle vaadata, kui süvitsi arusaamiseks ühest korrast ei piisa. Ka õppevideotel on omad negatiivsed küljed - üliõpilastega arutelu puudumine ehk kui üliõpilasel tekib video jooksul küsimus, siis ei ole tal võimalust kohe küsimust esitada ja video läheb omas tempos edasi. Samuti võib õppevideot filmivale õppejõule olla häirivaks teguriks publiku puudumine.

2013. aastal tegid J. H. Sharp ja L. A. Schultz uuringu, kus 35-lt C# programmeerimiskeele sissejuhataval kursusel osalevalt üliõpilaselt küsiti tagasisidet õppevideo kasutamise kohta. Antud uuringu raames loodi kursusele videonäited vastavalt igale programmeerimise õpiku peatükile (üks video oli umbes 20 minutit pikk). Uuringus osalesid nii statsionaarses kui ka veebipõhises õppes olevad üliõpilased. Antud uuringu tulemustest selgus, et ajaliselt vaadati õppevideoid rohkem kui loeti õpikut. Samuti selgus, et 82,9% uuringus osalejatest eelistasid õppevideoid õpikule [9].

Üliõpilastele kasuliku õppevideo loomiseks on seejuures väga oluline rakendada õppevideotele kohast struktuuri, mille elemente tutvustatakse järgmises peatükis.

3. Õppevideote struktuur ja informatsioonidisain

Antud peatükis tuuakse välja õppevideote struktuur lähtuvalt videojuhiste struktuurist, mida on hea kasutada ka erinevat liiki programmeerimise õppevideote loomisel. Samuti tutvustatakse informatsioonidisaini elemente, mida on kasulik õppevideoid tehes silmas pidada.

3.1 Õppevideote struktuur

J. Swarts viis läbi uuringu, mis hõlmas 46-t YouTube'is leiduvat erinevat tüüpi videojuhist [10]. Võrdlusesse võeti video, teksti, pildi ning heli töötlemise teemadel leiduvad videojuhised. Videojuhised jaotati vastavalt vaatajate keskmisele hinnangule, kus viiepallisüsteemis jääb „hea" videojuhis vahemikku 3,5-5,0, „keskmine" vahemikku 2,6-3,4 ning „halb" vahemikku 0-2,5. Vastavast analüüsist selgus, millised struktuursed erinevused on headel, keskmistel ning halbadel videojuhistel. Uuringu käigus leiti videote sarnasusi ning erinevusi struktuuris ning retoorilises ehk suulise ja kirjaliku kõne edastamise osas.

Uuringus võrreldi konkreetseid õppevideote elemente, mis olid jagatud kahte plokki:

- video üldine struktuur:
    - sissejuhatus – osa, mis annab ülevaate video sisust, vajalikud hoiatused ning ettevalmistused, et videos läbitehtut ise proovida;
    - sisu ehk sammud - video osa, kus ülesande täitmiseks vajalikke osi selgitatakse ja näidatakse;
    - kokkuvõte - võtab kokku videos tehtu ning kordab üle olulisemad elemendid.
- video sisu ehk sammu elemendid:
    - selgitus – vastavaks sammuks vajalik suuline või kirjalik jutt tegevuse näitamiseta;
    - tegevus – ekraanil sammu näitamine ilma suulise või kirjaliku selgituseta;
    - demonstratsioon - ekraanil sammu näitamine koos suulise või kirjaliku selgitusega (ehk tegevus koos selgitusega).

Kui võiks arvata, et videojuhistel ei ole sissejuhatus niivõrd oluline, siis uuringust selgus, et parematel videotel oli just sarnane struktuur, kus sissejuhatuses räägiti antud video teemast ja eesmärgist ning vajalikest eeldustest, et video sisu ise läbi proovida. Kõikide videote puhul tuli välja, et hinnanguliselt 73% jooksul tervest videost (mõõdetud sekundites)

keskenduti sisule ehk sammudele. Ülejäänud 27% videost oli umbes ⅔ ehk ~18% osas pühendatud sissejuhatusele ning ⅓ ehk ~9% kattis kokkuvõte, kus korrati üle videos olnud põhipunktid. Head ja keskmised videod järgisid üldiselt eelmainitud jaotust. Halbadel videotel aga puudus tihtipeale sissejuhatus ning vaatajate hinnangul jäi puudulikuks vajalike ettevalmistuste osa ning ka video eesmärgi tutvustus.

Videojuhise sisu ehk selgituse, tegevuse ning demonstratsiooni võrdluses sisaldasid videod vähem selgitusi (31% sisust ehk sammudest) ja rohkem demonstratsioone (51% sisust). Tegevuste osakaal oli kõige väiksem (18% sisust). Kasutajate keskmistele hinnangutele tuginedes selgusid erinevad mustrid. Halvad videod sisaldasid sisu elementide osas kõige rohkem tegevusi ning vähim selgitusi. Headel videotel oli rohkem selgitusi, rohkem demonstratsioone ning vähem tegevusi. Demonstreerides räägiti, mida parasjagu tehakse ning miks.

W. Sugar, A. Brown ja K. Luterbach tegid uurimuse 37 videojuhisega [11]. Analüüsiti nii professionaalide koostatud kui ka uuringut teinute endi koostatud videojuhiseid. Videojuhised olid sisult teatud tehniliste protsesside juhendid. Videote analüüsi tulemusena toodi välja videojuhiste puhul sageli kasutatavad elemendid, mida on samuti õppevideoid luues hea silmas pidada. Väljatoodud struktuursed elemendid olid:

- puhverklipid – video (või video osa) alguses ja/või lõpus olev klipp, mis tutvustab videot või võtab selle kokku;
- ekraani liikumine – ekraanil toimuv liikumine staatilise või dünaamilise liikumise vormis. Staatilise liikumise korral liigub kursor ning taust on paigal ja dünaamilise liikumise korral liigub taust ning kursor on ekraani keskel;
- seletused – jaguneb otseseks ja kaudseks seletuseks, kus otsese seletuse puhul öeldakse samm-sammult, mida tegema peab ning kaudse seletuse korral öeldakse üldisemalt, mida tegema peab. Suurem osa analüüsitud videotest sisaldas mõlema seletuse kombinatsiooni.

J. Swartsi uuringu käigus analüüsiti ka S. Carlineri kolmetasandilise informatsioonidisaini elemente [10]. Informatsioonidisaini ning selle elemente tutvustatakse järgmises alapeatükis.

3.2 Informatsioonidisain

Informatsioonidisain ehk teabe kujundamine on tava esitada informatsiooni viisil, mis soodustab selle tõhusat ja selget mõistmist. Informatsioonidisainil on kaks tähendust – üldine hea dokumendi kirjutamise protsess või viis, kuidas informatsioon on edastatud leheküljel või ekraanil [12].

Informatsioonidisain on oluline ka programmeerimise õppevideo puhul, kus üliõpilastele on vaja informatsiooni edasi anda selliselt, et õppevideost oleks neile õppetöös kasu. Informatsioonidisaini elemendid on head pidepunktid, mida jälgida ise õppevideoid luues, et loodud õppevideod ei jääks vaatajale arusaamatuks.

S. Carliner pakkus välja kolmetasandilise lähenemise informatsioonidisainile [13]:

- füüsiline disain – vajaliku informatsiooni leidmine;
- kognitiivne ehk tunnetuslik (intellektuaalne) disain – esitatud informatsioonist arusaamine;
- emotsionaalne disain ehk mõjuvus – arusaadud informatsiooni kasutamise motivatsioon ning informatsiooni korrektne kasutamine.

Hea füüsilise disaini puhul on kasutajale informatsioon lihtsasti leitav. Näiteks õppevideo korral peab olema video loogiliselt struktureeritud nii, et vaatajal oleks seda lihtne vaadata ning vajadusel hõlbus ka konkreetset informatsiooni üles leida. Kognitiivse ehk intellektuaalse taseme puhul on põhiküsimuseks, kas peale vajaliku informatsiooni leidmist on informatsioon arusaadav ning kasulik.

Mõjuvuse taseme eesmärgiks on tagada olukord, kus peale informatsiooni leidmist ning selle mõistmist on vaataja motiveeritud demonstreeritud tegevust ise tegema. Mõjuvuse tase jaguneb kaheks – tähelepanu ja motivatsioon. Kõigepealt tuleb äratada vaatajate tähelepanu ning seejärel motiveerid neid seda informatsiooni kasutama.

Eelnevalt esitatud kolme tasandi rakendumist peaks kindlasti ka õppevideot luues silmas pidama. Vastasel juhul ei ole informatsioon õppevideo vaatajale arusaadav ning see ei ole vaataja jaoks kasulik. Bakalaureusetöö raames loodud videojuhiste tegemisel jälgiti samuti informatsioonidisaini elemente.

4. Tartu Ülikooli programmeerimise õppejõudude suhtumine õppevideotesse ning kokkupuude õppevideotega

Tartu Ülikooli *Programmeerimise*, *Programmeerimise alused*, *Programmeerimise alused II*, *Programmeerimise harjutused* ja *Objektorienteeritud programmeerimise* õppejõudude seas viidi läbi anonüümne veebipõhine küsitlus (Lisa I). Küsitletute hulgas olid ka üliõpilastest praktikumijuhendajad. Küsitluse eesmärk oli välja selgitada õppejõudude kokkupuude õppevideote loomisega ja rakendamisega ning suhtumine õppevideote kasutamisse. Küsitlus koosnes 21-st küsimusest, kuid kõikidele küsimustele vastaja vastama ei pidanud. Mõningad küsimused sõltusid eelnevatest vastustest.

Küsitlusele vastas 35-st õppejõust 17, kellest 12 on meessoost ja 5 on naissoost. Suurem osa vastajatest (12) on kuni 30-aastased. 6 õppejõudu on õpetamiseks õppevideoid kasutanud ning 6 õppejõudu on ise õppevideoid loonud. Väheste vastajate arvu tõttu ei ole saadud tulemused piisavad Tartu Ülikooli arvutiteaduse instituudi programmeerimise õppejõudude arvamusest tervikpildi saamiseks.

Õppevideoid rakendanud õppejõudude puhul kasutatakse kõige rohkem loengu videot (5), videojuhiseid on kasutanud 4 ning videonäiteid ja videoloenguid 3 õppejõudu. Selgus, et üldiselt kasutatakse kas enda või Tartu Ülikooli kolleegide poolt koostatud õppevideoid (vastavalt 5 ja 4) ning mõnel juhul on kasutatud ka teistes ülikoolides või mujal koostatud õppevideoid. Õppevideoid kasutatakse enamjaolt kas enne või pärast auditoorset tööd, vähematel juhtudel auditoorse töö jooksul. Õppevideoid mittekasutanud õppejõudude puhul on järgnevalt välja toodud mõned vastused küsimusele „Miks Te ei kasuta kursustel õpetamiseks õppevideoid?" (kirjapilt muutmata):

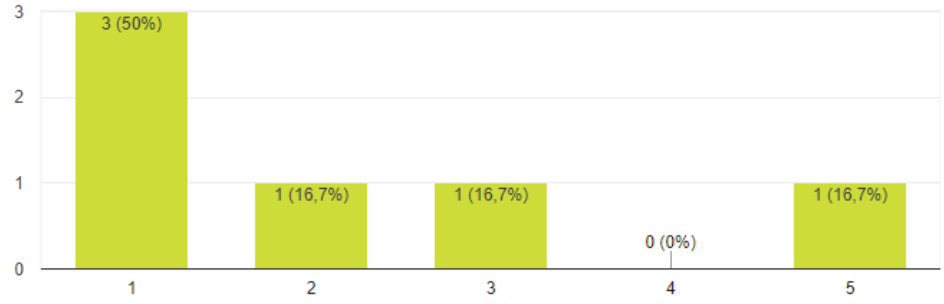
„Videoformaadi tarbimine tudengi jaoks pole nii mugav kui teksti lugemine. Kuna tase varieerub, siis paljude jaoks võib video olla aeglane ja igav, samas teistele jälle liiga kiire, kui on vaja rohkem asju läbi mõelda. Teksti lugemisel on tempo kergesti valitav. Lisaks on videos ebamugav navigeerida, kui hiljem on soov mingi koht jälle üles leida; sisu otsida ka ei saa."

„Ei näinud vajadust, püüdsin kõik vajaliku ise seletada ning tundus, et tudengitele see sobis."

„Aines Programmeerimine olid ette nähtud kindlad ülesanded, mida tunnis lahendasime. Muuks ei jätkunud kahjuks eriti aega. Usun, et algoritmide kasutamisel võivad õppevideod

olla efektiivsed (rahvatantsurühm, kes tantsides järjestab ennast kindla sorteerimis-algoritmiga).“

Kuuest õppevideoid ise koostanud õppejõust on kaks õppejõudu koostanud 1 kuni 5, üks õppejõud koostanud 6 kuni 20 ning kolm õppejõudu koostanud üle 20 õppevideo. Kõige rohkem on ise tehtud loengu videoid ning videojuhiseid (mõlemat neljal juhul). Kasutatud tarkvaradena toodi välja Camtasia, Panopto, Moodle, MS PowerPoint, Echo ning Audacity. Õppevideote koostamiseks on kasutatud kolleegide abi, ATI.comp tehnilist toetust ning juhendit ja haridustehnoloogi. Õppevideo loomiseks käsikirja kirjutamine jagunes järgnevalt (1 tähistas hinnangut „mitte kunagi“ ning 5 tähistas hinnangut „alati“):



Joonis 1. Hinnangute jaotus väitele „Kui tihti olete videotele teinud käsikirja?“.

Peamisteks põhjusteks õppevideote mittetegemiseks on teiste hulgas (kirjapilt muutmata):

„Tundub ajamahukas, palju vajalikke materjale on juba inglise keeles olemas nii et kasutan neid.“

„Pole tundnud suurt vajadust ja videote koostamine on töömahukas. Tunnen end kirjaliku teksti koostamisel mugavamalt.“

„Programmeerimise õpetamisega ei ole teinud õppevideosid. Olen kasutanud materjalides teiste loodud videosid, mis on kvaliteetsed. Samuti on videote tegemine ajamahukas ja mul puuduvad vajalikud oskused.“

Lisaks uuriti, milline abi oleks kasulik õppevideote koostamisel. Järgnevalt on välja toodud mõned vastused küsimustele “Milline abi oleks Teile videote koostamisel kasuks?” (kirjapilt muutmata):

„Kui keegi suudaks enesekindlust tõsta, siis vast saaksin hakkama.“

„Kui keegi teeks minu poolt paika pandud teksti ja juhiste järgi video valmis.“

„Kui õppevideoid luua, siis peaks seda tehtama nii, et asi näeb piisavalt professionaalne välja (a la Khanacademy), kasutades ära sobivad tehnoloogiad ja tarkvara. Seda iga õppejõud ise uurida ja omandada kindlasti ei jõuaks. Niisama ekraani ja veebikaamerat salvestada ja midagi rääkida pole mõtet.“

Lõpetuseks olid küsitluses väited, millega vastajad said viiepalliskaalal oma nõusolekut hinnata. Järgnes valikvastustega küsimus programmeerimise õppevideo kasutamise võimalikest eesmärkidest ning võimalus lisada ka kommentaare õppevideote teemal.

Kõikide väidete hinnangud olid samasugusel skaalal, kus 1 tähistas hinnangut „ei nõustu üldse“ ning 5 tähistas hinnangut „nõustun täielikult“. Järgnevalt on tabelina välja toodud väited ning neile antud hinnangud:

Tabel 1. Hinnangute jaotus õppejõududele esitatud väidetele õppevideote teemal.

	1	2	3	4	5
Programmeerimise õppevideol peab olema sissejuhatus	0	2	3	5	7
Programmeerimise õppevideol peab olema kokkuvõte	1	2	3	4	7
Programmeerimise õppevideo peab sisaldama demonstratsioone ehk tegevuse ja selgituse kooslust	0	0	0	5	12
Õppevideo koostamine on liiga aeganõudev	1	1	4	5	6

Antud tabelist selgub, et õppejõudude arvates on programmeerimise õppevideos sissejuhatus, kokkuvõte ja demonstratsioonid pigem olulised. Eelnevast kolmest variandist on vähemolulisem element kokkuvõte. Õppejõudude arvates on õppevideote loomine liiga aeganõudev.

Õppevideote rakendamise eesmärkidena märgiti kõige enam lisamaterjalidena kasutamist ja loengus õpitu kordamist (vastavalt 16 ja 15). 13 õppejõudu arvavad, et õppevideoid võiks kasutada kontrolltööks või eksamiks õppimiseks ning 12 õppejõudu arvavad, et õppevideoid võiks kasutada loenguks või praktikumiks ettevalmistumisel ja kodutöö tegemisel. Üks õppejõud on lisanud vastuseks ka *„ei usu video kasulikkusesse“*.

Lõpetuseks said vastajad jätta ka kommentaare õppevideote teemal. Järgnevalt on välja toodud mõned kommentaarid (kirjapilt muutmata):

„Kui keegi teeks kursuse, kus õpitakse häid õppevideosid tegema, siis oleksin käpp :).“

„Tekstilise materjaliga on peaaegu alati mugavam töötada, saab otsida, copy-pasteda, omas tempos läbida (eriti tähtis). Tekstilist materjali saab kergemini siduda interaktiivse formaadiga – ülesannete lisamine teksti sisse, praksis materjali läbi töötamisega paralleelselt õppejõuga suhtlemine. Video puhul on see kõik raskem, kui mitte võimatu, sest video läheb omas tempos edasi (eriti kui praksis ühiselt vaadata).“

„Õppevideod on ennekõike sobivad iseseisvaks tööks, mitte loengutes ja praktikumides näitamiseks.“

„Ajakulu sõltub sellest, milliseid õppevideosid teha. Kui loengusalvestusi (nagu mina olen seni teinud), siis seal ajakulu on minimaalne. Kui aga iseseisvaks õppimiseks mõeldud videoloenguna, siis on ajakulu üsna suur (aga sellele vaatamata ei ole see liiga suur, kuna ka kasu on suur).“

Läbiviidud küsitlusest selgus, et programmeerimise õppejõud kardavad, et õppevideote loomine on väga aja- ja töömahukas ning samuti arvatakse, et puuduvad vajalikud oskused. Käesoleva bakalaureusetöö raames koostati järgnevalt kaks õppevideot, et selgitada välja õppevideote loomise keerukus ning ajakulu.

5. Objektorienteeritud programmeerimise kursuse jaoks loodud õppevideod

Antud bakalaureusetöö raames loodud õppevideote eesmärk oli asendada 30. aprillil 2018. aastal kevadpüha tõttu ära jäänud *Objektorienteeritud programmeerimise* Java andmestruktuuride ning geneeriliste meetodite, klasside ja liideste teemalist loengut. Lisaks pidid üliõpilased tegema vaadatud õppevideote põhjal ka Moodle'i testi. Ärajäänud loengule eelnenud loengus tutvustati väga üldiselt Javas olemasolevaid andmestruktuure ning geneerilisi meetodeid, klasse ja liideseid. Lisaks jagati loodud õppevideoid ka avatud ülikooli raames *Objektorienteeritud programmeerimise* kursusel osalejatega, kellel oli võimalus videoid vaadata ning küsimustik täita.

Töö autor ei ole varasemalt ise ühtegi videot loonud ning ei ole kokku puutunud ka video loomiseks kasutatava tarkvaraga. Seega võib protsessi ning tulemusi tõlgendada kui õppevideote loomist algaja tasemel. Õppevideote filmimiseks ning töötlemiseks kasutati Camtasia Studio 9 tarkvara, mida võimaldatakse kasutada ka Tartu Ülikooli arvutiteaduste instituudi õppejõududel.

5.1 Ülevaade „Objektorienteeritud programmeerimise" kursusest

Käesolev peatükk tutvustab kursust *Objektorienteeritud programmeerimine*, selgitades kursuse ülesehitust ning käsitletavaid teemasid. Ülevaade on vajalik kursuse taustainfo teadmiseks, mille raames õppevideod on koostatud. Antud peatükis väljatoodud informatsioon on pärit Tartu Ülikooli õppeinfosüsteemist [14].

Õppeinfosüsteemist selgub, et „kursuse eesmärgiks on anda alusteadmised objektorienteeritud programmeerimise eripärast, oskused programmide koostamiseks ning esmased rühmatööoskused". Antud kursus on ennekõike suunatud esimese aasta informaatika bakalaureuse ning infotehnoloogia mitteinformaatikute magistriõppe õppekava üliõpilastele. Kursus eeldab eelnevat programmeerimise kogemust ehk peab olema läbitud mõni järgnevalt nimetatud eeldusainetest – *Programmeerimine* (MTAT.03.100, LTAT.03.001), *Riistvaralahenduste visuaalprogrammeerimine* (LOFY.03.004), *Andmehõive ja analüüs LabVIEW keskkonnas* (LOFY.03.017), *Programmeerimise alused II* (MTAT.03.256).

Objektorienteeritud programmeerimise kursuse maht on 6 EAP-d, kus 1 EAP märgib ~26 tundi õppetööd.

Objektorienteeritud programmeerimise kursus koosneb iganädalastest loengutest ning praktikumidest. Iganädalased loengud salvestatakse üldiselt loengu videona ning üliõpilastel on võimalik videoid hiljem uuesti vaadata. Samuti kasutatakse loengutes klikkereid, et juba loengu jooksul üliõpilasi kaasata ning saada ülevaade üliõpilaste teema mõistmisest. Klikker on elektrooniline vahend, mida kasutades viiakse läbi loengus valikvastustega küsitlusi ning tulemused kuvatakse koheselt pärast küsitluse sulgemist [15]. Praktikumid on korraldatud tekstimaterjalidena ning praktikumis olles harjutatakse koos praktikumijuhendajaga tekstina antud ülesandeid.

Kursusel käsitletakse mitmeid teemasid, alustades Java programmi loomisest, kompileerimisest ning käivitamisest, lõpetades võrguprogrammeerimisega. Teiste hulgas on käsitletavateks teemadeks Java andmestruktuurid ning geneerilised meetodid, klassid ja liidesed ning neid teemasid hõlmavad ka bakalaureusetöö raames koostatud õppevideod.

5.2 Videote liik ning struktuur

Bakalaureusetöö raames koostatud kaks õppevideot on videonäited seetõttu, et eelnenud loengus tutvustati juba üldiselt Javas olemasolevaid andmestruktuure ning geneerilisi meetodeid, klasse ja liideseid. Otsustati, et Java andmestruktuuride videonäites demonstreeritakse andmestruktuuride sarnasusi ja erinevusi ning geneeriliste asjade videos demonstreeritakse geneeriliste asjade loomist ja kasutamist. Videonäited tehti ekraanisalvestuse vormis, sest harilikult neid sellises vormis tehakse ning töö autor ei näinud selle muutmiseks vajadust.

Struktureerimiseks kasutati J. Swartsi uuringust tulenenud videojuhiste häid komponente, mida rakendati antud bakalaureusetöös videonäidete loomisel. J. Swartsi uuringust tulenevalt jagati videonäide kolme ossa: sissejuhatus, sisu ning kokkuvõte. Sisus kasutati peamiselt demonstratsioone ning mõningates olukordades ka seletusi, et demonstratsioonile eelnevalt vajalik informatsioon edasi anda või peale demonstratsiooni midagi lisaks rääkida.

Lisaks jälgiti videonäiteid koostades S. Carlineri kolmetasandilist informatsioonidisaini raamistikku. Füüsilise disaini osas otsustas töö autor lisada erinevate osade vahele W. Sugari, A. Browni ja K. Luterbachi uuringus välja toodud puhverklipid, et videonäidet sirvides oleks vaatajal lihtne ja mugav vajalikku osa üles leida. Samuti lisati YouTube'i üleslaaditud videonäidete kirjeldusse ajahetked, millal mingi teema videos algab. Kognitiivse disaini osas kasutati videos võimalikult lihtsaid ning loogilisi näiteid, et kursusel osalevale üliõpilasele oleksid näited arusaadavad ning räägiti pigem rahulikult, et

kellegi jaoks tempo liiga kiire ei oleks. Mõjuvuse osas anti üliõpilastele võimalus ka ise sellist koodi kirjutada ja/või jooksutada. Selle jaoks saadeti koos videonäidetega kaasa ka videotes demonstreeritud programmilõigud.

5.3 Ettevalmistus

Videonäidete ettevalmistus koosnes kolmest etapist. Etappideks olid materjaliga tutvumine, videonäite sisu elementide valimine ning osadeks jagamine, videonäite sisu osade käsikirja ning koodilõikude kirjutamine.

Materjalidega tutvumise etapp tulenes ennekõike sellest, et töö autor ei ole ise *Objektorienteeritud programmeerimise* kursuse õppejõud. Selleks, et kasutada korrektset terminoloogiat ning näidata ainele kohaseid materjale, pidi töö autor tutvuma kursuse loengu videotega, loenguslaididega ning praktikumimaterjalidega, mille leiab Tartu Ülikooli arvutiteaduste instituudi kursuste veebilehelt [16].

Videonäite sisu elementide valimine ning osadeks jagamine ei olnud eriti aegavõttev tegevus, sest loenguslaididel oli selgelt välja toodud kursuse raames käsitletavad Javas olemasolevad andmestruktuurid ning geneeriliste meetodite, klasside ja liideste osadeks jagamine on üsnagi iseenesest mõistetav. Töö autor kirjutas mõlema videonäite jaoks eraldi käsikirja räägitava teksti ning koodinäidetega (Lisa III, Lisa IV).

Kogu ettevalmistusele kulus Java andmestruktuuride videonäite puhul ~7 tundi ning geneeriliste asjade videonäite kohta veidi vähem - ~6 tundi. Geneeriliste asjade videonäite ettevalmistusaeg on veidi lühem seetõttu, et materjali on veidi vähem kui Java andmestruktuuride puhul. Tegelik käsikirja kirjutamine ning koodinäidete katsetamine võttis mõlema videonäite puhul 2-3 tundi.

5.4 Filmimine

Filmimiseks kasutati Camtasia Studio 9 tarkvara. Filmimisel oli väga kasulik varasemalt valmiskirjutatud käsikiri, mis võimaldas suhteliselt kiiresti ekraanisalvestised ning helifailid üles võtta. Käsikirja puudumisel oleks tõenäoliselt filmimisele rohkem aega kulunud. Filmimine jagati kahte ossa ning esimese osana tehti valmis ekraanisalvestised ehk koodi kirjutamist hõlmavad näited, kus iga teema filmiti eraldi failina. Teise osana salvestati helifailid, millele hiljem lisati vastav teemakohane pilt.

Videonäidete filmimine võttis ~2 tundi. Java andmestruktuuride videonäite monteeritavat materjali (ekraanisalvestised ja helifailid) oli lõpuks kokku ~36 minutit, millest ~33 minutit

oli ekraanisalvestisi ning ~3 minutit helifaile. Geneeriliste asjade videonäite monteeritavat materjali oli kokku ~19 minutit, millest ~17 minutit oli ekraanisalvestisi ning ~2 minutit helifaile.

Ekraanisalvestiste ning helifailide aja hulka ei ole arvestatud kustutatud faile, mida esines rohkem helifailide osas. Helifailide salvestamisel valesti läinud helifail salvestati uuesti ning valesti läinud osa ei säilitatud. Ekraanisalvestuse korral valesti läinud hetkel tehti pikem paus (vähemalt 10 sekundit, et monteerimisel oleks lihtsam uuesti alustamise kohta leida) ning alustati filmimist peatamata uuesti eelmisest lausest või osast.

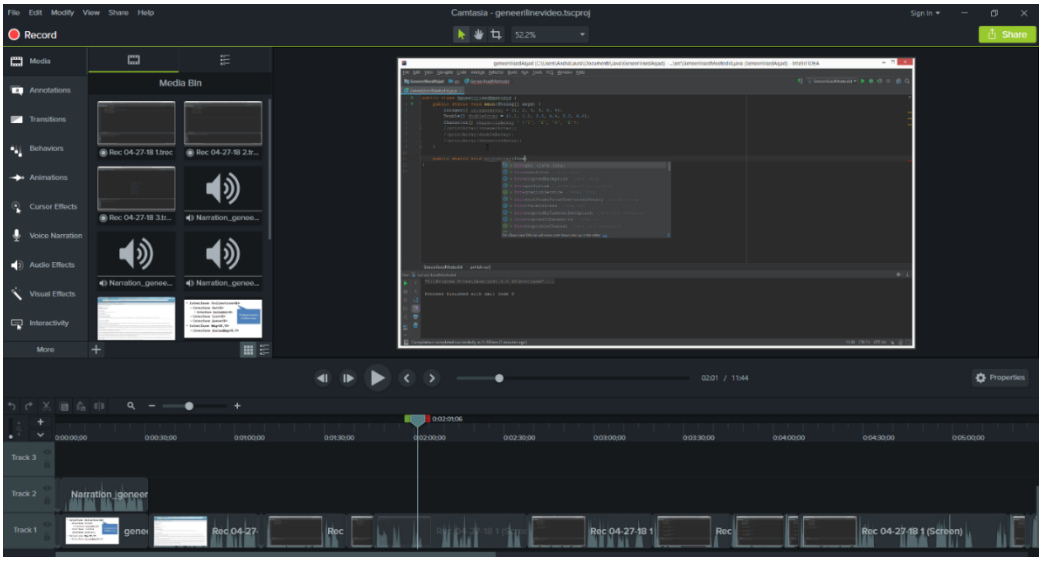
5.5 Monteerimine

Õppevideo monteerimine on õppevideo loomise aeganõudev osa. Videonäited monteeriti samuti Camtasia Studio 9 tarkvara kasutades. Monteerimisele kulus mõlema video puhul 3-4 tundi. Töötlemisel loodi esmalt kahesekundiline puhverklipp, millel on kirjas video pealkiri ning video autor:

Joonis 2. Näide videonäite alguses olevast puhverklipist.

Seejärel pandi paika sissejuhatus ja muud koodinäidetele eelnevad helifailid ning neile vastavad teemakohased pildid. Kolmanda sammuna hakati teemade kaupa monteerima ekraanisalvestisi. Ekraanisalvestist monteerides lõigati välja valesti läinud osad ning ka liiga pikaks veninud pausid. Seejärel pandi paika koodinäidetele järgnevad helifailid (k.a kokkuvõte) ning neile vastavad teemakohased pildid. Videonäite jooksul kasutati kahesekundilisi

puhverklippe iga teemavahetuse juures, et video vaatajal oleks videonäidet sirvides mugavam vajalikku teemat leida, ja et video oleks paremini struktureeritud.



Joonis 3. Näide video monteerimisest Camtasia Studio 9 tarkvaraga.

5.6 Tulemus

Ettevalmistuse, filmimise ning monteerimise tulemusena valmis kaks videonäidet. Java andmestruktuuride videonäite pikkuseks on 22 minutit ja 41 sekundit ning geneeriliste meetodite, klasside ja liideste videonäite pikkuseks on 11 minutit ja 44 sekundit. Videonäited laaditi üles YouTube'i veebilehele. Java andmestruktuuride videonäide on kättesaadav veebiaadressilt: https://www.youtube.com/watch?v=iHUYaWxed7g ning geneeriliste meetodite, klasside ja liideste videonäide on kättesaadav veebiaadressilt: https://www.youtube.com/watch?v=oNbKGSkvkZQ.

Java andmestruktuuride video tegemisele kulus ~12 tundi ning geneeriliste meetodite, klasside ja liideste videole ~9 tundi. Video ettevalmistuse ajakulu võib kursuse õppejõud paar tundi lühemaks arvestada, sest ollakse teemaga juba eelnevalt kursis ning tõenäoliselt on üsna selgelt teada, mida videos on vaja näidata (nii jääks ära palju aega nõudnud materjaliga tutvumise etapp). Töö autor tõdes, et pärast esimese videonäite tegemist oli teist videonäidet juba oluliselt lihtsam teha ning aega kulus vähem. Õppevideo loomise kogemust arendades väheneb ilmselt mõnevõrra ka ettevalmistusele ning monteerimisele kuluv aeg. Filmimise osas võib kuluv aeg väheneda pigem seetõttu, et filmimisel tehakse vähem vigu, sest niisuguse tegevusega ollakse juba harjunud. Kokkuvõttes on siiski õppevideo loomine pigem ajakulukas, kuid ajakulu väärt, kui selle tulemusena valmib kasulik õppevideo.

6. Üliõpilaste tagasiside loodud videonäidetele ning suhtumine programmeerimise õppevideotesse

Tartu Ülikooli *Objektorienteeritud programmeerimise* kursuse üliõpilaste seas viidi läbi anonüümne veebipõhine küsitlus (Lisa II). Küsitluse eesmärk oli välja selgitada üliõpilaste tagasiside loodud õppevideotele ning suhtumine programmeerimise õppevideote kasutamisse. Küsitlus koosnes 17-st küsimusest, millest neljale tuli vastata vaid siis, kui vastav õppevideo oli vaadatud. Küsitlusele vastas 84 üliõpilast, kellest 68 oli vaadanud Java andmestruktuuride ning 65 oli vaadanud geneeriliste meetodite, liideste ja klasside videonäidet.

Java andmestruktuuride ning geneeriliste meetodite, klasside ja liideste videonäidete tagasiside koosnes mõlema õppevideo puhul kolmest väitest, millega nõusolekut said vastajad hinnata. Samuti oli mõlema videonäite tagasiside osas võimalik lisada kommentaare antud videonäidete kohta.

Kõikide väidete hinnangud olid samasugusel skaalal, kus 1 tähistas hinnangut „ei nõustu üldse“ ning 5 tähistas hinnangut „nõustun täielikult“. Järgnevalt on tabelina välja toodud väited ning nendele antud hinnangud:

Tabel 2. Hinnangute jaotus üliõpilastele esitatud väidetele loodud õppevideote teemal.

	1	2	3	4	5
Java andmestruktuuride õppevideo sissejuhatusest tuli selgelt välja video eesmärk	0	0	6	24	38
Java andmestruktuuride õppevideo sisu oli arusaadav	0	1	3	21	43
Java andmestruktuuride õppevideos tulid välja liideste erinevused	0	1	9	18	40
Geneeriliste asjade õppevideo sissejuhatusest tuli selgelt välja video eesmärk	0	0	8	19	38
Geneeriliste asjade õppevideo sisu oli arusaadav	0	0	5	23	37
Geneeriliste asjade õppevideost tuli välja geneeriliste asjade ja kindla tüübiga asjade erinevus	0	1	9	19	36

Lisaks on välja toodud mõned kommentaarid loodud õppevideotele (kirjapilt muutmata):

„Väga head videod, hästi lihtsalt seletatud, head näited.“

„Tempo võiks veidi kiirem olla, sest videos saab alati ise pausile panna. Muidu väga tore.“

„Jutt oleks võinud natuke ilmekam olla. Praegu jäi tunne, et see on lihtsalt maha loetud, emotsiooni ei olnud üldse. Kaasahaaravat ja ilmekat juttu on palju lihtsam kuulata.“

„Väga hea, sain rohkem aru kui slaide vaadates.“

„Kui ma ei oleks enne läbi vaadanud slaide ja teinud kodutööd, siis ma ilmselt ei oleks saanud kõigest aru.“

„Võibolla oleks võinud olla näiteid, millistes olukordades ühe või teise andmestruktuuri kasutamist eelistada.“

„Nii praeguse kui eelmise puhul võiks rakendada video puhul esitluse varianti IJs, on suurem tekst.“

Antud tagasisidest selgus, et õppevideotega oldi pigem rahul. Toodi välja ka elemente, mida oleks hea edaspidi parandada, näiteks suurem tekst koodi näitamisel ning ilmekam jutt. Esines ka kommentaare, et video oleks võinud olla kiirem, kuid töö autor usub, et videot on võimalik kiirendusega käima panna ning seega on parem pigem aeglasem kui kiirem video.

Küsitluse viimase osa eesmärgiks oli tuvastada üliõpilaste suhtumist programmeerimise õppevideotesse. Antud plokk koosnes väidetest, millega vastajad said taaskord viiepalliskaalal oma nõusolekut hinnata. Lisaks oli valikvastustega küsimus programmeerimise õppevideote kasutamise eesmärkide kohta ning vaba vastusega kommentaarid õppevideote kohta programmeerimise kursusel.

Kõikide väidete hinnangud olid samasugusel skaalal, kus 1 tähistas hinnangut „ei nõustu üldse“ ning 5 tähistas hinnangut „nõustun täielikult“. Järgnevalt on tabelina välja toodud väited ning nendele antud hinnangud:

Tabel 3. Hinnangute jaotus üliõpilastele esitatud väidetele õppevideote teemal.

	1	2	3	4	5
Programmeerimise õppevideol peab olema sissejuhatus	0	5	13	27	39
Programmeerimise õppevideol peab olema kokkuvõte	2	4	6	24	48
Programmeerimise õppevideo peab sisaldama demonstratsioone ehk tegevuse ja selgituse kooslust	0	0	3	16	65
Õppevideod lihtsustavad programmeerimise õppimist	0	3	14	28	39
Oleksin valmis ise õppevideoid koostama	27	28	18	8	3

Saadud vastustest selgub, et sissejuhatus ja kokkuvõte on pigem õppevideo olulised osad, kuid kokkuvõte on üliõpilaste jaoks vähemolulisem kui sissejuhatus. Samuti ei piisa üliõpilastele vaid selgitusest või ainult selgituseta tegevusest. Üldiselt arvatakse, et õppevideod lihtsustavad programmeerimise õppimist. Küsitlusse lisati väide „Oleksin valmis ise õppevideoid koostama“ eesmärgiga selgitada välja, kas on võimalik, et üliõpilased kursuse raames ka ise õppevideoid teeksid. Üliõpilased pigem ei ole ise valmis õppevideoid koostama, mis tundub üllatav, arvestades tänapäeval levinud videote tegemise ja jagamise praktikat sotsiaalmeedias (näiteks YouTube’is).

Programmeerimise õppevideote rakendamise eesmärkidena valitud vastused kahanevas järjekorras olid - lisamaterjal (65), loengus õpitu kordamine (59) ja kontrolltööks või eksamiks õppimine (59) ning loenguks või praktikumiks valmistumine (48). Lisaks oli variantideks pakutud loengus käimise asendamist (2) ning mingite asjade toimumise visualiseerimist (1).

Järgnevalt on välja toodud mõned kommentaarid õppevideote kohta programmeerimise kursustel (kirjapilt muutmata):

„Õppevideod on väga kasulikud. Kuigi neid leidub palju inglise keeles, on eesti keeles neid kohati parem vaadata. Eriti hea, kui videod on laetud keskkonda nagu nt YouTube, kus neile on kõigil ligipääs. Video vaatamine, kus tehakse läbi näited koos seletustega annab kindlasti parema ja kiirema tõuke uuest teemast arusaamisele, kui nt ise internetist dokumentatsiooni otsimine ja lugemine. Aitäh!“

„Vahel on see, et video kestab teatud aja aga kirjalikust materjalist käid silmadega üle, otsid seda, mida ei tea ja hoiad aega kokku. Jällegi, kui kirjalikust osast aru ei saa, siis on hea õppevideost teatud kohad järgi vaadata.“

„Päris lahe oleks kui oleks lühikesed videod iga kirjaliku osa kohta, seega saab neid vastavalt vajadusele vaadata ja saab aega kokku hoida sellega, et ei pea otsima videost kindlat kohta, kindlasti oleks sellel veel mingeid eeliseid. Kõike head :D!“

„Õppevideod aitavad siis, kui sul pole teemast veel mingit ettekujutust, aga kui oled teooria juba omandanud, siis minu arvates näiteks kodutööde tegemisel sealt väga palju abi ei saa, sest õppevideod keskenduvad enamasti üldistele teemadele, mitte konkreetsetele juhtumitele. Ülesanded on aga tihtipeale üsna spetsiifilised. Miski ei aita rohkem kui ise programmeerimine.“

„Pigem eelistan kirjalikku materjali, kui videomaterjali, sest saab mugavalt mõnda asja üle lugeda ja mõnda lugemata jätta. Video puhul on see palju tüütum.“

„Õppevideod + iseseisev töö on kindlasti parem kui mistahes ülikoolile omasem õppevorm (loeng + praktikum, aga kuna ülikoolile nii hirmsasti meeldib peale suruda igasugu tobedusi, siis kahjuks ei jää eriti aega muude õppematerjalidega tegelemiseks.“

Ärajäänud *Objektorienteeritud programmeerimise* loengule järgnevas loengus esitati tudengitele klikkeritega küsimus „Millist õppimise viisi eelistaksite?“. Vastustest selgus, et tudengid eelistavad kindlalt (36%) ja pigem (34%) loengut. 12%-l tudengitest ei ole vahet ning 15% eelistavad pigem videot ja Moodle’i testi ning 3% eelistavad kindlalt videot ja Moodle’i testi [16]. Küsimuse põhjal ei saa teha järeldust, kas üliõpilased eelistavad loengut videole, sest vastus võib olla tingitud ka Moodle’i testi soovimatusest.

Küsitlusest selgus, et loodud õppevideotega jäädi pigem rahule, välja olid toodud mõningad vead, mida on edaspidi kindlasti vajalik silmas pidada. Õppevideotesse suhtumise teemal oli vastakaid arvamusi, kus mõned vastajad arvasid, et õppevideod on kasulikud ning oli ka vastajaid, kes arvasid, et kirjalik materjal on parem kui õppevideod. Vastakad arvamused selgusid lisatud kommentaaridest. Üldiselt arvatakse siiski, et õppevideod lihtsustavad programmeerimise õppimist, kuid üliõpilased ise õppevideoid pigem valmis tegema ei oleks. Õppejõudude ja üliõpilaste arvamus programmeerimise õppevideote struktuuri ning kasutamise eesmärkide kohta oli väga sarnane.

7. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli anda ülevaade õppevideote rakendamisest programmeerimise kursustel.

Töö esimeses osas uuriti kirjanduse põhjal, millised on õppevideote liigid ning millistel eesmärkidel kasutatakse õppevideoid programmeerimise kursustel. Samuti toodi välja erinevused õppevideote ja traditsiooniliste programmeerimise õpetamise ja õppimise vahendite ning materjalide vahel. Lisaks tutvustati õppevideote struktuuri ning informatsioonidisaini elemente, mida saab kasutada pidepunktidena õppevideo loomisel.

Antud töö raames paluti programmeerimise õppejõududel vastata küsitlusele, mille abil püüti välja selgitada õppejõudude kokkupuude õppevideote kasutamise ja loomisega ning nende suhtumine õppevideote kasutamisse. Õppejõudude vastustest selgus, et ühtset suhtumist õppevideotesse programmeerimise kursustel ei ole – esines nii õppevideoid eelistavaid, neutraalseid kui ka õppevideote vastu olevaid õppejõude.

Bakalaureusetöö autor koostas kaks videonäidet *Objektorienteeritud programmeerimise* kursuse tarbeks. Loodud videonäited on Java andmestruktuuride ning geneeriliste meetodite, klasside ja liideste teemal. *Objektorienteeritud programmeerimise* kursuse üliõpilastel paluti vastata küsitlusele, et välja selgitada tagasiside loodud videonäidetele ning suhtumine õppevideote rakendamisse programmeerimise kursustel. Üldiselt oli tagasiside pigem positiivne ning üliõpilased suhtuvad ka õppevideote kasutamisse pigem positiivselt, kuid leidus ka neid, kelle jaoks ei ole õppevideod materjali omandamisel esmane valik.

Käesoleva töö materjale saab kasutada programmeerimise õppevideo loomise protsessist ülevaate saamiseks ning samuti võib rakendada antud bakalaureusetööd alusmaterjalina uurimaks põhjalikumalt, kas õppevideod võiksid asendada traditsioonilisi õpetamise vahendeid ja materjale ülikooli kursustel. Lisaks usub töö autor, et käesolevat bakalaureuse-tööd saab edukalt kombineerida M. Gaiduki „Videojuhised programmeerimise aluste kursustel“ magistritööga, et aidata luua programmeerimise kursuste tarbeks kvaliteetsed ja kasulikud õppevideod, mis lihtsustavad üliõpilastel materjali omandamist ja tõhustavad õppejõududel selle edasiandmist.

8. Viidatud kirjandus

[1] „MOOCid,“ [Võrgumaterjal]. Available: https://www.ut.ee/et/oppimine/moocid. [Kasutatud 13. mai 2018].

[2] M. Gaiduk, „Videojuhised programmeerimise aluste kursustel,“ Tartu, Eesti, 2013.

[3] J. Bennedsen ja M. E. Caspersen, „Revealing the Programming Process - Using Videos to Unfold Basic Programming Techniques,“ 2005.

[4] I. Vieira, A. P. Lopes ja S. Filomena, „The Potential Benefits Of Using Videos In Higher Education,“ *EDULEARN14*, Barcelona, Hispaania, 2014.

[5] P. Joa ja T. Mee, „Video õppetöös,“ [Võrgumaterjal]. Available: http://www.e-ope.ee/_download/repository/Priit_Joa.pdf. [Kasutatud 13. mai 2018].

[6] H. D. Brecht ja M. O. Suzanne, „Enabling a Comprehensive Teaching Strategy: Video Lectures,“ *Journal of Information Technology Education, vol. 7*, pp. 71-86, 2008.

[7] M. Ronchetti, „Using Video Lectures to Make Teaching More Interactive,“ *iJET, vol. 5, no. 2*, pp. 45-48, 2. juuni 2010.

[8] L. Pappano, „The Year of the MOOC,“ *The New York Times*, 2. november 2012.

[9] J. H. Sharp ja L. A. Schultz, „An Exploratory Study of the use of Video as an Instructional Tool in an Introductory C# Programming Course,“ *Information Systems Education Journal, vol. 11, no. 6*, pp. 33-39, Detsember 2013.

[10] J. Swarts, „New Modes of Help: Best Practices for Instructional Video,“ *Technical Communication, vol. 59, no. 3*, pp. 195-206, August 2012.

[11] W. Sugar, A. Brown ja K. Luterbach, „Examining the Anatomy of a Screencast: Uncovering Common Elements and Instructional Strategies,“ *IRRODL, vol. 11, no. 3*, pp. 1-20, 2010.

[12] J. C. Redish, „What Is Information Design?,“ *Technical Communication, Second Quarter*, pp. 163-166, 2000.

[13] S. Carliner, „Physical, Cognitive, and Affective: A Three-part Framework for Information Design,“ *Technical Communication, Fourth Quarter*, pp. 561-576, 2000.

[14] „Tartu Ülikooli õppeinfosüsteem,“ [Võrgumaterjal]. Available: https://ois.ut.ee. [Kasutatud 13. mai 2018].

[15] „Klikker,“ [Võrgumaterjal]. Available: https://www.cs.ut.ee/et/oppimine/klikkerid. [Kasutatud 13. mai 2018].

[16] „Tartu Ülikooli arvutiteaduse instituudi veebileht,“ [Võrgumaterjal]. Available: https://courses.cs.ut.ee/. [Kasutatud 13. mai 2018].

Lisad

I. Küsitlus programmeerimise õppejõududele

Tere, mina olen Andra Laura Meeksa ja teen bakalaureusetööd programmeerimise kursustel õppevideote kasutamise kohta. Antud küsitluse eesmärgiks on teada saada programmeerimise õppejõudude suhtumine õppevideote kasutamisse ning õppejõudude kogemus õppevideote koostamisel.

1. Sugu
2. Vanus
3. Kas olete oma tundides õppematerjalidena õppevideoid kasutanud?

Küsimused õppevideoid kasutanud õppejõududele:

4. Milliseid õppevideoid olete õpetamiseks kasutanud?
5. Kelle koostatud õppevideoid olete programmeerimise kursustel kasutanud?
6. Millal on üliõpilased pidanud õppevideoid vaatama?

Küsimus õppejõududele, kes ei ole õppevideoid kasutanud:

7. Miks Te ei kasuta kursustel õpetamiseks õppevideoid?

Küsimus kõikidele vastajatele:

8. Mitu õppevideot olete ise koostanud?

Küsimused õppevideoid koostanud õppejõududele:

9. Milliseid õppevideoid olete koostanud?
10. Milliseid programme olete õppevideo tegemiseks kasutanud?
11. Kui tihti olete videotele teinud käsikirja?
12. Millist abi olete videote koostamisel kasutanud?
13. Milline abi oleks Teile videote koostamisel kasuks?

Küsimused õppejõududele, kes ei ole õppevideoid koostanud:

14. Miks Te ei ole õppevideoid teinud?
15. Milline abi oleks Teile videote koostamisel kasuks?

Küsimused kõikidele vastajatele:

16. Programmeerimise õppevideol peab olema sissejuhatus
17. Programmeerimise õppevideol peab olema kokkuvõte

18. Programmeerimise õppevideo peab sisaldama demonstratsioone ehk tegevuse ja selgituse kooslust
19. Õppevideo koostamine on liiga aeganõudev
20. Millistel eesmärkidel võiksid üliõpilased õppevideoid kasutada?
21. Veel kommentaare õppevideote teemal

II. Küsitlus üliõpilastele

Tere, mina olen Andra Laura Meeksa ja teen bakalaureusetööd programmeerimise kursustel õppevideote kasutamise kohta. Antud küsitluse eesmärgiks on teada saada Teie tagasiside loodud õppevideotele ning Teie üldine suhtumine õppevideotesse programmeerimise kursustel.

1. Kas vaatasite Java andmestruktuuride õppevideot?
2. Java andmestruktuuride õppevideo sissejuhatusest tuli selgelt välja video eesmärk

Küsimused Java andmestruktuuride õppevideot vaadanud üliõpilastele:

3. Java andmestruktuuride õppevideo sisu oli arusaadav
4. Java andmestruktuuride õppevideos tulid välja liideste erinevused
5. Kommentaare Java andmestruktuuride õppevideole

Küsimus kõikidele vastajatele:

6. Kas vaatasite geneeriliste meetodite, klasside ja liideste õppevideot?

Küsimused geneeriliste meetodite, klasside ja liideste õppevideot vaadanud üliõpilastele:

7. Geneeriliste asjade õppevideo sissejuhatusest tuli selgelt välja video eesmärk
8. Geneeriliste asjade õppevideo sisu oli arusaadav
9. Geneeriliste asjade õppevideost tuli välja geneeriliste asjade ja kindla tüübiga asjade erinevus
10. Kommentaare geneeriliste meetodite, klasside ja liideste õppevideole

Küsimused kõikidele vastajatele:

11. Programmeerimise õppevideol peab olema sissejuhatus
12. Programmeerimise õppevideol peab olema kokkuvõte
13. Programmeerimise õppevideo peab sisaldama demonstratsioone ehk tegevuse ja selgituse kooslust
14. Õppevideod lihtsustavad programmeerimise õppimist
15. Oleksin valmis ise õppevideoid koostama
16. Millistel eesmärkidel võiks õppevideoid programmeerimise õpetamisel kasutada?
17. Kommentaare õppevideote kohta programmeerimise kursustel

III. Java andmestruktuuride videonäite käsikiri

Tere! Selle video teemaks on andmestruktuurid ja Java Collection Framework.

Videos kordan üle, mis on andmestruktuur ning tutvustan Collection liidese alamliideseid koodinäidete abil.

Liidesed, milles videos juttu tuleb on List, Set, Queue ja Deque. Video teises pooles tuleb juttu Map liidesest, mis ei kuulu Java Collection Frameworki.

Näidete jaoks kasutan IntelliJ arenduskeskkonda ning olen juba eelnevalt valmis teinud Java klassid, kus liideste realiseerimise ja kasutamise kohta näited tulevad.

Videos on erinevate liideste puhul näidatud enamkasutatavad meetodid nagu uue elemendi lisamine, elemendi sisalduvuse kontroll, elemendi kättesaamine ja elemendi eemaldamine. Rohkemaid meetodeid leiab Java APIst.

Collection liides

Collection liides kujutab endast ühemõõtmelist elementide kogu.

Collection liidesel on erinevaid alamliideseid, mille erinevus seisneb selles, et neis hoitakse, lisatakse ja eemaldatakse elemente erinevalt.

Collection on liides ehk abstraktsete meetodite komplekt, seega liidese realiseermiseks tuleb kasutada klassi.

List liides

List liidest olete juba varasemates praktikumides kasutanud, seega Listi kohta näited selles videos ei tule.

List liidest saab realiseerida lisaks ArrayListile ka näiteks LinkedListiga.

Nende vahe on selles, et ArrayList kasutab elementide hoidmiseks kasutatakse massiivi. LinkedList hoiab elemente mälus eraldi ja kasutab listi kooshoidmiseks viitasid. Mõlemal on omad eelised - ArrayList töötab paremini elemendi võtmisel indeksiga, aga LinkedList on andmete lisamisel etteotsa või keskele efektiivsem.

Set liides

Set liides ehk hulk on ühemõõtmeline elementide kogu, kus saab iga elementi olla vaid üks kord. Seti realiseerimiseks saab kasutada näiteks klassi HashSet. HashSet ei kindlusta, et elemendid jäävad hulka lisamise järjekorras.

Uue hulga olen juba valmis teinud Set liidesega, mida realiseerin HashSet klassiga. Hakkame elemente hulka lisama, lisame näiteks numbrid ühest neljani. Selleks kasutame add() meetodit:

```
hulk.add(1);        Hulk.add(2);        Hulk.add(3);        Hulk.add(4);
System.out.println(hulk);
```

Nüüd aga lisame hulka elemendi, mis hulgas juba olemas on ning näeme, et hulgas ei ole kahte samasugust elementi:

```
hulk.add(1); System.out.println(hulk);
```

Vaatame, kuidas kontrollida kas element on hulgas. Kõigepealt kontrollime, kas hulga on element 0 ning seejärel, kas hulgas on element 1:

```
System.out.println(hulk.contains(0));
System.out.println(hulk.contains(1));
```

Näeme, et contains() meetod tagastab false, kui elementi hulgas ei ole ning true, kui element on hulgas.

Hulga puhul elemente hulgast niisama kätte ei saa. Mida saab teha, on hulga läbimine tsükliga ning selle jooksul midagi elementidega teha, näiteks neid kuvada:

```
for(Integer element : hulk){ System.out.println(element); }
```

Elemente saab hulgast eemaldada remove meetodiga. Eemaldame näiteks elemendi 2:

```
hulk.remove(2); System.out.println(hulk);
```

Kui on soov, et hulga elemendid oleksid hulgas järjestatult, siis on hea kasutada liidest SortedSet ja selle realiseerimiseks klassi TreeSet. Muudame hulga liidese ja seda realiseeriva klassi ära ning vaatame, kuidas on elemendid hulgas. Ülejäänud osa kommenteerime hetkel välja.

```
SortedSet<Integer> hulk = new TreeSet<>();
```

Queue liides

Queue liides ehk järekord kujutab endast ühemõõtmelist FIFO (*first in first out*) elementide kogu. FIFO omadus tähendab seda, et elemente lisatakse lõppu ja kätte saab elemente ainult algusest. Queue realiseermiseks võib kasutada LinkedListi.

Uue järjekorra olen juba teinud, selleks kasutasin Queue liidest ja LinkedListi klassi. Hakkame järjekorda elemente lisama. Ka järjekorra puhul kasutame add() meetodit ning lisame arvud ühest neljani.

```
jarjekord.add(1); jarjekord.add(2); jarjekord.add(3); jarjekord.add(4);
```

Lisame veel ühe ühe järjekorda, siis näeme, et samu elemente võib järjekorras ka mitu olla:

```
jarjekord.add(1); System.out.println(jarjekord);
```

Vaatame, kuidas kontrollida, kas element on hulgas. Kõigepealt kontrollime, kas hulga on element 0 ning seejärel, kas hulgas on element 1:

```
System.out.println(jarjekord.contains(0));
System.out.println(jarjekord.contains(1));
```

Näeme, et contains() meetod tagastab false, kui elementi hulgas ei ole ning true, kui element on hulgas.

Kuna järjekorral saab kätte elemente algusest, siis esimese elemendi vaatamiseks saab kasutada meetodit peek():

```
System.out.println(jarjekord.peek());
```

Samuti käib elementi eemaldamine algusest, selleks saab kasutada meetodeid poll() ja remove(), mis on samaväärsed:

```
jarjekord.poll(); System.out.println(jarjekord);
jarjekord.remove(); System.out.println(jarjekord);
```

Deque liides

Deque liides on ühemõõtmeline double ended queue ehk elementide kogu, mis toetab elementide lisamist ja eemaldamist mõlemast otsast. Deque realiseerimiseks võib kasutada ArrayDeque.

Objektorienteeritud programmeerimise kursusel kasutakse Deque liidest ja ArrayDeque klassi magasini loomiseks. Magasin on LIFO ehk *last in first out* omadusega, kus magasini viimasena lisatud elemendid saadakse kätte kõige esimesena.

Uus magasin on juba loodud, kasutades Deque liidest ja ArrayDeque klassi. Jätkame sama näitega, kus lisame magasini numbrid ühest neljani.

Magasini elementide lisamiseks kasutatakse meetodit push(), mis lisab elemendi listi lõppu:

```
magasin.push(1);   magasin.push(2);   magsin.push(3);   magasin.push(4);
magasin.push(1); System.out.println(magasin);
```

Nagu näha, siis saab ka magasinis korduvaid elemente olla.

Vaatame, kuidas kontrollida, kas element on hulgas. Kõigepealt kontrollime, kas hulga on element 0 ning seejärel, kas hulgas on element 1:

```
System.out.println(magasin.contains(0));
System.out.println(magasin.contains(1));
```

Näeme, et contains() meetod tagastab false kui elementi hulgas ei ole ning true kui element on hulgas.

Elementide vaatamiseks saab kasutada peekLast() meetodit, mis näitab magasini viimasena lisatud elementi:

```
System.out.println(magasin.peekLast());
```

Elemendi eemaldamiseks kasutatakse magasini puhul pop() meetodit, mis eemaldab viimasena magasini lisatud elemendi:

```
System.out.println(magasin); Magasin.pop(); System.out.println(magasin);
```

Map liides

Map liides ehk kujutus on kahemõõtmeline elementide kogum. Map liideses hoitakse elemente nagu Pythoni dictionarys, kus kirje on kaheosaline. Kirje esimest komponenti nimetatakse võtmeks ning teist väärtuseks. Sama võtmega mitu kirjet kujutuses olla ei saa. Mapi realiseerimiseks võib kasutada HashMapi.

Uus kujutus on juba loodud, kasutades Mapi liidest ning HashMapi klassi. Lisame kujutusse kirjeid kujul nimi, mis on String ja vanus, mis on Integer.

Lisame kolm nime koos vanusega, kasutades selleks put() meetodit:

```
kujutus.put(“Mari”,20); kujutus.put(“Jüri”,22); kujutus.put(“Peeter”,30);
System.out.println(kujutus);
```

Vaatame, mis saab siis, kui lisame juba olemasoleva nimega uue vanuse:

```
kujutus.put(“Mari”, 15); System.out.println(kujutus);
```

Nagu hetk tagasi räägitud sai, siis korduvaid võtmeid kujutuses olla ei saa ning korduva võtme lisades jääb kujutusse võti uue väärtusega.

Kujutuse puhul saab containsKey() meetodiga kontrollida, kas kujutuses on olemas mingi võti. Näiteks kontrollime, kas kujutuses on nimed Maria ja Mari.

```
System.out.println(kujutus.containsKey(“Maria”));
System.out.println(kujutus.containsKey(“Mari”));
```

Nagu ikka, tagastab meetod false, kui võtit kujutuses ei ole ning true, kui võti kujutuses olemas on.

Get() meetodi korral saab kätte võtme väärtuse, näiteks tahame teada Jüri vanust, siis

```
System.out,println(kujutus.get(“Jüri”);
```

Võtme ja tema väärtuse eemaldamiseks saab kasutada remove() meetodit, näiteks kustutame Peetri ja tema vanuse kujutusest:

```
kujutus.remove(“Peeter”); System.out.println(kujutus);
```

Kui on soov, et kirjed oleksid kujutuses võtmete järgi sorteeritud, siis võib kasutada liidest SortedMap ja selle realiseerimiseks klassi TreeMap.

```
SortedMap<String, Integer> kujutus = new TreeMap<>();
```

Kokkuvõte

Kokkuvõteks kordan üle veel iga käsitletud liidese põhiomadused:

Seti ehk hulga puhul ei saa olla hulgas korduvaid elemente.

Queue ehk järjekorra puhul on tegu FIFO (*first in first out*) omadusega, kus elemente lisatakse lõppu ning kätte saab algusest.

Deque ehk antud juhul magasini korral on tegu LIFO (*last in first out*) omadusega, kus elemente lisatakse lõppu ning saadakse ka kätte lõpust.

Mapi ehk kujutuse korral on tegu kirjetega, kus kirje esimene osa on võti ja teine osa on võtmele vastav väärtus.

Sellega on video lõppenud, tänan vaatamast.

IV. Geneeriliste meetodite, klasside ja liideste videonäite käsikiri

Tere. Antud video teemaks on geneerilised liidesed, klassid ja meetodid. Ka Java Collection Framework koosneb geneerilistest liidestest. Geneeriliste andmestruktuuride korral on jäetud tüübid lahtiseks ehk mingi kindla tüübi, näiteks Integeri, asemel on kasutatud tüübiparameetreid.

Sageli kasutatavad tüübiparameetrid:

E – Element (kasutatakse palju Java Collections Frameworkis)

K – Key (võti)

N – Number (arv)

T – Type (tüüp)

V – Value (väärtus) jne

Tüübiparameetreid võib ka ise tähistada, näiteks Tuup või midagi muud. Hea tava on tähistada tüübiparameetrid ühe suure tähega.

Geneerilised meetodid

Koodi kirjutades võib juhtuda, et oleks vaja kirjutada mitu sarnast meetodit erinevatele tüüpidele. Näiteks erinevat tüüpi massiivi kuvamiseks võib kirjutada mitu meetodit, neist igaüks erinevale tüübile.

Vaatame koodinäidet, kus me tahame kolme erinevat tüüpi massiivi kuvamise meetodit luua. Olgu tüübid Integer, Double ja Character. Need massiivid on juba valmis tehtud.

Kõigepealt teeme valmis meetodi Integeri tüüpi massiivi väljastamiseks:

```
public static void printArray(Integer[] inputArray){
        for (Integer element : inputArray) System.out.print(element+" ");
        System.out.println();}
```

Nüüd on meil olemas meetod Integer tüüpi massiivi väljastamiseks. Kui me nüüd Double tüüpi massiivi väljastamiseks meetodit tahame, siis võime teha uue meetodi, kus muudame ära massiivi ja massiivi elemendi tüübi:

```
public static void printArray(Double[] inputArray){
        for (Double element : inputArray) System.out.print(element + " ");
        System.out.println(); }
```

Character tüüpi massiivi väljastamiseks võime samuti muuta massiivi ja massiivi elemendi tüübi.

```
public static void printArray(Character[] inputArray){
        for(Character element : inputArray) System.out.print(element+" ");
        System.out.println(); }
```

Nende kolme meetodi ainsaks erinevuseks ongi massiivi tüüp ning massiivi elemendi tüüp „for“ tsüklis. Selleks, et vältida väga sarnase sisuga meetodite kirjutamist, saab kirjutada geneerilise meetodi, kus on kasutatud kindla tüübi asemel tüübiparameetreid. Selleks võime võtta ette ühe printArray() meetodi ning muuta kindlad tüübid tüübiparameetriks. Tähistame selle E ehk Elemendiga. Selle testimiseks kommenteerime teised meetodid välja.

```
public static void printArray(E[] inputArray){
        for (E element : inputArray) System.out.print(element + " ");
        System.out.println(); }
```

E võib asendada ka enda tüübiparameetri tähistusega, näiteks Tuup (näite E asendamisest Tuup-iga).

Tüüpide piiramine

Tüüpe on võimalik ka piirata, et nad ei oleks päris vabad. Lisades tagastustüübiks T ning selle juurde <T extends Comparable<T>>, siis piiranguks on see, et kasutatav tüüp peab realiseerima Comparable liidest.

Näiteks kui me soovime leida kahe elemendi seast suurimat, siis saame teha sellise meetodi:

```
public static <T extends Comparable<T>> T maksimaalne(T x, T y){
        if(x.compareTo(y) > 0){ return x; }
        else{ return y; }
    }
```

Kontrollime, mis tulemuse saame kahe Integeri ja kahe Double’i puhul:

```
System.out.println(maksimaalne(1, 2));

System.out.println(maksimaalne(1.2, 1.1));
```

Geneerilised klassid

Ka klassid võivad olla geneerilised ehk hoida enda sees suvalist tüüpi objekti.

Teeme näiteks klassi GeneerilineKlass. Selles klassis võib hoida suvalist tüüpi objekti. Klassi loomisel peab selle tüübiparameetri väärtustama.

Konstruktorisse teeme T tüüpi isendivälja, set() ja get() meetodid.

```
public class GeneerilineKlass<T> {
    private T element;
    public void set(T element){ this.element = element; }
    public T get(){ return element; }
}
```

Antud klassi katsetamiseks olen loonud testklassi koos main meetodiga. Loome nüüd main meetodisse uue Integer tüüpi objekti kasutades GeneerilineKlass klassi.

```
GeneerilineKlass<Integer> integeriKlass = new GeneerilineKlass<>();
```

Lisame objektile elemendi 1 ning vaatame, mida annab meile get() meetod:

```
integeriKlass.set(1); System.out.println(integeriKlass.get());
```

Loome nüüd ka uue String tüüpi objekti kasutades seda sama GeneerilineKlass klassi:

```
geneerilineKlass<String> stringiKlass = new geneerilineKlass<>();
```

Lisame objektile elemendi “tere!” ning vaatame, mida annab meile get() meetod:

```
stringiKlass.set("tere!"); System.out.println(stringiKlass.get());
```

Nagu näha toimib meie tehtud geneeriline klass erinevate tüüpide puhul.

Geneerilised liidesed

Geneeriliste liideste puhul on võimalik samuti erinevaid tüüpe realiseerimisel kasutada.

Antud List liidese puhul on võimalik kasutada erinevaid tüüpe näiteks Integer, String jne. Seda näitab <E> ehk Element. Ka geneerilise liideseid saame ise luua meie isetehtud geneerilisele klassile sarnaselt.

Kokkuvõte

Kokkuvõtteks kordan üle geneeriliste meetodite, klasside ja liidestega seonduvad olulisemad asjad.

Geneeriliste meetodite, klasside ja liideste puhul on tüübid jäetud lahtiseks ehk kasutatakse tüübiparameetreid.

Enim kasutatavad tüübiparameetrid on E – Element, K – Key (võti), N – Number (arv), T – Type (tüüp) ja V – Value (väärtus).

Tüübiparameetrit võib ka ise tähistada, heaks tavaks on üks suurtäht.

Sellega on video lõppenud, tänan vaatamast!

V. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Andra Laura Meeksa**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   Õppevideote rakendamine programmeerimise kursustel,
   (*lõputöö pealkiri*)

   mille juhendaja on Eno Tõnisson,
   (*juhendaja nimi*)

   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **14.05.2018**