TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Georg Demidov

Objektorienteeritud programmeerimise õppematerjalide koostamine Tartu Ülikooli kursusele “Programmeerimine”

Bakalaureusetöö (9 EAP)

Juhendajad: Helle Hein PhD, Tauno Palts PhD

Tartu 2022

Objektorienteeritud programmeerimise õppematerjalide koostamine Tartu Ülikooli kursusele “Programmeerimine”

Lühikokkuvõte:

Tartu Ülikooli kursust "Programmeerimine" oleks kasulik täiendada objektorienteeritud programmeerimise õppematerjalidega, sest kursusel ei käsitleta objektorienteeritud programmeerimist. Selle jaoks oleks vajalik luua kahe nädala jagu praktikumi- ja koduülesandeid ning õpiku peatükke, mis tutvustaksid uut teemat. Objektorienteeritud programmeerimise teema tutvustus on oluline, sest see aitab tudengitel paremini mõista, kuidas saab suuremat ülesannet lahti harutada väiksemateks alamülesanneteks. Lisaks on esimese aasta informaatika eriala tudengitel kevadsemestril õppeaine „Objektorienteeritud programmeerimine“. Uute materjalide abil tehakse tudengitele sissejuhatus kursusesse ning antakse esialgne ülevaade objektorienteeritud programmeerimise põhitõdedest.

Võtmesõnad: Objektorienteeritud programmeerimine, ADDIE mudel, Merrilli mudel, õppematerjalid

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

Development of object-oriented programming instructional materials for the University of Tartu course "Programming"

Abstract:

There is a need to create instructional materials on object-oriented programming for the course "Programming" at the University of Tartu, as the course does not cover object-oriented programming. For this purpose, it is necessary to create two-week-long practical and homework exercises along with textbook chapters introducing the new topic. It is vital to introduce the topic of object-oriented programming, because it will help students to understand how a larger task can be decomposed into smaller sub-tasks. In addition, first-year computer science students have a course on "Object-Oriented Programming" in the spring semester. By means of new study materials, students will get an introduction to the course and a preliminary overview of the basics of object-oriented programming.

Keywords: Object-oriented programming, ADDIE model, Merrill model, instructional materials

CERCS: P175 Informatics, Systems Theory, S270 Pedagogy and Didactics.

Sisukord

1. Sissejuhatus **4**

2. Kursus “Programmeerimine” **5**
2.1 Uute materjalide vajadus **5**
2.2 Õppeaine „Programmeerimine“ ülesehitus **5**
2.3 Ümberpööratud klassiruum **7**

3. Ülevaade objektorienteeritud programmeerimisest **9**
3.1 Objekt **9**
3.2 Klass ja meetodid **9**
3.3 Pärilikkus ja polümorfism **10**

4. Õppematerjalide koostamise mudelid **11**
4.1 ADDIE mudel **11**
4.2 Merrilli mudel **13**

5. Metoodika **15**
5.1 Analüüs **15**
5.2 Kavandamine **16**
5.3 Arendamine **17**
5.4 Materjalide loomine **18**

6. Tulemused **19**
6.1 Õppematerjalide koostamise protsess **19**
6.2 Õppematerjalide tulevik ja edasiarendused **23**

7. Kokkuvõte **24**

8. Kasutatud kirjandus **25**

Lisad **28**
I Loodud õppematerjalid **28**
II Litsents **29**

1. Sissejuhatus

Õppeaine „Programmeerimine“ on Tartu Ülikooli üks suurimatest ainetest, milles osaleb iga aasta keskmiselt 400 üliõpilast [1]. Õppeaine populaarsus on tingitud sellest, et kursus annab põhjaliku ülevaate programmeerimise põhitõdedest, mille abil saavad aine läbinud tudengid hakata ise lihtsamaid programme koostama ja rakendama omandatud oskuseid enda eriala valdkonnas [1].

Esimese aasta kevadsemestril on informaatika tudengitel kohustuslik õppeaine „Objektorienteeritud programmeerimine“. Kursus tutvustab tudengitele esmakordselt programmeerimiskeelt Java ning objektorienteeritud programmeerimise aluseid. Paljudel informaatika tudengitel võib esialgu esineda raskusi uue programmeerimiskeele ja kontseptsiooni teadmiste omandamisega, mida käsitleb kursus “Objektorienteeritud programmeerimine”.

Õppeaines „Programmeerimine“ läbitakse rekursiooni teemat kahe nädala jooksul [2]. Rekursiooni teemat läbitakse lisaks ainele “Programmeerimine” ka ainetes “Programmeerimine II” ning “Algoritmid ja andmestruktuurid” [3]. Tegemist on kursuse “Programmeerimine” jätkukursustega, milles rekursioon on läbiv teema. Eelmainitud asjaolude tõttu otsustasid aine õppejõud kursusel „Programmeerimine“ asendada rekursiooni teema oluliselt objektorienteeritud programmeerimise teemaga jättes rekursioonist alles vaid sissejuhatuse.

Bakalaureusetöö eesmärk on luua aine „Programmeerimine“ jaoks õppematerjale objektorienteeritud programmeerimise kohta. Selleks on vaja luua kahe nädala ulatuses õppematerjale, mille hulka kuuluvad praktikumi- ja koduülesanded ning kaks õpiku peatükki. Töö käigus valminud õppematerjalid tutvustavad objektorienteeritud programmeerimise aluseid programmeerimiskeele Python baasil.

Bakalaureusetöö on jagatud viieks osaks. Esimeses osas antakse ülevaade kursuse „Programmeerimine“ ülesehitusest ja kasutatavast õppemetoodikast. Teises osas tutvustatakse objektorienteeritud programmeerimise põhilisi kontseptsioone. Kolmandas osas analüüsitakse kahte õppematerjalide koostamise mudelit, milleks on ADDIE ja Merrilli mudelid. Neljandas peatükis kirjeldatakse metoodikat, mida kasutati õppematerjalide koostamisel. Viimases peatükis tutvustatakse loodud õppematerjale ja tuuakse välja nende võimalikud edasiarendused.

2. Kursus “Programmeerimine”

Selles peatükis kirjeldab autor objektorienteeritud materjalide olulisust kursuse “Programmeerimine” jaoks. Antakse ülevaade kursuse ülesehitusest ja hindamissüsteemist. Uuritakse ümberpööratud klassiruumi õppemetoodikat, mille alusel toimub antud kursusel õppetöö. Autor toob välja eelmainitud õppemetoodika tugevused ja nõrkused.

2.1 Uute materjalide vajadus

Informaatika õppekava üliõpilaste jaoks on kursus “Programmeerimine” kohustuslik [1]. Aine on kohustuslik ka paljude teiste erialade jaoks ning kursust läbitakse samuti vabaainena. Tudengitele, kes ei õpi informaatika õppekaval, valmistab rekursiooni teema raskusi.[1] Lisaks õpetatakse rekursiooni teemat põhjalikult kursuse “Programmeerimine” jätkuainetes “Programmeerimine II” ja “Algoritmid ja andmestruktuurid” ning muudes informaatika õppekava ainetes. Sellest tingitult leidsid kursuse eest vastutavad õppejõud, et rekursiooni teema pole nii suures mahus vajalik kursuse „Programmeerimine“ õppekavas. Sellest tulenevalt tekkis vajadus uue teema integreerimise järele. Paljud tudengid leidsid, et aine „Objektorienteeritud programmeerimine“ uued kontseptsioonid on algul raskesti mõistetavad ja et oleks kasulik, kui neid oleks tutvustatud juba esimese semestri jooksul. Seega asendatakse kursusel „Programmeerimine“ rekursiooni teema osaliselt objektorienteeritud programmeerimise teemaga.

2.2 Õppeaine „Programmeerimine“ ülesehitus

Õppeaine „Programmeerimine“ kestab 16 nädalat ning iga nädal on tudengitele ette nähtud loengud, milles tutvustatakse uusi teemasid; praktikumid, kus lahendatakse ülesandeid, ning kodutööd, mis on mõeldud iseseisvaks lahendamiseks ja loengus tutvustatud teemade kinnistamiseks [2]. Igas praktikumis on ette nähtud koduülesannete lahendamine, juhul kui ei toimu kontrolltöid või projektide esitlusi.

Kokku on semestri jooksul 13 kodutööd, mis koosnevad ühest või enamast ülesandest ning 13 praktikumitööd, mis on mahu poolest suuremad kui koduülesanded. Suurem maht on

[1] Kursuse “Programmeerimine” eest vastutavatel õppejõududel ja kursuse õppeassistentidel toimuvad aine õpetamise jooksul iganädalased koosolekud. Töö autor on olnud kursuse õppeassistent 2020/21 ja 2021/22 õppeaastatel. Nii koosolekute käigus kui ka tudengite tagasisidest leiti, et rekursiooni teema on liiga keeruline kursusel osalejate jaoks, ning et rekursiooni teemat läbitakse kursuse jooksul liiga suures mahus. Kuna teemat käsitletakse põhjalikult kursuse “Programmeerimine” jätkukursustel “Programmeerimine II” ja “Algoritmid ja andmestruktuurid”, siis leiti, et rekursiooni teema asendamine on põhjendatud.

vajalik selleks, et ka edasijõudnud tudengitele jätkuks praktikumi jooksul ülesandeid. Iga praktikumi- ja kodutöö eest on võimalik saada kuni 0,5 punkti ehk kokku maksimaalselt 13 punkti. Alampiiriks on 7 punkti ja lävendi mittetäitmine tähendab üliõpilasele kursusel läbikukkumist. Loengumaterjalide kinnistamiseks on loodud testid, mille korreksel lahendamisel on võimalik teenida maksimaalselt 7 punkti. Kontrolltöid on kaks ja need koosnevad kahest osast: arvestuslikust ja punktilisest osast. Punktilise osa eest on võimalik saada kuni 20 punkti ning seda juhul kui arvestuslik osa on sooritatud. Kursuse jooksul peab iga tudeng koos paarilisega koostama projekti, milles luuakse programm kasutajaliidesega. Projekti eest saab punkte kahes osas: projekti esialgse ja lõpliku versiooni eest. Mõlemad osad annavad 5 punkti ja projekti alampiiriks on 5 punkti. Eksam on üles ehitatud analoogselt kontrolltöödega: arvestuslik osa ja punktiline osa. Punktiline osa jaotub kaheks, milles on võimalik teenida 10 punkti teoreetiliste ülesannete eest ja 20 punkti programmeerimise ülesannete eest. Aktiivsematel üliõpilastel on võimalik ka saada lisapunkte keerulisemate ülesannete lahendamise eest. Kokku on 10 lisaülesannet, millest igaüks annab ühe punkti. Kursuse jooksul on võimalik teenida kokku kuni 110 punkti (vt tabel 1).

Tabel 1. Kursuse “Progammeerimine” hindamisskeem.

	Aeg	**Alampiir**	**Max**	**Kommentaar**
Testid	1.-16. nädal		7	0,5 punkti testist
Kodutöö+praktikum	1.-16. nädal	7	13	0,5+0,5 kodutööst ja praktikumist
Projekt	7.-16. nädal	5	10	Kahes osas (5+5)
1. kontrolltöö	6. nädal	Arvestusliku osa arvestus	20	Kahes osas: arvestuslik ja punktiline osa
2. kontrolltöö	12. nädal	Arvestusliku osa arvestus	20	Kahes osas: arvestuslik ja punktiline osa
Eksam	jaanuaris	Arvestusliku osa arvestus	30	Kahes osas: arvestuslik ja punktiline osa
Lisaülesanded			10	

Tavaline nädal näeb õppeainet „Programmeerimine“ võtvale tudengile välja järgmine: loenguvideote vaatamine ja kaasatöötamine, õpiku materjali lugemine ning materjali läbitöötamine, koduülesannete lahendamine loengu ja õpiku materjalide põhjal ning osalemine praktikumis, kus kinnistatakse uusi teemasid [4]. Sellisel kujul õpetamise eeliseks on see, et praktikumis ei pea aega kulutama uute teadmiste õpetamisele ja saab kohe asuda

ülesannete lahendamise juurde. Eelmainitud õppemetoodika on tuntud kui ümberpööratud klassiruum (ingl *flipped classroom*) [5].

2.3 Ümberpööratud klassiruum

Nwosisi, Ferreira, Rosenberg ja Walsh [5] järgi on ümberpööratud klassiruumi mudel järgmine: õpilased omandavad iseseisvalt uut teemat, tutvudes vastavate materjalidega. Tundides kinnistavad õpilased uusi teadmisi, lahendades praktilisi ülesandeid ja saades vajadusel juhendajalt abi. Sellise õpetamise peamisteks eelisteks on ajavõit kontaktõppes kohtumisel, sest ei pea kulutama aega teooria õpetamisele ning saab ülesannete lahendamise juurde asuda. Tundide juhendaja saab keskenduda õpilastele, kes nõuavad õppimisel rohkem aega samal ajal kui õpilased, kes uue teema kiiremini omandavad, saavad iseseisvalt kaasõppijaid aidata.

Awidi ja Painter [6] kirjeldavad, kuidas ümberpööratud klassiruumi õppele üle läinud kursused saavad säästa nii rahalisi kui ajalisi ressursse õpetamisel. Lisaks on sellistel kursustel ka lihtsam toime tulla uute tehnoloogiate integreerimisega õppesse. Ümberpööratud klassiruumi õppe integratsioon tõstab õpilaste osavõttu tundides, sest neil tekib võimalus enda teadmisi kinnistada praktiliste ülesannete abil ja vajadusel üle täpsustada ebaselgeks jäänud teemad. Tundidest osavõtmise motivatsioon kasvab suurel määral teadmisest, et tunni jooksul ei pea kulutama aega teooria õppimisele, vaid saab kohe asuda ülesannete lahendamise juurde.

Lisaks leiavad Nwosisi, Ferreira, Rosenberg ja Walsh [5], et ümberpööratud klassiruum viib õpilased paremate tulemusteni. Selleks viidi läbi uuring, kus valiti välja kaks kursust, mis korraldati ümber nii, et nad kasutaks ümberpööratud klassiruumi õpet. Võrreldes eelneva aasta tulemustega, tõusis mõlemal aastal kursuse läbinud tudengite keskmine punktisumma vähemalt 2,6% võrra (vt tabel 2). Kursust mitteläbinud õpilaste osakaal vähenes märkimisväärselt: esimesel kursusel 33% võrra ja teisel kursusel 15% võrra. Seega saab järeldada, et edukalt koordineeritud ümberpööratud klassiruumi õpe vähendab mitteläbinud õpilaste osakaalu, viib kõrgemate tulemusteni ja paremate teadmisteni õpilaste hulgas.

Tabel 2. Keskmiste punktisummade võrdlus kursuste edukalt sooritatud õpilaste hulgas koos ümberpööratud klassiruumi õppemeetodikaga ja ilma.

Kursuse nimetus	Keskmine hinne ümberpööratud klassiruumi õppega	Keskmine hinne ilma ümberpööratud klassiruumi õppeta	Tulemuste parandus
Cisco Networking Basics	85,1	82,5	2,6%
Adult Development in the Workplace	90,5	87	3,5%

Peamiseks puuduseks ümberpööratud klassiruumi puhul on keerukus kohaneda uue õpetamismetoodikaga [6]. Õpilastel on raske harjuda asjaoluga, et õpetamisviis ei ole enam traditsiooniline ja et materjalide selgeks saamine tuleb läbi iseseisva töö. Õpilaste tagasisidest saab järeldada, et valmidus selliseks õpetamismetoodikaks oleks suurem, kui nad oleks sellega juba varajases eas tutvunud [6]. Lisaks on leitud, et ümberpööratud klassiruumi õppemetoodikaga kursuse kavandamine on ajakulukas [7]. Selleks, et õpilased saaksid uued teemad edukalt selgeks, on vaja koostada õppematerjalid, mis ka sellist õpet toetavad. Iseseisvalt õpitavate materjalide koostamine on paljude kursuste jaoks suureks väljakutseks. Vastutavad õppejõud peavad pühendama palju aega uute materjalide arendamisele ning kursus peab pakkuma õpilastele digitaalseid lahendusi, millega saab mugavalt ja arusaadavalt uusi materjale omandada.

3. Ülevaade objektorienteeritud programmeerimisest

Selles peatükis tutvustatakse objektorienteetitud programmeerimise põhitõdesid. Selgitatakse, kuidas saab kirjeldada objekti olemust ja käitumist. Tuuakse välja, milleks on vajalik klass, ning millist funktsionaalsust see võimaldab. Lisaks annab autor ülevaate alam- ja ülemklassidest, pärilikkusest ja polümorfismist.

3.1 Objekt

David Amos [8] leiab, et objektorienteeritud programmeerimise puhul on kõige olulisem objektide loomine. Objekt on andmetüüp, mis koosneb erinevatest muutujatest ja meetoditest. Muutujad defineerivad objekti olemuse ning meetodid objekti käitumise. Objektide abil on programmid selgemad ja struktureeritumad.

Näiteks võib tuua tavalise inimese. Inimese olemust võime kirjeldada võtmesõnadega nagu pikkus, juukse värvus, rahvus, sugu jne. Objektorienteeritud programmeerimises eelmainitud omadused moodustaksid objekti muutujad. Meetodite ehk inimese käitumise alla läheksid tegevused nagu söömine, õppimine, magamine, trenni tegemine jms.

3.2 Klass ja meetodid

Objekte võib vaadata kui andmestruktuure, mis on seotud kindla käitumise ja omadustega [9]. Koostades programme, milles tuleb jaotada ülesanne väiksemateks alamülesanneteks, on vaja mitmeid erinevaid objekte. Selleks, et defineerida objekte ja nende käitumist, peab looma klasse. Klass on andmestruktuur, mis defineerib objekti olemuse ja käitumise, ning objekt on klassi isend.

Dusty Phillipsi sõnul [9] saab objekti olemust kirjeldada andmete ehk omaduste põhjal. Klassi sees on võimalik määrata palju erinevaid omadusi ja kõik need omadused võivad olla nii erinevat kui ka samat andmetüüpi. Peamine on see, et omadused oleksid unikaalsete nimetustega. Klassi isenditel võivad olla nii erinevad kui ka samad omadused identsete väärtustega. Kui tulla tagasi eespool toodud näite juurde, siis päris elus võib olla mitmel erineval inimesel sama pikkus, kaal, juuste värvus jne.

Fanghao Yang ja Jonathan E. Menard väidavad [10], et objekti käitumist võib kirjeldada läbi tegevuste, mis võivad objektiga juhtuda. Objekti käitumist saab defineerida klassis ja seda saab teha meetodite abil. Klassis olevaid meetodeid võib vaadelda justkui tavaliste

funktsioonidena, kuid klassis paiknevatel meetoditel on ligipääs kõikidele omadustele, mis on seotud objektiga. Sarnaselt funktsioonidega on meetoditel ka võimalus kasutada parameetreid ja tagastada erinevaid väärtuseid.

Dusty Phillips [9] kirjeldab, et luues objekti, pöördutakse erilise meetodi poole, mida nimetatakse konstruktoriks. Konstruktor on meetod, mis vastutab objekti isendimuutujate seostamise eest klassi konkreetse isendiga. Konstruktor võtab parameetriteks nii palju argumente kui soovitakse objektiga siduda. Kui luuakse klassi isend, siis see initsialiseeritakse läbi konstruktori ning kõik omadused seostatakse klassi selle konkreetse isendiga.

3.3 Pärilikkus ja polümorfism

Erinevaid objekte luues võib tekkida olukord, mis nõuab, et objektidel oleksid identsed isendimuutujad, ent objektid käituksid erinevalt. Selleks on võimalik luua ülemklasse, mis pärandavad kõik enda käitumise ja omadused alamklassidele [9]. Pärilikkuse ainus eelis ei seisne ainult koodi taaskasutamises, vaid ka programmeerijate võimaluses rakendada päriselu loogikat enda programmides [10]. Kõik olemasolevad ülemklassi meetodid on kasutatavad alamklassides vajaduseta neid uuesti defineerida. Sama loogika kehtib ka isendimuutujatega: luues alamklasside isendeid, peab väärtustama samu isendimuutujaid nagu ka ülemklassis [11].

Ülemklassid pärandavad kõik meetodid alamklassidele, ent kohati on vajalik, et pärandatud meetod käituks alamklassis erinevalt. Selleks on võimalik pärandatud meetodeid üle kirjutada ehk kasutada polümorfismi [12]. Polümorfismi abil saab alamklassidesse pärandatud meetodeid muuta vastavalt vajadusele [9]. Meetodi ülekirjutamiseks on vaja alamklassis defineerida ülemklassi meetod identse päisega - sama nimetus ja parameetrid. Vastasel juhul luuakse alamklassi lihtsalt uus meetod ja päritud meetod säilitab ülemklassis defineeritud käitumise.

4. Õppematerjalide koostamise mudelid

Materjalide koostamisel lähtutakse mudelitest, mille abil on võimalik jagada materjalide loomine etappideks. Antud peatükis kirjeldab autor ADDIE ja Merrilli mudelit, mille põhjal koostati mõlemad õpiku peatükid ning praktikumi- ja koduülesanded.

4.1 ADDIE mudel

Saltsman ja Shelton [13] kirjeldavad ADDIE mudelit kui ühte enim levinud materjalide koostamise mudelit. Tegemist on viiesammulise süstemaatilise materjalide väljatöötamise mudeliga, mida saab kasutada nii e-õppes kui ka traditsioonilise õppe puhul. ADDIE mudeli alusel materjalide koostamine on olnud edukas e-õppe rakendamisel, sest mudel on õpilasekeskse lähenemisega [14]. Varasemate mudelite lähenemine materjalide loomisele oli juhendajakeskne, ent e-õppe puhul kaotasid materjalid juhendaja kaasamiseta oma väärtuse.

Iga mudeli samm on kooskõlas õpiväljunditega, mis on omakorda välja kujunenud pärast põhjalikku analüüsi õpilaste vajaduste kohta [15]. Etappide vahel võib täheldada kattuvust, ent sellele vaatamata pakuvad need sammud üheskoos paindlikke ja läbimõeldud suuniseid tõhusate õppematerjalide arendamiseks. Need viis etappi on järgmised [15]:

- Analüüs (ingl *analysis*);
- Kavandamine (ingl *design*);
- Arendus (ingl *development*);
- Rakendamine (ingl *implementation*);
- Hinnang (ingl *evaluation*).

Analüüsi etapp on esimene ja kõige olulisem mudeli etapp. Analüüsi etapi olulisus peitub selles, et põhjaliku analüüsi tagajärjel saab kursuse koostamisel säästa oluliselt ressursse ja aega. Etapp jaguneb järgnevaks neljaks sammuks [15]:

1. Analüüs õppija kohta (ingl *analysis of the learner*) - õppijate eelnevate teadmiste, oskuste ja vajaduste kaardistamine uue teema kohta. Seda arvesse võttes saab planeerida, mida peaksid õppematerjalid sisaldama.
2. Analüüs materjalide eesmärgi kohta (ingl. *analysis of instructional goals*) - milliseid teadmisi uued materjalid edastavad õpilastele. Õppematerjalide eesmärkide kaardistamine aitab jõuda võimalikult tõhusa lõpp-produktini.

3. Materjalide eesmärkide arendus (ingl *developing instructional analysis*) - tegevuste kaardistamine, mis aitab eelmises etapis seatud eesmärgid täita;
4. Õpiväjundite arendus (ingl *developing learning objectives*) - selles etapis on vaja välja selgitada, milliseid tulemusi oodatakse õpilaselt, kes on materjalid selgeks saanud. Tulemuste hinnanguks on vaja seada kindlad lävendid, mida õpilaselt oodatakse. Näiteks peab kursuse läbinud õpilane olema võimeline korrektselt selgitama polümorfismi mõistet.

Peale põhjaliku analüüsi läbiviimist saab liikuda kavandamise etapi juurde. Selles etapis koostatakse analüüsis kogutud andmete põhjal plaan, mille järgi hakatakse koostama õppematerjale [16]. Korrapärase ja sidusa plaani koostamine on eelduseks edukate õppematerjalide valmimisel [13]. Analüüsis seatud õpiväljundite saavutamine on lihtsustatud ning vähendab arusaamatuste tekkimist õpilaste seas uute materjalide omadamisel.

Kavandamise etapi väljunditest saab sisendit arenduse etapiks [14]. Selle etapi töömaht sõltub eelmise kahe etapi edukusest. Juhul, kui analüüsi ja kavandamise etapiga koguti piisavalt andmeid ja pandi paika pädev plaan, siis arenduse etapis on töö lihtsustatud [15]. Selle etapi käigus tehakse kindlaks, milliseid tehnoloogiad kasutatakse materjalide edastamiseks õpilastele. Arenduse etapi lõpuks valmivad uued materjalid, mis on valmis kasutamiseks koos sobivate tehnoloogiliste lahendustega [16].

Rakenduse etapis jõuavad uued materjalid õpilasteni [14]. Selle etapi puhul on oluline, et valminud materjalid oleksid edastatud õpilastele mugaval ja arusaadaval kujul [15]. Mahukate õppematerjalide valmimisel viiakse läbi ka katseid, milles väiksem kontrollrühm lahendab kõik materjalid läbi [16]. Lahendamise käigus saadakse esialgne hinnang materjalide ajakulust ja võimalikest takistustest. Kui valminud õppematerjalide maht ei ole liiga suur, siis materjalide lahendamine kontrollrühma poolt ei ole vajalik.

Viimase etapi ehk hinnangu eesmärk on saada tagasisidet materjalide kohta. Hinnatakse valminud materjalide efektiivsust ja kasulikkust [16]. On olemas kaks erinevat tüüpi hinnangut: pidev ja kokkuvõtlik hinnang [14]. Pideva hinnangu puhul hinnatakse saavutatud tulemusi iga etapi järel. Kokkuvõtvat hinnangut kogutake õpilastelt, kes on materjalid juba läbi töötanud. Kogutud tagasiside abil tehakse vajalikud parandused selleks, et tulevikus oleksid materjalid paremini mõistetavad ja sidusamad.

4.2 Merrilli mudel

Haridusteadlane Marriner David Merrill [17] leiab, et kõige efektiivsemad õppemudelid toetavad probleemikeskset õpet. Selliseid mudeleid on võimalik jaotada neljaks erinevaks etapiks:

- Eelnevate kogemuste aktiveerimine (ingl. *activation of prior experiences*);
- Oskuste demonstreerimine (ingl. *demonstration of skills*);
- Oskuste rakendamine (ingl *application of skills*);
- Oskuste lõimimine igapäevaelu tegevustesse (ingl. *integration of these skills into real-world activities*).

Nendest etappidest kujunesid välja 5 põhiprintsiipi, mille kasutamisel on õpe kõige mõtestatum. Merrilli põhiprintsiipe järgides kasvab õppijate huvi õppeprotsessi vastu ning sellega ka nende aktiivsus [18]. Neid printsiipe on defineerinud Merrill järgmiselt:

- Demonstreerimisprintsiip (ingl *the demonstration principle*) - õppija mõistab õpitut läbi näidete. Näite abil tehakse selgeks, kuidas on võimalik uusi teadmisi rakendada.
- Rakendusprintsiip (ingl *the application principle*) - õppija kinnistab õpitud materjali seda rakendades;
- Probleemikeskne printsiip (ingl *the problem-centered principle*) - õppija kinnistab teadmisi, osaledes ülesandekeskses õppimisstrateegias;
- Aktiveerimisprintsiip (ingl *the activation principle*) - õppimine on edukam, kui õppija kasutab õppe jooksul eelnevaid teadmisi või kogemusi;
- Integreerimisprintsiip (ingl *the integration pricniple*) - õppija mõistab paremini õpitut, kui ta rakendab uusi oskuseid ja teadmisi igapäevaelus.

J. Gardner [19] viis läbi uuringu, milles uuris, kuidas Merrilli põhiprintsiipide rakendamine mõjutab Utah ülikooli tudengite õppeedukust. Uuringu käigus leiti, et õpilaste probleemikesksete ülesannete lahendamine paranes oluliselt. Õpilaste huvi õppe vastu kasvas samuti olulisel määral ja sellega ka õppeaktiivsus tundide ajal.

Merrilli põhiprintsiipidest kujunesid välja ülesandekesksed õppestrateegiad. Merrilli õppestrateegiate abil saab luua õppematerjale, mis toetavad eelmainitud printsiipe. Õppestrateegiaid on neli [20]:

- Selgitus (ingl *tell*);

- Näide (ingl *show*);
- Meenutamine (ingl *recall*);
- Rakendamine (ingl *apply*).

David Merrill [20] kirjeldab, kuidas peaks kasutama eelmainitud õppestrateegiaid materjalide loomisel. Selgituse osas peab tutvustama uusi materjale koos definitsioonidega. Informatsiooni kinnistamiseks peab esitama näiteid. Meenutamise abil kontrollitakse, kas õpilane mõistis näiteid uue materjali kohta. Seda saavutatakse, kasutades kontrollküsimusi. Rakenduse faasis peab õpilane olema võimeline demonstreerima õpitut. Teadmisi kontrollitakse ülesande põhjal, mida õpilane peab olema võimeline iseseisvalt lahendama.

5. Metoodika

Selles peatükis annab autor ülevaate metoodikast, mida kasutati materjalide loomisel. Materjalide planeerimine ja kavandamine toimus ADDIE mudeli põhjal. Õpiku peatükid ning praktikumi- ja koduülesanded koostati järgides Merrilli mudeli õppestrateegiat vastupidises järjekorras.

5.1 Analüüs

Õppematerjalide loomisel lähtus töö autor ADDIE mudelist. Seega esimene ülesanne oli läbi viia analüüs. Esiteks, oli vajalik kaardistada, millised teadmised on üliõpilastel juba olemas objektorienteeritud programmeerimise kohta. Kursuse esimese 12. nädala jooksul on üliõpilastel objektorienteeritud programmeerimisega kokkupuude puudulik. Informaatika õppekava esimese semestri jooksul ei tutvusta ka ükski teine aine objektorienteeritud programmeerimise aluseid. Sellest lähtuvalt oli tarvilik koostada sellised õpiku peatükid ja ülesanded, mis tutvustaksid objektorienteeritud programmeerimist täiesti algusest.

Üks õppematerjalide loomise eesmärkidest oli tagada sujuvam sissejuhatus kursusesse "Objektorienteeritud programmeerimine". Sellest lähtuvalt analüüsis töö autor eelmainitud kursuse õppetööd esimeste nädalate jooksul. Esimeste nädalate jooksul tutvustatakse üliõpilastele objektorienteeritud programmeerimise põhilisi kontseptsioone nagu objekt, klass, klassimuutujad, klassimeetodid, konstruktor, pärilikkus ja polümorfism (vt tabel 3). Eesmärgi saavutamiseks otsustas töö autor, et esimesed kaks õpiku peatükki peavad sisaldama neid teemasid.

Tabel 3. Kursuse "Objektorienteeritud programmeerimine" teemade jaotus esimese nelja nädala jooksul.

Nädal	Teemad
1	Java põhikonstruktsioonid.
2	Objektid. Klassid (konstruktor, klassimuutujad, klassimeetodid).
3	Sõned. Failid. Listid.
4	Polümorfism. Pärilikkus. Liidesed.

Analüüsi etapi jooksul oli tarvilik analüüsida seni kasutatud õpiku peatükke kursusel “Programmeerimine”. Praeguseni kasutati kursusel teemade tutvustamiseks Aivar Annamaa programmeerimise õpikut [21]. Õpiku peatükkide koostamisel otsustas töö autor jälgida sarnast mustrit Aivar Annamaa loodud peatükkidega: sissejuhatus peatükki, uue teema tutvustus ja näited. Taoline peatükkide ülesehitus toetab ka Merrilli mudelis sõnastatud õppestrateegiaid.

Lõpetuseks oli vajalik töö autoril selgelt välja tuua õpiväljundid, mida peaks materjale omandanud tudeng saavutama. Õpiväljundite kaardistamine on oluline, sest sellest sõltub kavandamise etapp. Õpiväljundid sõnastati nii, et materjalid läbi töötanud tudeng:

- mõistab, mis on objekt ja kuidas seda luua;
- oskab luua klasse ja defineerida erinevaid isendimuutujaid;
- oskab luua erinevaid klassiisendeid ja kasutada isendimeetodeid;
- saab aru, millised seosed on alam- ja ülemklasside vahel;
- oskab kasutada nii pärilikkust kui ka polümorfismi.

5.2 Kavandamine

Peale analüüsi läbiviimist alustas töö autor kavandamise etapiga. Analüüsis kogutud informatsiooni põhjal koostati plaan, mille põhjal õppematerjale koostama hakata. Esiteks pidi otsustama, millist keskkonda kasutatakse õpiku peatükkide esitamiseks. Juhendajatega nõu pidades leidsid kursuse “Programmeerimine” eest vastutavad õppejõud ja töö autor, et parim lahendus oleks koondada õpiku peatükid courses.cs.ut.ee õppekeskkonda. Sellisel moel on nii õpiku peatükid kui ka ülesanded ühes keskkonnas. Hetkel peavad tudengid kursust “Programmeerimine” läbides kasutama kolme erinevat õppekeskkonda: courses.cs.ut.ee, Moodle ja Aivar Annamaa õpiku keskkonda. Õppeaine üks tuleviku eesmärk oleks koondada kõik ülesanded ja peatükid Courses keskkonda. Seda eesmärki silmas pidades otsustati koondada uued õpiku peatükid ja ülesanded eelmainitud keskkonda.

Plaani kuulus ka üldisema õpiku struktuuri planeerimine. Oluline oli otsustada loogiline teemajaotus kahe õpiku peatüki vahel. Töö autor jaotas teemad nii, et esimese nädala jooksul tutvutakse objekti, klassi, konstruktori, muutujate ja meetodiga (vt tabel 3). Teise nädala jooksul läbitakse alam- ja ülemklasside, polümorfismi ja pärilikkuse teemad.

Tabel 3. Loodud objektorienteeritud õppematerjalide teemajaotus.

Nädal	Teemad
1	Objekt. Klass. Isendimuutujad ja meetodid. Konstruktor. Parameeter self.
2	Alam- ja ülemklassid. Pärilikkus. Polümorfism. Meeetod __str__.

Lisaks otsustas autor, et iga uue teema tutvustust saadab koodipõhine näide ning õpiku peatükis on ka üks kontrollküsimus, millele peab tudeng olema võimeline iseseisvalt vastama. Antud otsus on kooskõlas Merrilli õppestrateegiatega ehk selgitus, näide, meenutamine ja kontrollimine.

Lõpetuseks oli vaja selgeks teha, kui palju ülesandeid on vaja koostada. Analüüsides ülesannete mahtu kursusel "Programmeerimine" leidis autor, et koduülesandeid peaks mahu poolest olema vähem kui praktikumi ülesandeid. Seda seetõttu, et praktikumi jooksul peab ka kõige usinamatele tudengitele jaguma ülesandeid, mida lahendada pooleteise tunni jooksul. Lisaks leiti, et praktikumi ülesannete puhul on võimalik harjutuste taset tõsta vastavalt üliõpilase oskustele. See tähendab, et igal ülesandel on olemas baaslahendus, milleni peaks jõudma iga üliõpilane. Baaslahendusele järgneb esimene lisaülesanne, mille korral peab lahendust täiendama oskuslikuma meetodiga. Teise lisaülesande puhul peab olema tudeng võimeline täiendama baaslahendust veelgi oskuslikuma meetodiga, mis ei kuulu üliõpilaste õpiväljunditesse. Sel moel tagavad õppematerjalid, et ka erineva õppevõimekusega tudengid on kaasatud õppetöösse.

5.3 Arendamine

Arenduse etapi käigus toimus õppematerjalide loomine eelmises etapis koostatud plaani järgi. Selle etapi käigus valmisid mõlemad õpiku peatükid, praktikumiülesanded ja koduülesanded. Töö autor alustas materjalide arendamist 2021. aasta augustis ja kokku kestis arendus üheksa kuud ehk materjalid valmisid lõplikult 2022. aasta aprilliks. Materjalide loomine oli jagatud etappideks. Tööjaotus oli ehitatud üles Merrilli õppestrateegiate alusel. See tähendab, et etappideks olid teooria kirjutamine (selgitus), koodipõhiste näidete lisamine (näide), kontrollküsimuste integreerimine (meenutamine) ning praktikumi- ja koduülesannete loomine (rakendamine). Etappe läbiti vastupidises järjekorras ehk alustati ülesannete loomisest, siis

liiguti kontrollküsimuste juurde ja lõpetuseks loodi koodipõhine näide ning tutvustati uue teema teoreetilist poolt.

5.4 Materjalide loomine

Õppematerjalide loomisel tugineti Merrilli õppestrateegiatele vastupidises järjekorras. Seega algas materjalide loomine ülesannetest. Analüüsi ja kavandamise etapi käigus loodud plaani põhjal oli teada, mis teemasid peaksid ülesanded käsitlema. Ülesannete koostamise protsess algas vastavate teemade analüüsist. Kõigepealt tutvuti teemaga ning kaardistati elemendid, mida peaksid ülesanded sisaldama. Lisaks tutvus autor lähemalt kursusega "Objektorienteeritud programmeerimine", et saada parem ülevaade sellest, kuidas lähenetakse objektorienteeritud programmeerimise ülesannetele. Peale vajalike teadmiste saamist, liiguti ülesannete loomise juurde. Ülesannete koostamisel järgiti järgnevaid printsiipe:

- Ülesande idee peab olema originaalne. See aitab üliõpilast paremini kaasata õppetööse ning annab värskust kursuse "Programmeerimine" õppematerjalidele.
- Ülesande sõnastus peab olema selge ja arusaadav. See aitab vältida nii üliõpilastel kui ka õppejõududel tekkivaid arusaamatusi.
- Ülesandele loodud näidislahendus peaks järgima parimaid objektorienteeritud programmeerimise tavasid. Näidislahendus on paljude üliõpilaste jaoks kõrge väärtusega just tagasiside saamiseks.

6. Tulemused

Viimases peatükis tutvustatakse loodud materjale. Kirjeldatakse ka protsessi, kuidas loodi õppematerjale, alustades ülesandest ja lõpetades õpiku peatükiga.

6.1 Õppematerjalide koostamise protsess

Kokku valmis kaks õpiku peatükki ja 12 ülesannet, millest kuus olid esimese nädala ja viis teise nädala jaoks ning üks koduülesanne järgmiste aastate jaoks (Lisa 1). Ülesannete koostamisel lähtus autor metoodika peatükis tutvustatud kavandamise plaanist, mille kohaselt on praktikumiülesanded mahukamad kui koduülesanded. Autor toob välja ka näite, kuidas Merrilli mudelis sõnastatud õppestrateegiate vastupidise järjekorra alusel loodi ülesanne, kontrollküsimus, koodipõhine näide ja teoreetiline teemalõik.

Näitena esitatakse esimesel nädalal läbitud teemat isendimeetod:

1. Rakendamine (vt joonis 1) - isendimeetoditest põhjalikuks arusaamiseks koostas autor koduülesande, kus on vaja luua kolm erinevat isendimeetodit. Selleks oli vaja luua klassi *Raamatukogu* jaoks erinevaid meetodeid, mis lisavad klassile kogu funktsionaalsuse. Neid meetodeid luues saavutavad üliõpilased parema arusaamise isendimeetodite kohta. Lisaks on tudengitele esitatud näidisväljund, mis aitab neil paremini mõista, milline peaks ülesande lahendus välja nägema.

1. Raamatukogu

Koosta klass nimega `Raamat`, mille konstruktori parameetriteks on raamatu pealkiri, raamatu autor, raamatu lehekülgede arv ning raamatu liik (romaan, muinasjutt või detektiiv). Klassis peab ka olema meetod *kuvaInfo*, mis väljastab raamatu kohta kogu info loetaval kujul.

Koosta klass nimega `Raamatukogu`, mille konstruktori parameetriks on tühi järjend. Klassis peavad olema järgmised meetodid:

- Meetod *lisaRaamat*, mis võtab argumendiks klassi `Raamat` isendi ja lisab selle klassi `Raamatukogu` järjendisse.
- Meetod *kuvaRaamatud*, mis kasutades meetodit *kuvaInfo* kuvab kõikide raamatukogu raamatute info.
- Meetod *laenutaRaamat*, mille väljakutsel kuvatakse ekraanile kõik raamatukogus olevad raamatud (meetodi *raamatuteInfo* abil) ning küsitakse kasutajalt, millist raamatut ta laenutada soovib. Juhul, kui kasutaja sisestas olemasoleva raamatu pealkirja korrektselt (ei pea jälgima korrektset suur- ja väiksetähelisust), eemaldada raamat järjendist ja kuvada sobilik sõnum. Vastasel juhul peab meetod uuesti küsima, millist raamatut soovitakse laenutada, kuni kastuaja sisestab korrektselt mõne olemasoelva raamatu pealkirja.

On olemas fail nimega *raamatud.txt*:

```
Tõde ja õigus I, A.H.Tammsaare, 453, romaan
Sherlock Holmesi seiklused, O.C.Doyle, 233, detektiiv
Läänerindel muutuseta, E.M.Remarque, 310, romaan
Punamütsike, Charles Perrault, 17, muinasjutt
Eesriie, Agatha Christie, 98, detektiiv
Suur Gatsby, F. Scott Fitzgerald, 188, romaan
```

Koosta programm, mis lisab kõik failis olevad raamatud klassi `Raamatukogu` isendisse, laenutab raamatu "Eesriie" ning väljastab kõik raamatukogus olevad raamatud peale raamatu laenutust.

Näide:

```
>>> raamatukogu.laenutaRaamat()
>>> raamatukogu.kuvaRaamatud()
```

Väljund:

```
Raamatukogus olevad raamatud:
Tõde ja Õigus I, A.H.Tammsaare, 453, romaan
Sherlock Holmesi seiklused, O.C.Doyle, 233, detektiiv
Läänerindel Muutuseta, E.M.Remarque, 310, romaan
Punamütsike, Charles Perrault, 17, muinasjutt
Eesriie, Agatha Christie, 98, detektiiv
Suur Gatsby, F. Scott Fitzgerald, 188, romaan

Sisesta raamatu nimi, mida sa laenutada soovid: esi
Ei leidnud sellist raamatut nimekirjast, proovi uuesti
Sisesta raamatu nimi, mida sa laenutada soovid: eesRiie
Raamat "Eesriie" edukalt laenutatud

Tõde ja Õigus I, A.H.Tammsaare, 453, romaan
Sherlock Holmesi seiklused, O.C.Doyle, 233, detektiiv
Läänerindel Muutuseta, E.M.Remarque, 310, romaan
Punamütsike, Charles Perrault, 17, muinasjutt
Suur Gatsby, F. Scott Fitzgerald, 188, romaan
```

Joonis 1. Esimese nädala koduülesanne, mis kontrollib erinevate isendimeetodite loomist.

2. Meenutamine (vt joonis 2) - üliõpilase teadmiste kontrollimiseks isendimeetodite kohta, lisas töö autor õpiku peatükki kontrollküsimuse. Kontrollküsimuse abil saab üliõpilane kiiret tagasisidet selle kohta, kui hästi ta läbitud teema omandas. Iga vastusevariant sisaldab ka tagasisidet, mis selgitab, miks see variant ei sobi vastuseks (vt joonis 3).

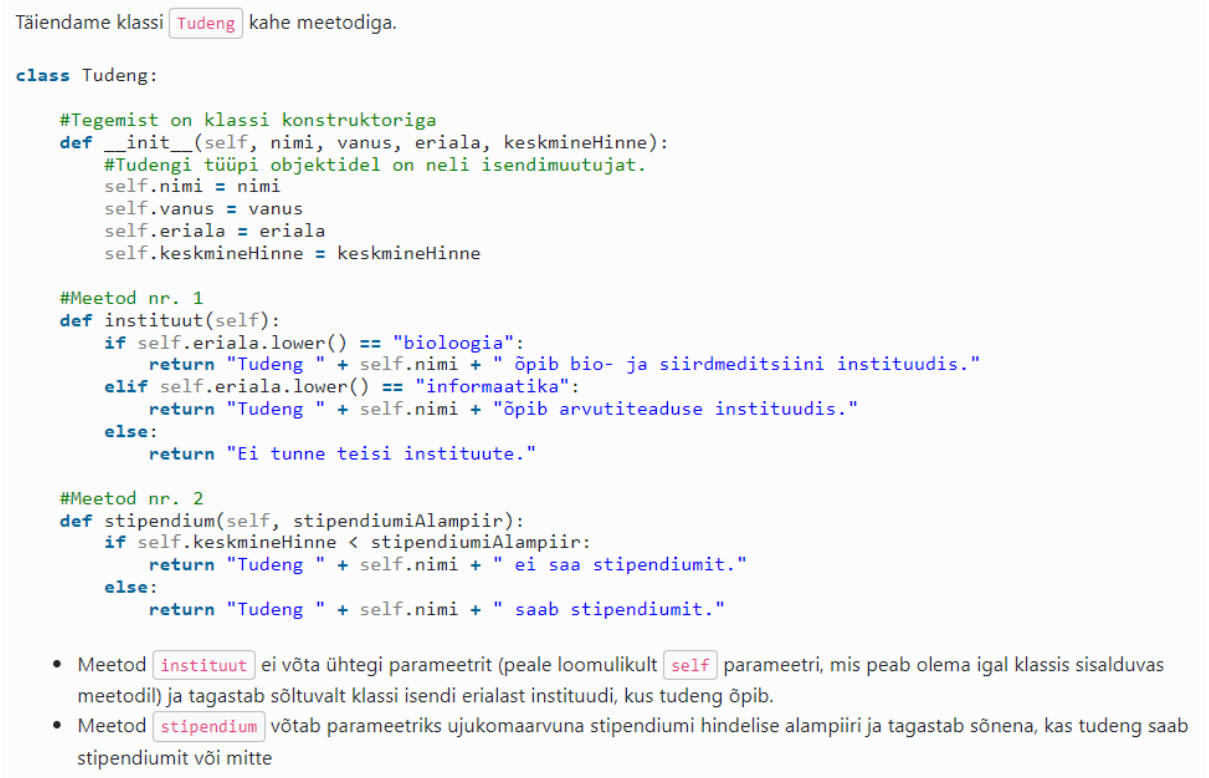
Mis väljastatakse ekraanile?

```
class Kass:
  def __init__(self, nimi, vanus, lemmiksöök):
    self.nimi = nimi
    self.vanus = vanus
    self.lemmiksöök = lemmiksöök

  def väljastaInfo():
    print(self.nimi, self.vanus, self.lemmiksöök)

kass = Kass("Bruuno", 10, "kohupiim")

kass.väljastaInfo()
```

Joonis 2. Õpiku esimeses peatükis olev kontrollküsimus isendimeetodi kohta.

Joonis 3. Esimeses peatükis oleva kontrollküsimuse vastusevariantide tagasiside.

3. Näide (vt joonis 4) - selleks, et üliõpilasele demonstreerida, kuidas isendimeetodeid Pythonis luua, lisas autor õpiku peatükki ka koodipõhise näite. Lisatud näited aitavad üliõpilastel kinnistada loetud teooriat ka läbi praktiliste teadmiste.



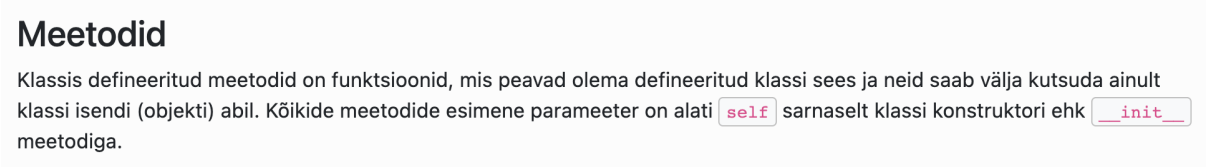
Täiendame klassi `Tudeng` kahe meetodiga.

```
class Tudeng:

    #Tegemist on klassi konstruktoriga
    def __init__(self, nimi, vanus, eriala, keskmineHinne):
        #Tudengi tüüpi objektidel on neli isendimuutujat.
        self.nimi = nimi
        self.vanus = vanus
        self.eriala = eriala
        self.keskmineHinne = keskmineHinne

    #Meetod nr. 1
    def instituut(self):
        if self.eriala.lower() == "bioloogia":
            return "Tudeng " + self.nimi + " õpib bio- ja siirdmeditsiini instituudis."
        elif self.eriala.lower() == "informaatika":
            return "Tudeng " + self.nimi + "õpib arvutiteaduse instituudis."
        else:
            return "Ei tunne teisi instituute."

    #Meetod nr. 2
    def stipendium(self, stipendiumiAlampiir):
        if self.keskmineHinne < stipendiumiAlampiir:
            return "Tudeng " + self.nimi + " ei saa stipendiumit."
        else:
            return "Tudeng " + self.nimi + " saab stipendiumit."
```

- Meetod `instituut` ei võta ühtegi parameetrit (peale loomulikult `self` parameetri, mis peab olema igal klassis sisalduvas meetodil) ja tagastab sõltuvalt klassi isendi erialast instituudi, kus tudeng õpib.
- Meetod `stipendium` võtab parameetriks ujukomaarvuna stipendiumi hindelise alampiiri ja tagastab sõnena, kas tudeng saab stipendiumit või mitte

Joonis 4. Koodipõhine näide kahest erinevast isendimeetodist.

4. Selgitus (vt joonis 5) - isendimeetodeid tutvustatakse üliõpilasele koos sobivate definitsioonidega. Täpsustatakse, kuidas saab kasutada isendimeetodeid ja milleks need vajalikud on.



Meetodid

Klassis defineeritud meetodid on funktsioonid, mis peavad olema defineeritud klassi sees ja neid saab välja kutsuda ainult klassi isendi (objekti) abil. Kõikide meetodide esimene parameeter on alati `self` sarnaselt klassi konstruktori ehk `__init__` meetodiga.

Joonis 5. Õpiku peatüki esimene osa, kus tutvustatakse isendimeetodeid.

Sarnast mustrit järgides loodi ka teisi teemasid käsitlevad ülesanded, kontrollküsimused, koodipõhised näited ja uut teemat tutvustavad lõigud.

6.2 Õppematerjalide tulevik ja edasiarendused

Kursusel “Programmeerimine” hakatakse autori poolt loodud objektorienteeritud õppematerjale kasutama alates 2022. aasta sügissemestrist. Materjalid on juba kättesaadaval courses.cs.ut.ee keskkonnas [22]. Selleks, et tudengid saaksid kiirelt tagasisidet enda lahenduse kohta, on tarvilik luua Moodle keskkonnas automaatkontrolle. Automaatkontrollid lihtsustavad ka õppejõudude tööd, sest lahenduse kohta saab üliõpilane esmase ülevaate automaatkontrolli abil. Töö autoril on plaan kirjutada valmis automaatkontrollid 2022. aasta suvel, enne kursuse algust. Lisaks on õppeaine “Programmeerimine” eest vastutavad õppejõud otsustanud, et iga nädal võiks ühe kodutöö ülesande jaoks olla murelahendaja. Murelahendajad on vihjete komplektid, mis üritavad küsimuse vormis selgitada probleeme ning pakkuda neile lahendusi [23]. Pärast seda, kui üliõpilased on uued õppematerjalid läbinud, on plaanis koguda tagasisidet. Tagasiside abil saab materjalides teha parandusi. Lisaks saab ülevaate loodud õppematerjalide kohta, näiteks ülesannete ajakulu ja raskusastme ning õpikus olevate materjalide selguse kohta. Kui kõik eelmainitud sammud saavad tehtud, siis eesmärk luua vajalik objektorienteeritud programmeerimist käsitlev õppematerjal kursuse “Programmeerimine” jaoks on saavutatud.

7. Kokkuvõte

Bakalaureusetöö tulemusena koostati kursuse “Programmeerimine” jaoks objektorienteeritud programmeerimise õppematerjale. Materjalide koostamise eesmärk oli asendada osaliselt rekursiooni teema ning anda tudengitele sujuvam sissejuhatus õppeainesse “Objektorienteeritud programmeerimine”. Kokku valmis kaks õpiku peatükki ja 12 ülesannet. Kahjuks ei jõutud koguda tagasisidet tudengite käest, mis oleks andnud parema ülevaate koostatud õppematerjalide efektiivsusest ja kasulikkusest.

Õppematerjalide loomiseks tutvus töö autor “Programmeerimine” kursuse ülesehitusega ja kasutatava õppemeetodikaga. Seejärel tehti põhjalik ülevaade objektorienteeritud programmeerimise põhitõdedest ja tähtsamatest kontseptsioonidest. Nende abil tehti selgeks, milliseid teemasid peaksid uued õppematerjalid sisaldama. Materjalide koostamise plaani loomiseks lähtuti ADDIE mudeli etappidest. Peale põhjalikku analüüsi ja kavandamist hakati looma õppematerjale. Materjalide loomisel lähtuti Merrilli mudelis defineeritud õppestrateegiatest. Autor koostas õppematerjale Merrilli õppestrateegiate järgi vastupidises järjekorras, milles alustati ülesande tegemisest ja lõpetati uue teema tutvustamisega õpikus. Ülesannetele loodi näidisväljundid ja -lahendused.

Autori loodud õppematerjale hakatakse kasutama kursusel “Programmeerimine” 2022. aasta sügissemestril. Selleks, et materjalid oleksid lõplikult valmis kursusel kasutamiseks, plaanib autor koostada ülesannetele automaatkontrollid ning murelahendajad. Pärast õppematerjalide läbimist on oluline saada üliõpilastelt tagasisidet. Selle abil on võimalik saada põhjalikum ülevaade õppematerjalidest tervikuna ning teha parandusi vastavalt vajadusele.

8. Kasutatud kirjandus

[1] Tartu Ülikooli õppeinfosüsteem. https://ois.ut.ee (06.12.2021)

[2] Kursuse „Programmeerimine“ 2021/22 õppeaasta koduleheküIg. https://courses.cs.ut.ee/2021/programmeerimine (06.12.2021)

[3] Kursuse „Programmeerimine II“ 2021/22 õppeaasta koduleheküIg. https://courses.cs.ut.ee/2022/P2/spring (06.12.2021)

[4] Tartu Ülikooli Moodle. Programmeerimine. https://moodle.ut.ee/course/view.php?id=500 (09.12.2021)

[5] C. Nwosisi, A. Ferreira, W. Rosenberg, K. Walsh. A Study of the Flipped Classroom and Its Effectiveness in Flipping Thirty Percent of the Course Content. *International Journal of Information and Education Technology*, 2016, pp. 348-351. https://www.researchgate.net/publication/276427778_A_Study_of_the_Flipped_Classroom_and_Its_Effectiveness_in_Flipping_Thirty_Percent_of_the_Course_Content

[6] S.J.M. Díaz, J.G. Antequera, M.C. Pizarro. Flipped Classroom in the Context of Higher Education: Learning, Satisfaction and Interaction. *Education Sciences, v11 Article 416*, 2021, pp. 13-14. https://web-s-ebscohost-com.ezproxy.utlib.ut.ee/ehost/detail/detail?vid=4&sid=8f6c7ed1-14cf-4cea-9c8f-17fa29f95f20%40redis&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=eric&AN=EJ1307774

[7] S. Veres, A.D. Muntean. The Flipped Classroom as an Instructional Model. *Romanian Review of Geographical Education*, 2021, pp. 63-64. https://eric.ed.gov/?q=Flipped+Classroom&id=EJ1293911

[8] D. Amos. Object-Oriented Programming (OOP) in Python 3. https://realpython.com/python3-object-oriented-programming/ (20.12.2021)

[9] D. Phillips. Python 3 Object-oriented Programming. 2015, pp. 3-6. https://library.kre.dp.ua/Books/2-4%20kurs/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%2B%20%D0%BC%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F/Python/Python_3_Object_Oriented_Programming.pdf (20.12.2021)

[10] F. Yang, J.E. Menard. PyISOLVER—A Fast Python OOP Implementation of LRDFIT Model. *IEEE Transactions on Plasma Science*, 2020, pp 1793-1796. https://ieeexplore-ieee-org.ezproxy.utlib.ut.ee/document/8941043?arnumber=8941043 (20.12.2021)

[11] I. Hadar. When intuition and logic clash: The case of the object-oriented paradigm. *Science of Computer Programming. Vol. 78 Issue 9*, 2013, pp. 1410- 1411. https://web-p-ebscohost-com.ezproxy.utlib.ut.ee/ehost/detail/detail?vid=8&sid=35a52568-c0a7-4b80-a445-1032a4abff3d%40redis&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#AN=89520728&db=a9h

[12] Q. Kong, T, Siauw, A. Bayen. Python Programming And Numerical Methods: A Guide For Engineers And Scientists. *Academid Press*, 2020. https://pythonnumericalmethods.berkeley.edu/notebooks/Index.html

[13] G. Saltsman, K. Shelton. Using the Addie Model for Teaching Online. *International Journal of Information And Communication Technology Education 2(3)*, 2006, pp 14-26.

https://www.researchgate.net/publication/220202796_Using_the_Addie_Model_for_Teaching_Online

[14] S. J. McGriff. Instructional System Design (ISD): Using the ADDIE Model. *Instructional Systems, College of Education, Penn State University*, 2000. https://www.lib.purdue.edu/sites/default/files/directory/butler38/ADDIE.pdf

[15] N. Aldoobie. ADDIE Model. *American International Journal of Contemporary Research Vol. 5, No. 6*, 2015, pp 68-71. http://www.aijcrnet.com/journals/Vol_5_No_6_December_2015/10.pdf

[16] L. Cheung. Using the ADDIE Model of Instructional Design to Teach Chest Radiograph Interpretation. *Journal of Biomedical Education*, 2016, pp 1-4. https://www.researchgate.net/publication/304187606_Using_the_ADDIE_Model_of_Instructional_Design_to_Teach_Chest_Radiograph_Interpretation

[17] M. D. Merrill. First Principles of Instruction. *Educ. Technol., Res. Dev. 50*, 2002, pp. 43–59. https://mdavidmerrill.files.wordpress.com/2019/04/firstprinciplesbymerrill.pdf

[18] M. Jalilehvand. Study the Impact of Merrill's First Principles of Instruction on Students' Creativity. *Mediterranean Journal of Social Sciences 7(2)*, 2016, pp 314.

https://www.researchgate.net/publication/297684149_Study_the_Impact_of_Merrill's_First_Principles_of_Instruction_on_Students'_Creativity

[19] J. L. Gardner. Testing the efficacy of Merrill's First Principles of Instruction in improving student performance in introductory biology courses. *Dissertation Abstracts International Section A: Humanities and Social Sciences 72(6)*, 2011, pp 100-129. https://web-p-ebscohost-com.ezproxy.utlib.ut.ee/ehost/detail/detail?vid=3&sid=35541448-0727-4ec2-bf53-c55de246cefd%40redis&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#AN=2011-99231-028&db=psyh

[20] M. D. Merrill. A Task-Centered Instructional Strategy. *Journal of Research on Technology in Education*, 2007, pp 13-19. https://files.eric.ed.gov/fulltext/EJ826059.pdf

[21] A. Annamaa. Programmeerimise õpik. https://progeopik.cs.ut.ee/index.html (06.04.2022)

[22] „Programmeerimine" õppeaine õppeaasta 2022/23 koduleheküIg. https://courses.cs.ut.ee/2021/programmeerimine (08.04.2022)

[23] J. Kaimre. Murelahendajate koostamine Tartu Ülikooli kursuse „Programmeerimine" jaoks. *TÜ arvutiteaduseinstituudi bakalaureusetöö*, 2021. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=71614&year=2021 (08.04.2022)

Lisad

I Loodud õppematerjalid

Alljärgnevalt on lingid loodud õpiku peatükkidele ja ülesannetele:

1. nädala õpiku peatükk: https://courses.cs.ut.ee/2022/programmeerimine/Main/Oop

1. nädala praktikumiülesanded:
https://courses.cs.ut.ee/2022/programmeerimine/fall/Main/Praks13

1. nädala koduülesanded: https://courses.cs.ut.ee/2022/programmeerimine/fall/Main/Kodu13

2. nädala õpiku peatükk: https://courses.cs.ut.ee/2022/programmeerimine/Main/Oop2

2. nädala praktikumiülesanded:
https://courses.cs.ut.ee/2022/programmeerimine/fall/Main/Praks14

2. nädala koduülesanded: https://courses.cs.ut.ee/2022/programmeerimine/fall/Main/Kodu14

II Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Georg Demidov,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Tartu Ülikooli kursuse "Programmeerimine" jaoks objektorienteeritud õppematerjalide loomine**, mille juhendajateks on Helle Hein ja Tauno Palts, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, alates 10.05.2022 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.

Georg Demidov
10.05.2022