

Lappeenrannan–Lahden teknillinen yliopisto LUT
Software Engineering

Eetu Asikainen

**OHJELMOINTIOPPAAN LUONTI OLIO-OHJELMOINNIN
KURSSILLE**

Kandidaatintyö

Tarkastaja: Yliopisto-opettaja (TkT) Erno Vanhala

Ohjaaja: Yliopisto-opettaja (TkT) Erno Vanhala

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikka

Eetu Asikainen

Ohjelmointioppaan luonti olio-ohjelmoinnin kurssille

Kandidaatintyö

2022

40 sivua, 10 kuvaa, 8 kaaviota.

Tarkastaja: Yliopisto-opettaja (TkT) Erno Vanhala

Hakusanat: java, latex, olio-ohjelmointi

Keywords: java, latex, object oriented programming

Tässä työssä suunniteltiin ja toteutettiin uudistettu ohjelmointiopas LUT-yliopiston Olio-ohjelmointi -kurssille. Työssä vertailtiin kolmea teknistä toteutustapaa oppaan kirjoitusprosessiin. Vertailutavat olivat Jyväskylän yliopiston TIM-opetusympäristö, LaTeX-taittokieli ja tekstinkäsittelyohjelma kuten Microsoft Word. Osana työtä arvioitiin myös oppaan sisältötarvetta, muun muassa kielivalinnan ja Android-API:n käsittelyn suhteen.

Työn avuksi toteutettiin kysely keväällä 2021 kurssilaisille. Kyselyn tuloksia käytettiin apuna oppaan suunnittelussa niin teknisen toteutuksen valinnan kuin sisältötarpeen arvioinnin suhteen.

Työssä todettiin LaTeX-taittokielen olevan sopivin valinta oppaan toteutukseen. Tähän vaikuttivat kielen tarjoama mahdollisuus koodin esittämiseen osana opasta helposti ja kielen tuottaman pdf-dokumentin käyttämisen vaivattomuus opiskelijan näkökulmasta. Oppaan kieleksi valittiin suomi opiskelijapalautteen perusteella.

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Software Engineering

Eetu Asikainen

Creating a programming guide for an Object Oriented Programming course

Bachelor's Thesis

2022

40 pages, 10 figures, 8 tables.

Examiner: Erno Vanhala D.Sc. (Tech.)

Keywords: java, latex, object oriented programming

This study was about planning and executing a new programming guide for the LUT-university "Object oriented programming" course. The study compared the different technical solutions for writing the guide. The solutions were the TIM course environment of University of Jyväskylä, the LaTeX typesetting language and text editing programs like Microsoft Word. As a part of the study the planned content of the guide was also examined regarding for example the choice of language and whether to include Android API related chapter(s).

As a guideline for the planning process a survey was held for the participants of the spring 2021 "Object oriented programming" course. The results of the survey were used in both deciding the technical solution for the guide as well as planning the contents of the guide.

The study concluded LaTeX to be the most fitting technical solution for the guide. The main reasons for this were the effortless code displaying and highlighting it provided and the simplicity and usability of the pdf-document the language compiles into. Finnish was chosen as the language of the guide based on the survey results.

SISÄLTÖ

1	Johdanto	6
1.1	Työn tausta	6
1.2	Työn tavoitteet	6
1.3	Työn rakenne	7
2	Kirjallisuuskatsaus	8
3	Tutkimusprosessi	11
3.1	Tutkimusongelman rajaaminen	11
3.2	Yleiskatsaus aihealueesta	12
3.3	Oppaan kirjoittaminen	12
3.4	Lopputuloksen arviointi	13
4	Oppaan tekninen toteutus	15
4.1	The Interactive Material -alusta	15
4.2	LaTeX-taittokieli	16
4.3	Tekstinkäsittelyohjelmat	18
4.4	Päätös toteutustavasta	19
4.5	Käytetyt ohjelmistot	20
4.6	Versionhallinnan käyttäminen LaTeX-dokumentin yhteydessä	21
4.7	Koodin esittäminen oppaassa	22
4.8	LaTeXin käyttö oppaassa	24
5	Oppaan sisällön suunnittelu	25
5.1	Oppaan yleinen rakenne	25
5.2	Sanastot	25
5.3	Android-API:n käsittely	27
5.4	Kurssitavoitteiden ulkopuolisen materiaalin käsittely	28
6	Oppaan sisällön esittely	29
6.1	Oppaan alkumateriaali ja luvut 0 ja 1	29
6.2	Javan perusteet: oppaan luvut 2, 3 ja 4	30
6.3	Perehtyminen periytymiseen ja oliopohjaiseen suunnitteluun: oppaan luvut 5 ja 6	30
6.4	Suunnittelumallien maailman oven raottamista: oppaan luvut 8 ja 9	31
6.5	Lopun kurssin aihealueen käsittely: oppaan luvut 10 ja 11, sekä sanastot	32
7	Pohdintaa	33

	5
7.1 Oppaan jatkokehitys	33
7.2 GitHub issues	33
7.3 Arvio työn onnistumisesta	34
7.4 Jatkotutkimus	34
8 Yhteenveto työstä	36
LÄHTEET	37

1 Johdanto

Vaikka Lappeenrannan-Lahden teknillisessä yliopistossa (tästä lähtien LUT-yliopisto) opiskelijat saavat ohjelmoinnin opiskeluun vankan pohjan ohjelmoinnin perusteet -kurssilla, pidetään sitä seuraavaa olio-ohjelmoinnin kurssia haastavana opiskelijoiden keskuudessa. Osasyyn tähän on kurssin vanhentunut oppimateriaali. Tämän kandidaatintyön on tarkoitus auttaa tilannetta, tuottamalla kurssille uusi ohjelmointiopas opetusmateriaaliksi.

1.1 Työn tausta

LUT-yliopiston olio-ohjelmoinnin kurssi on kokenut suuria muutoksia viimeisten vuosien aikana, samalla kun kurssin käyttämä Java-kieli on kehittynyt ja modernisoitunut. Nämä vaikutukset yhdessä ovat tehneet kurssin nykyisestä ohjelmintiooppaasta vanhentuneen. Kurssin sisältö on muuttunut, mutta suurin muutos kurssin opiskelijoille on kurssin siirtyminen suositellussa opiskelujärjestyksessä toisen vuoden kurssista ensimmäisen vuoden kurssiksi. Olio-ohjelmoinnin opettaminen opintojen alkupuolella on yleistymässä, joten päätös on looginen [Kölling, 1999a]. Kuitenkin se luo myös omia ongelmiaan.

Ensimmäisen vuoden opiskelijat eivät ole vielä tottuneet omatoimiseen tiedonhakuun, joten yliopiston olisi järkevää tarjota itse laadukas tapa opiskella kurssin aihealue. Yksi ratkaisu ongelmaan on vanhan ohjelmointiooppaan uusiminen. Tämä tarjoaa opiskelijoille modernimman tietolähteen osana kurssin oppimateriaaleja.

1.2 Työn tavoitteet

Työn tärkeimpänä tavoitteena on tuottaa uusi ohjelmointiopas LUT-yliopiston olio-ohjelmoinnin kurssille. Tavoitteena on tuottaa laajennettava ja helppokäyttöinen opas, joka hyödyttää opiskelijoita opiskelijan taitotasosta ja ohjelmointikokemuksen määrästä riippumatta. Opas panostaa suureen koodiesimerkkien määrään tarjotakseen opiskelijoille mahdollisimman paljon käytännön esimerkkejä opiskelun tueksi. Tämä päätös on tehty, koska opiskelijat suosivat esimerkkejä muihin opettamisen muotoihin ja oppivat tutkimuksen mukaan valmiin esimerkin tutkimisesta jopa enemmän kuin saman ongelman ratkaisemisesta itsenäisesti [Caspersen and Bennedsen, 2007].

1.3 Työn rakenne

Tätä johdantolukua seuraa lyhyt katsaus kirjallisuuteen, jota on käytetty teorialähteenä työn kirjoittamisprosessissa. Kolmas luku käy läpi työn tekemiseen käytetyt tutkimusmetodit. Neljäs luku käy läpi oppaan teknisessä toteutuksessa huomioitavia seikkoja ja tarvittavia päätöksiä. Viides luku kertoo tuotettavan ohjelmointioppaan sisällön suunnittelusta ja toteutuksesta kun taas kuudes luku esittelee oppaan pääpiirteittäin. Seitsemäs ja kahdeksas luku sisältävät pohdintaa ja yhteenvedon työstä.

2 Kirjallisuuskatsaus

Työn toteutus pedagogisena resurssina tarkoittaa, että työn teknisen osuuden onnistuneen suunnittelun pohjana on järkevää käyttää tutkimuksia ohjelmoinnin ja erityisesti oliopohjaisen ohjelmoinnin opettamisesta. Esimerkiksi Michael E. Kaspersenin ja Jens Bennedsenin artikkeli *Instructional design of a programming course: A learning theoretic approach* [Caspersen and Bennedsen, 2007] sisältää paljon hyödyllistä tietoa oppaan kirjoittamisen suhteen. Artikkelin käsittelee ohjelmointikurssin rakentamista oppimisteorian näkökulmasta ja korostaa erityisesti monipuolisten esimerkkien tärkeyttä. Artikkelin mukaan oppimisen kannalta parhaat esimerkit antavat opiskelijoille puolivalmiin pohjan, joka sisältää vihjeitä valmiista ratkaisusta antaen näin opiskelijoiden itse löytää uusia konsepteja opiskeltavasta kielestä.

Näitä pedagogisia lähteitä on etsitty Google Scholar -palvelusta ja LUT-yliopiston tiedekirjaston tarjoamasta Ex-libris -palvelusta. Hakusanoina on käytetty variaatioita fraaseista ”teaching object oriented programming”, ”teaching programming” ja ”object oriented programming programming guide”. Lisäksi lähteitä on haettu muiden LUT-yliopiston ohjelmointikurssien uudistuksista tehtyjen tutkimusten joukosta.

Yksi tärkeä lähdejoukko työlle pedagogisen teorian alalla löytyi Michael Köllingin tuotannosta. Kölling on tuottanut laajan joukon tutkimuksia ja artikkeleita, jotka käsittelevät oliopohjaisen ohjelmoinnin opettamista esimerkiksi ohjelmointikielen valinnan suhteen [Kölling, 1999a] ja käytettävän ohjelmointiympäristön suhteen [Kölling, 1999b]. Ohjelmointikielen valintaa koskeva artikkeli nostaa esille Javan huonoja puolia oliopohjaisen ohjelmoinnin opetuskielenä, näiden huomiointi on tärkeää opasta kirjoittaessa, koska kurssin kielen vaihtaminen ei tule kysymykseen. Artikkelin perusteella erityisesti mainfunktion ongelmallinen monimutkaisuus suhteessa sen aikaiseen esittelytarpeeseen kursilla on huomioitava oppaan sisällön suunnittelussa. Ohjelmointiympäristön valintaa koskeva artikkeli korostaa Kaspersenin ja Bennedsenin artikkelin tapaan opiskelijan tekemisiin välittömästi reagoivan ympäristön tarvetta.

LUT-yliopiston kurssien kehittämistä koskevien tutkimusten ja hakusanoilla löydettyjen tutkimusten kesken esiintyi päällekkäisyyttä Oliio-ohjelmointi -kurssin uudistusta koskevien tutkimusten suhteen. Näitä tutkimuksia löytyi kaksi kappaletta. Tutkimuksista ensimmäinen keskittyi kurssin uudistukseen ja ohjelmointikielen vaihtoon C++:sta Javaan. Siinä esiteltiin toteutettavan oppaan kannalta tärkeät kurssin tavoitteet. Näistä erityisesti oliot-ensin lähestymistapa ja koodin lukemisen painottaminen osana oppimista ovat huomioitavia oppaan rakennetta suunniteltaessa [Herala et al., 2015]. Toinen tutkimus käsitteli

paisi olio-ohjelmoinnin kurssin uudistamista, myös kahden muun LUT-yliopiston kurssin uudistusta. Artikkelin nostaa esiin kurssin uudistuksen kohtaamia ongelmia, JavaFX:n ja kurssitehtävien kanssa [Vanhala and Kasurinen, 2018].

Muut LUT-yliopiston kurssi uudistuksia koskevat tutkimukset, joita työn yhteydessä tutkittiin, käsittelivät joko ”Ohjelmoinnin perusteet-kurssin uudistamista, tai useampaa samaan aikaan uudistettua kurssia. ”Ohjelmoinnin perusteet-kurssia käsittelevistä lähteistä ensimmäinen keskittyi lähinnä kurssirakenteeseen, mutta perusteli myös hieman kurssin ohjelmointikieleksi valittua Pythonia suhteessa Javaan ja C++:aan. Tämän suhteen nostettiin esille standardoidun ohjelmointiympäristön tärkeys ja kaarisulkusyntaksin hyvät ja huonot puolet [Kasurinen and Nikula, 2007a]. Toinen tämän kurssin uudistusta käsittelevistä tutkimuksista painottaa suomenkielisen ohjelmointioppaan hyötyä kurssille ja oppaan jatkokkehityksen tärkeyttä oppaan pitämiseksi ajan tasaisena [Kasurinen and Nikula, 2007b]. Viimeinen LUT-yliopiston kurssista käsittelevä lähde, jota työn osana tutkittiin käsitteli ”Olio-ohjelmointi” ja ”Web Applications-kurssien muuttamista flipped classroom -tyyliseksi. Tutkimus nostaa esille ohjelmointioppaan tärkeyden osana kurssimateriaalia [Knutas et al., 2016].

Muita hyödyllisiä lähteitä työn pedagogiseen teoriaan löytyi muutamia ohjelmoinnin perusteiden opettamista yleistasolla käsitteleviä tutkimuksia. Koulourin, Laurian ja Macredien artikkeli ohjelmoinnin perusteiden opettamisesta käsittelee kielivalintaa, palautteen vaikutusta opiskeluun ja ongelmanratkaisutaitojen kehittämistä osana ohjelmoinnin perusteiden opettamista. Näiden lisäksi se sisältää kritiikkiä aihealueen tutkimusten arviointimenetelmistä. Esimerkiksi kurssitulosten käyttämistä tutkimustulosten arviointiin pidetään tutkimuksen mukaan arveluttavana [Koulouri et al., 2014]. Toinen laaja-alainen monikansallinen tutkimus puolsi ongelmanratkaisutaitojen kehittämisen tärkeyttä osana ohjelmointikursseja: tutkimuksen tulosten mukaan ongelmanratkaisutaidot vaikuttivat ohjelmointiosaamiseen enemmän kuin käytetty ohjelmointikieli [McCracken et al., 2001].

Näiden lisäksi yksi speisifimpi yhden ala-alueen opettamiseen keskittyvä tutkimus löytyi this-avainsanan käytöstä. Tutkimuksen mukaan this-avainsana on erityisen useasti väärinymmärretty osa olio-ohjelmointia. Tutkimus jakoi erilaiset väärin ymmärtämisen tyypit yhdeksään eri kategoriaan, mutta yleisin ongelma oli etteivät opiskelijat ymmärtäneet this-avainsanan viittaavan olio-olioon [Shmallo and Ragonis, 2021].

Suuri osa löydetyistä olio-ohjelmoinnin opettamista käsittelevistä lähteistä käsitteli opettamista pelien tai tiettyjen opetusympäristöjen tai IDE-teknologioiden avulla. Näitä tutkimuksia ei hyödynnetty lähteinä niiden kurssin opetusteknologiasta poikkeavan luonteen vuoksi.

Tämäntyyppisten akateemisten lähteiden lisäksi oppaan suunnittelussa käytetään apuna LUT-yliopiston Olio-ohjelmointi -kurssin opiskelijoille suunnattua kyselyä kurssin nykyisestä ohjelmointioppaasta. Näin pyritään huomioimaan kurssin nykyisen toteutuksen heikkoudet ja vahvuudet, sekä opiskelijoiden omat toiveet uudistuvan oppaan toteutuksesta. Koska työn tavoitteena on kirjoittaa ohjelmointiopus käyttäen valmiiksi dokumentoitua kieltä, toimii toteutuksen pääasiallisena lähteenä Javan standardikirjaston dokumentaatio. Javan dokumentaatio tarjoaa virallisen lähteen syntaksille esimerkkikoodissa avainsanoista ja rakennemuotoilusta tarjottujen luokkien ja metodien signatuureihin ja rajapintoihin [jav, 2022]. Tämän lisäksi toinen pääasiallinen lähde teknillisessä toteutuksessa on kursseille määritelty kurssikirja, Bruce Eckelin kirjoittama ”Thinking in Java”. Kirjaa käytetään lähinnä oppaan rakenteen hienosäätämässä [Eckel, 2003]. Opas ei kuitenkaan itsessään tule sisältämään lähdeviitauksia luettavuuden parantamiseksi.

Lisäksi oppaan tavoitteena on edistää laadukkaan koodin tuottamisen filosofiaa. Konseptit kuten laadukas ja testattava arkkitehtuuri, joka saavutetaan esimerkiksi käänteisten riippuvuuksien periaatteella ja ehdoton luettavuuden painottaminen ovat edistyneempiä kuin oppaan tavoitetaso. Tästä syystä luettavuutta ja koodin laadun tarkkailua näin tarkasti ei tulla painottamaan, mutta kun mahdollista opas tulee ohjaamaan opiskelijoita laadukkaan ja helppolukuisen koodin kirjoittamista edistäviä ajatustapoja kohti. Näistä filosofioista, yksinkertaisuudesta ja siististä koodista löytyy paljon ohjenuoria ja tietoa Robert C. Martinin kirjoista ”Clean Code: A Handbook of Agile Software Craftsmanship” [Martin, 2009] ja ”Agile Software Development, Principles, Patterns and Practices” [Martin and Martin, 2006], sekä Corey Hainesin julkaisusta ”Understanding the Four Rules of Simple Design” [Haines, 2014].

Työn osana tutkitut lähteet jakautuvat näin kahteen tyyppiin. Tutkimukset tai artikkelit ohjelmoinnin opettamisesta ohjaavat toteutettavan oppaan suunnittelupäätöksiä korkealla tasolla. Näitä lähteitä on etsitty lähinnä Google Scholar palvelun ja LUT-yliopiston tiedekirjaston tarjoaman ex libris palvelun kautta. Käyttöopasmaisemmat lähteet kuten Javan Standardikirjasto ja ohjelmointifilosofiaan pureutuvat kirjat kuten Clean Code ja Understanding the Four Rules of Simple Design puolestaan ohjaavat oppaan asiasisältöä. Näitä lähteitä ei ole haettu erikseen työn toteutusvaiheessa, vaan ne ovat muodostaneet pohjan oppaan tekijän ohjelmointiosaamiselle, jota voidaan ajatella oppaan asiasisällön pääasiallisena lähteenä. Tärkeimmäksi huomioitavaksi asiaksi oppaan kehityksessä tutkittujen lähteiden pohjalta nousi esimerkkien tärkeys oppimisen tukena.

3 Tutkimusprosessi

Koska työn tavoitteena on tuottaa uutta materiaalia, on työ konstruktivisen tutkimusmallin mukainen. Malli voidaan jakaa kuuteen vaiheeseen [Oyegoke, 2011]. Nämä kuusi osaa ovat järjestyksessä:

- Löydä käytännön ongelma, jolla on tutkimuspotentiaalia
- Rakenna yleispätevä ymmärrys ongelman aihealueesta
- Kehitä uusi konstruktio
- Osoita, että konstruktio toimii
- Osoita ratkaisun teoreettiset yhteydet ja tutkimusyhteydet
- Tarkastele luodun ratkaisun kattavuutta ja käyttökohteiden laajuutta

Koska toteutettava tutkimus on osa kandidaatintyötä, täytyy sen laajuutta hallita tarkasti. Täten kolme viimeistä vaihetta jäävät toteutukseltaan joko vajaiksi tai kokonaan tekemättä ajan puutteen ja liiallisen laajuuden vuoksi.

3.1 Tutkimusongelman rajaaminen

Työn tutkimusongelma vastaa edellä esitellyn mallin ensimmäiseen vaiheeseen: löydä käytännön ongelma, jolla on tutkimuspotentiaalia. Pelkkä ”tuota ohjelmointiopas kurssille” ei tätä tyydytä, joten ongelmaa on rajattu tarkemmin. Johdannossa todettiin jo toivotun toteutuksen olevan helppokäyttöinen ja tehokkaasti laajennettavissa. Täten tutkimuskysymystä lähdettiin miettimään tältä kannalta. LUT-yliopiston ”Ohjelmoinnin perusteet-kurssin opas on pdf-dokumentti melko hillityllä muotoilulla [Vanhala and Nikula, 2020] ja Olio-ohjelmointi -kurssin edellinen opas yksinkertainen Google Docs -dokumentti. Työn aloituksen yhteydessä nostettiin tärkeäksi tavoitteeksi selvittää, onko toinen näistä toimiva tapa toteuttaa opas, vai löytyykö vaihtoehtoisia parempia tapoja oppaan toteutukseen. Tältä pohjalta lopulliseksi tutkimuskysymykseksi valittiin ”*Kuinka parantaa Olio-ohjelmointi -kurssin ohjelmointiopasta?*” jaettuna kahteen alakysymykseen: ”*Millä tekniikalla opas tulisi kasata ja ylläpitää?*” ja ”*Mitä oppaan tulisi sisältää?*”

3.2 Yleiskatsaus aihealueesta

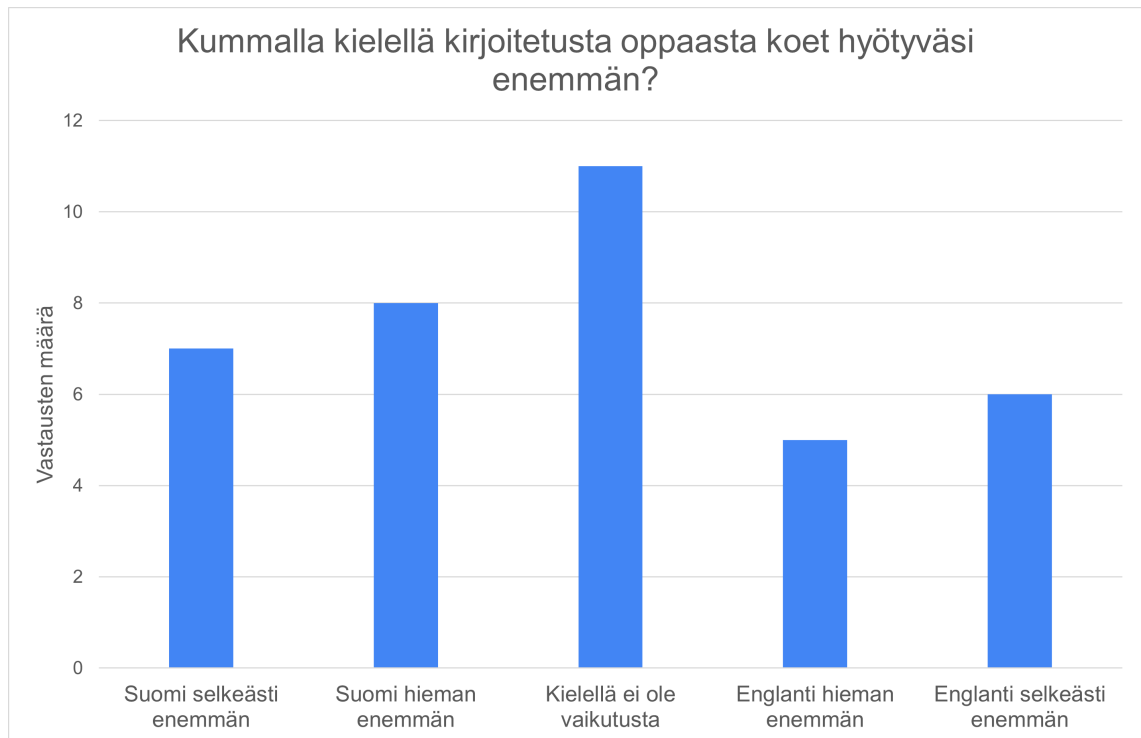
Edellä mainittujen toteutustapojen vahvuuksien ja heikkouksien selvittämisen lisäksi työn yhteydessä tutkittiin muitakin tapoja kirjamaisen ohjelmointioppaan toteuttamiseen. Yksi esille nostetuista ideoista oli oppaan koodiesimerkkien laajentaminen interaktiivisen koodin ajo- ja muokkausominaisuuksien avulla. Tämän pohjalta mahdollisten toteutustapojen listalle lisättiin Jyväskylän yliopiston interaktiivinen oppimisalusta The Interactive Material (tästä eteenpäin TIM) [tim, 2022]. Lisäksi jonkinlaista itsetehtyä verkkosivutoeutusta harkittiin, mutta se hylättiin nopeasti valtavaksi kasvavan luomisen ja ylläpidon työmäärän vuoksi.

Perinteisemmissä dokumenttipohjaisissa ratkaisuissa todettiin pdf-muodon olevan paras ratkaisu. Dokumentin toteutustapaa mietittiin LaTeX-taittokielen ja tekstinkäsittelyohjelmataratkaisun, kuten Microsoft Wordin välillä. LaTeX-kieli tarjoaa mahdollisuuden monimuotoisempiin teknisiin ominaisuuksiin, kuten koodin esittämiseen siistimmin ja monipuolisempiin viitauksiin dokumentin sisällä, mutta sen riski on hankalampi aloituskynnys ja tästä muodostuva mahdollinen muokkaamisen hankaloituminen. Yksinkertaisemmat tekstinkäsittelyohjelmat helpottaisivat oppaan kirjoittamisen aloitusta ja mahdollista tulevaa tekstisisällön muokkaamista, mutta uhraavat toiminnallisuutta suhteessa LaTeX-kieleen [León, 2010].

Näiden kolmen vaihtoehdon välillä päädyttiin lopulta LaTeX-kielellä tuotettuun dokumenttiin. Päätökseen johtaneet seikat esitellään tarkemmin työn kappaleessa 4, mutta pääsyyt olivat luotettavuus ja helppokäyttöisyys suhteessa TIM-alustaan ja paremmat koodin esittelyyn tarjotut työkalut sekä esimerkkikoodin muokkaamisen helppous suhteessa tekstinkäsittelyohjelmiin.

3.3 Oppaan kirjoittaminen

Toteutustapojen analyysin perusteella opas kirjoitetaan LaTeX-kielellä ja jaetaan opiskelijoille pdf-muodossa. Oppaan kielenä käytetään suomea opiskelijapalautteen perusteella (Kaavio 1). Oppaan rakenteen pohjana käytetään kurssin materiaalia, joka rakentuu osittain Eckelin Thinking in Java -kirjan päälle [Eckel, 2003]. Valmiin oppaan tavoitteena on näin olla ajankohtaisempi, helppokäyttöisempi opiskelijoille ja helpommin laajennettavissa kurssin toteuttajille kuin nykyisen kurssilla käytössä olevan ohjelmointioppaan.



Kaavio 1. Opiskelijakyselyn tulokset oppaan kielen koetusta vaikutuksesta oppaan käyttöön

3.4 Lopputuloksen arviointi

Kuten edellä todettiin, työn laajuuden rajaamiseksi oppaan toteutuksen onnistumisen testaaminen ja arviointi on rajattu työn alueen ulkopuolelle. Testaamiseen voitaisiin käyttää muokattua versiota kyselystä, jonka pohjalta opas alun perin luotiin. Poistamalla kyselystä toiveet oppaan tyylistä ja keskittymällä koettuun oppaan hyödyllisyyteen sekä erityisesti kyselyn taitotason itsearviointiin ja yhdistämällä vastaukset jo toteutetun kyselyn tuloksiin, saataisiin vertailukelpoisia lukuja ennen ja jälkeen oppaan julkaisun.

Oppaan arviointi puhtaana opiskelijoiden muuttuneen kurssimenestyksen mukaan saattaisi myös olla mahdollista, mutta siinä on ennustettavissa pieniä ongelmia. Kyselyn mukaan suurin osa opiskelijoista käyttää tai on käyttänyt ohjelmointiopasta jollain LUT-yliopiston sen tarjoamasta kolmesta kurssista. Ongelmallista tässä on, ettei kysely erikseen eritellyt Ohjelmoinnin perusteiden, Olio-ohjelmoinnin ja C-ohjelmoinnin kurssien oppaiden käyttöasteita ja kurssiopettajien mukaan Olio-ohjelmointi -kurssin opas on tarjotuista oppaista vähiten käytetty. Jos ohjelmointiopasta ei käytetä osana kurssin opiskelua ei mahdollisen opiskelumenestyksen muutoksen syyksi voida tulkita ohjelmointioppan vaikutusta. Tästä syystä olisi järkevää toteuttaa kysely oppaan käyttöasteesta ennen suoran oppaan julkaisun mahdollista korrelaatiota opiskelijoiden muuttuneisiin kurssituloksiin käytettäisiin oppaan arviointiin. Näin vältettäisiin virheellinen arvio, jossa vain

pieni osa kurssilaisista käyttää opasta, mutta oppaan julkaisua pidettäisiin pääasiallisena muuttujana kurssimenestyksen arvioinnin yhteydessä.

Opasta voidaan näin ollen pitää onnistuneena, jos se tuottaa positiivisen muutoksen opiskelijoiden kurssimenestykseen, nostaa oppaan käyttöastetta kurssilla, tai mahdollistaa oppaan helpomman ylläpidon suhteessa nykyiseen oppaaseen huonontamatta oppaan laatua. Negatiivinen opiskelijapalaute olisi varmin merkki epäonnistumisesta, mutta myös esimerkiksi oppaan käyttöasteen lasku tai jatkokehityksen mahdottomuus voidaan laskea epäonnistumisen merkeiksi.

4 Oppaan tekninen toteutus

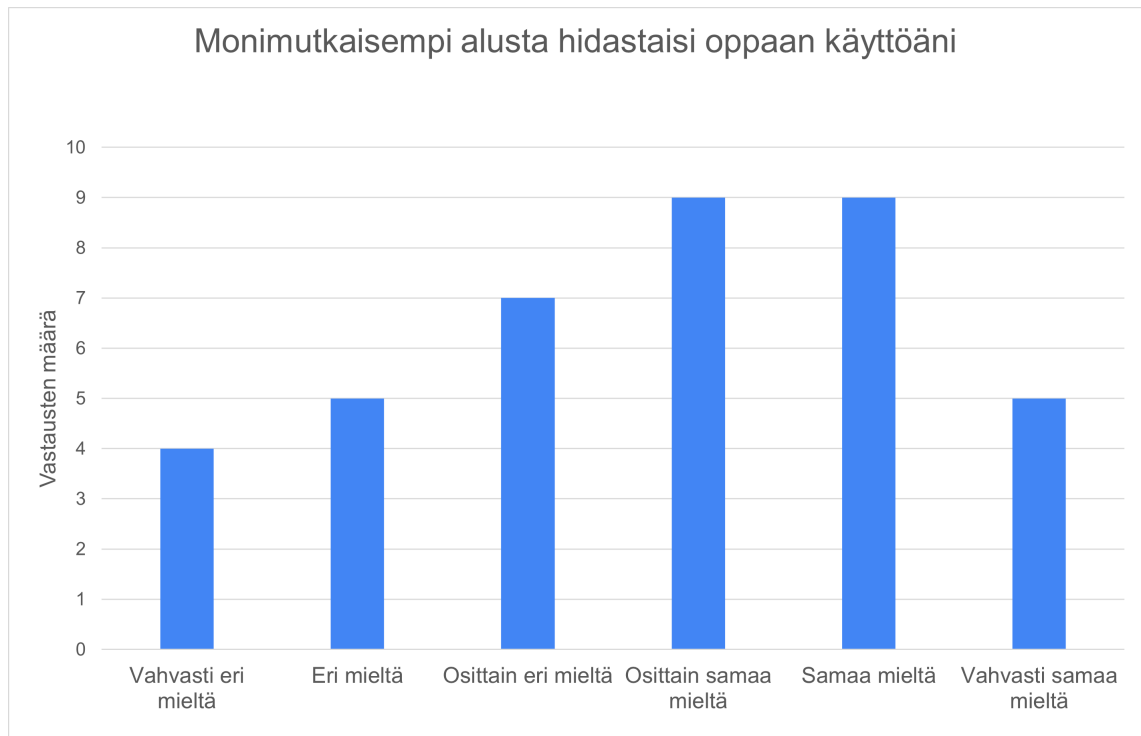
Oppaan teknisen toteutuksen suunnittelussa päädyttiin harkitsemaan kolmen vaihtoehdon välillä. Nämä vaihtoehdot olivat Jyväskylän yliopiston TIM-alusta interaktiivisille oppimateriaalikonaisuuksille, LaTeX-taittokielellä toteutettu pdf-dokumentti ja normaalilla tekstinkäsittelyohjelmalla, esimerkiksi Microsoft Wordilla toteutettu pdf-dokumentti. Toteutustapoja arvioitiin suhteessa oppaan toivottuihin ominaisuuksiin, toteutustavan käytettävyyteen, sekä opiskelijakyselyssä esiin nousseisiin toiveisiin.

4.1 The Interactive Material -alusta

The Interactive Material (TIM) on Jyväskylän yliopiston tuottama verkkopohjainen palvelu, joka nimensä mukaisesti tähtää tarjoamaan oppilaitoksille mahdollisuuden luoda interaktiivista kurssimateriaalia helposti. Palvelu tarjoaa muun muassa mahdollisuuden tuottaa ja jäsenellä papyrusmaista jatkuvasti rullaavaa materiaalia, upottaa materiaaliin apuohjelmia, sekä tuottaa elektronisen materiaalin pohjalta fyysisiä dokumentteja [tim, 2022].

TIM-palvelun suurimpana vahvuutena nähtiin mahdollisuus upottaa toimivaa ja muokattavaa esimerkkikoodia oppaan keskelle. Tämän toivottiin auttavan oppimisprosessissa vahvistamalla käytännön ja teorian sidettä oppaassa. Tämänlainen esimerkin ja harjoituksen yhdistäminen olisi erinomainen tapa toteuttaa Kaspersenin ja Bennedsenin suositus nopeasta vaihtelusta näiden kahden välillä [Caspersen and Bennedsen, 2007].

Tämän lisäksi TIM:in tarjoama helppo navigointi oppaan osien välillä, sekä mahdollisuus merkitä osia luetuksi ja piilottaa luettuja osia näkyvistä nähtiin positiivisena puolena palvelun käytössä. Opiskelijakyselyn palaute ei kuitenkaan tukenut oletusta tällaisten monipuolisten, mutta monimutkaisten ominaisuuksien tarpeesta oppaassa. Moni opiskelija koki monimutkaisemman alustan hidastavan oppaan käyttöä (Kaavio 2). Tämä lopputulos yksin oli odotettavissa, mutta kun sen yhdistää oletettua kysyntää matalampiin keskiarvoihin hyödyttömien aihealueiden piilottamisen tarpeessa opiskelijoiden keskuudessa (Kaavio 3), sekä hieman suurempaan, muttei silti ylitsevuotavaan kysyntään interaktiiviselle koodikonsolille (Kaavio 4), herää kysymys tuottaisiko TIM-palvelun käyttö vain turhaa monimutkasuutta oppaan käyttöön ja kehitykseen. Palvelun tarjoamien lisätoimintojen huonot puolet eivät välttämättä korvaannu lisätoimintojen luomalla arvolla. TIM kärsi vertailussa myös tuntemattomuudesta ja epävarmuudesta. Palvelu ei ole erityisen iso ja



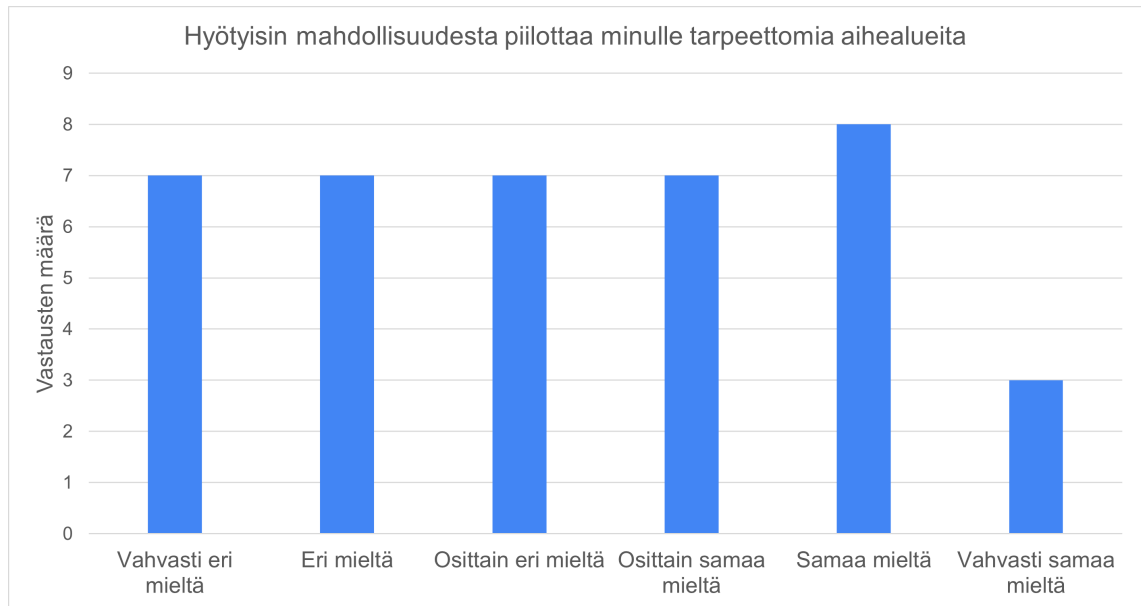
Kaavio 2. Opiskelijakyselyn tulokset oppaan alustan monimutkaisuuden koetusta vaikutuksesta oppaan käyttöön

siellä toteutettu opas luottaisi Jyväskylän yliopiston kykyyn ylläpitää palvelua.

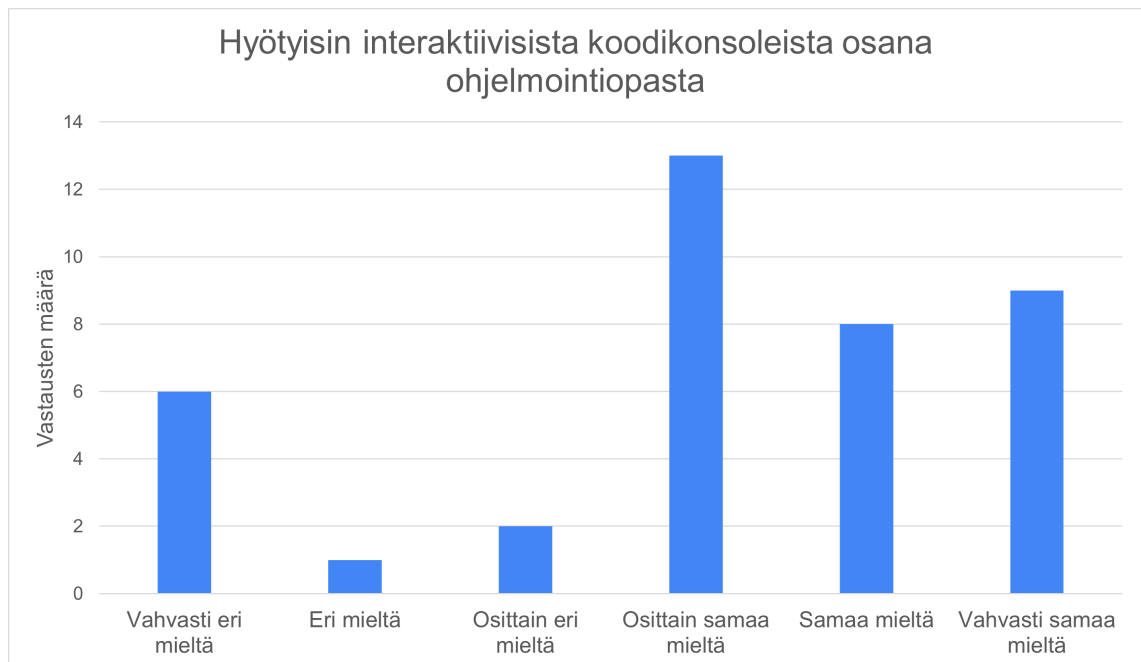
4.2 LaTeX-taittokieli

LaTeX-taittokieli on saavuttanut vakiintuneen paikan matemaattis-luonnontieteellisessä akateemisessa kirjallisuudessa [Brischoux and Legagneux, 2009]. Se mahdollistaa koodimaisen dokumentin rakentamisen esittämälle esimerkiksi kaavat, upotukset ja viittaukset erilaisina komentoina tai erityisympäristöinä [León, 2010]. Ohjelmointioppaan toteutuksessa sen vahvuutena nähtiin kyky tuottaa yksinkertainen dokumentti monipuolisilla taustatoiminnallisuuksilla. Vaikkei LaTeX-kielellä tuotettu pdf-dokumentti kykene TIM:in kaltaiseen interaktiivisuuteen, toimivat esimerkiksi dokumentin osien väliset viittaukset ja mahdollisen sanaston luonti LaTeX:lla huomattavasti perinteisiä tekstinkäsittelyohjelmia tehokkaammin.

Yksi LaTeX:in käyttöä arvioitaessa esiin noussut vahvuus suhteessa tekstinkäsittelyohjelmiin on esimerkkikoodin esittäminen oppaassa. Tekstinkäsittelyohjelmat vaativat joko erillistä ulkoasun muokkausta tekstille, jota esitellään esimerkkikoodina, tai kuvien käyttöä esimerkkikoodina. Molemmat tuottavat ongelmia esimerkkikoodia kehitettäessä. LaTeX



Kaavio 3. Opiskelijakyselyn tulokset aihealueiden piilotuksen mahdollisuuden koetusta vaikutuksesta oppaan käyttöön



Kaavio 4. Opiskelijakyselyn tulokset interaktiivisten koodikonsolien koetusta vaikutuksesta oppaan käyttöön

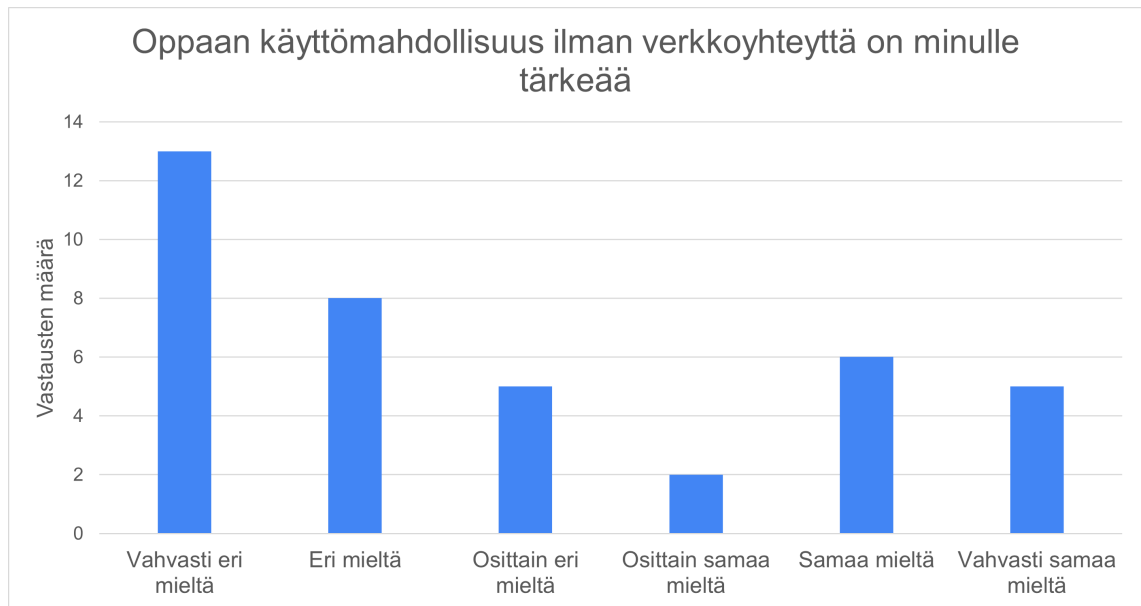
sen sijaan tarjoaa useamman ratkaisun koodin esittämiseen helposti joko ilman väreillä tapahtuvaa korostusta, tai väreillä korostettuna. Erityisesti jälkimmäinen vaihtoehto olisi lähes mahdoton ylläpitää tekstinkäsittelyohjelmassa. Toisaalta koodin korostaminen väreillä ei välttämättä auta koodin lukemisessa [Hannebauer et al., 2018], mutta koska se vähintään parantaa lukemiskokemusta [Hakala et al., 2006], oli se toivottu ominaisuus oppaaseen.

LaTeX:in huonot puolet toteutustapojen arvioinnissa kohdistuivat lähinnä sen erityistä opettelua vaativaan käyttöön [April, 2015]. LaTeX-toteutuksen nähtiin tarjoavan selviä etuja oppaaseen, mutta hankaloittavan kehitystyötä suhteessa pdf-dokumentin tuottamiseen tekstinkäsittelyohjelmaalla. Lisäksi jokaisen mahdollisen oppaan jatkokehittäjän olisi opiskeltava kielen alkeet ja oppaassa käytetyt LaTeX-kirjastot ja -komennot opasta muokataksien.

LaTeX:in vertailu TIM-palveluun puolestaan keskittyi vaihtoehtojen luotettavuuteen ja käytettävyyteen opiskelijakäytössä. Molemmat toteutustavat vaativat enemmän panostusta ja asiantuntijuutta toteutusvaiheessa, mutta tarjoavat erilaisia etuja suhteessa toisiinsa. TIM:in interaktiivisuus ja LaTeX:in luotettavuus ja yksinkertaisuus käyttäjän näkökulmasta ovat molemmat huomattavia myönteisiä puolia. Alustava epävarmuus alustojen välillä kuitenkin hälveni hieman opiskelijakyselyssä esiin nousseiden toiveiden pohjalta. TIM:iä käsitelleen kappaleen tulosten lisäksi opiskelijoista yhteensä noin neljäsosa koki oppaan käytön ilman verkkoyhteydestä tärkeäksi tai erittäin tärkeäksi (Kaavio 5). Tätä vastausta painotettiin valinnan yhteydessä, koska oppaan haluttiin ennen kaikkea olla opiskelijoilla saatavilla aina heidän sitä tarvitessaan. TIM tarjoaa mahdollisuuden hakea materiaalin tekstiosuuden pdf-muodossa, mutta tässä muodossa palvelun interaktiivisuuden edut häviävät.

4.3 Tekstinkäsittelyohjelmat

Kolmas harkittu toteutustapa oppaalle oli Microsoft Wordin tai Libre Office Writerin kaltaisella tekstinkäsittelyohjelmalla tuotettu pdf-tiedosto. Tämän vaihtoehdon hyvänä ja huonona puolena pidettiin sen yksinkertaisuutta. Niin oppaan kehitys kuin mahdollinen tulevaisuuden jatkokehitys ei vaatisi erityisosaamista. Microsoft Word on myös tuotavampi ympäristö normaalin tekstin kirjoittamiseen LaTeX:iin verrattuna [April, 2015]. Toisaalta oppaan normaalista tekstistä poikkeava luonne antoi aiheita epäillä, näkyisivätkö tutkimuksessa esitellyt hyödyt oppaan kirjoitustyössä LaTeX-kappaleessa esiin nostettujen seikkojen vuoksi.



Kaavio 5. Opiskelijakyselyn tulokset oppaan verkkoyhteydettömän käyttömahdollisuuden koetusta vaikutuksesta oppaan käyttöön

4.4 Päätös toteutustavasta

Lopullinen päätös käytettävästä toteutustavasta muodostui valinnaksi TIM:in ja LaTeX:in välillä. Tekstinkäsittelyohjelmien hyviä puolia, eli editoinnin helppoutta ei katsottu voitavan hyödyntää työn valmiiksi monipuolisten ja teknisten vaatimusten takia. Monet tekstinkäsittelyohjelmien helppokäyttöisyyttä edistävät toiminnot eivät auta esimerkiksi esimerkkikoodien esittämisessä tai sanastojen kirjoittamisessa. Täten tekstinkäsittelyohjelmat nousivat esille lähinnä puuttellisten ominaisuuksiensa takia pystymättä tarjoamaan pääasiallisia hyötyjään. Taulukko 1 esittelee eri toteutustapojen tunnistetut hyvät ja huonot puolet.

	TIM	LaTeX	Tekstinkäsittelyohjelma
Hyvää	<ul style="list-style-type: none"> - Interaktiivinen - Piilotettavat alaosiot 	<ul style="list-style-type: none"> - Laadukas koodin esittäminen - Helppokäyttöinen dokumentti - Verkkoyhteydetön käyttö 	<ul style="list-style-type: none"> - Helppo leipätekstin muokkaus - Helppokäyttöinen dokumentti - Verkkoyhteydetön käyttö
Huonoa	<ul style="list-style-type: none"> - Käytettävyys epävarmaa - Alustan nuoruus - Riippuvainen alustan ylläpidosta 	<ul style="list-style-type: none"> - Oppimiskäyrä kehityksessä - Jatkokehitys uuden kehittäjän toimesta vaatii paneutumista 	<ul style="list-style-type: none"> - Esimerkkikoodin esittäminen hankalaa - Kehittyneempien tekstien ominaisuuksien puute

Taulukko 1. Oppaan toteutustapojen hyvät ja huonot puolet

Päätös TIM:in interaktiivisuuden ja LaTeX:in monipuolisuuden välillä ei ollut helppo ja lopulta opiskelijapalautteen tarpeeksi huomattava määrä verkkoyhteydetöntä käyttöä toivovia vastauksia käänsi mielipiteen kohti LaTeX:illa toteutettua pdf-dokumenttia. Koodikonsolien upottamisen mahdollisuuden käyttämättä jättäminen ei ollut helppo päätös, mutta edellä mainitut syyt nähtiin LaTeX:ia suosivina.

4.5 Käytetyt ohjelmistot

Oppaan kehittäminen suoritettiin LaTeX-kielellä käyttäen versionhallintaan Git-ohjelmaa ja GitHub-palvelua. Esimerkkikoodit kirjoitettiin omaan Git-repositorioonsa. Alun perin tarkoituksena oli kehittää opas Overleaf-pilvipalvelulla [Browder and Cross, 2018], mutta sen ilmaisversion todettiin olevan yhteensopimaton suunnitellun kansiorakenteen ja Git-työkalun käytön kanssa. Repositoriot ovat näkyvillä julkisesti [opa, 2022] [esi, 2022].

Lopulliset ohjelmistovalinnat oppaan kirjoittamiseen olivat Texmaker-ohjelmisto LaTeX-tiedostojen muokkaamiseen [tex, 2022], IntelliJ IDEA-kehitysympäristö esimerkkikooditiedostojen muokkaamiseen [int, 2022] ja Sourcetree-ohjelmisto versionhallinnan graafisena käyttöliittymänä [sou, 2022].

```

Juuri
|
|___ OhjelmointiOpas.tex
|
|___ Sanastot
|   |___ JavaAvainsanat.tex
|   |___ Sanasto.tex
|
|___ Kuvat
|
|___ Esimerkkikoodirepositorio

```

Kuva 1. Havainnollistus ohjelmointiopasprojektin kansiorakenteesta

4.6 Versionhallinnan käyttäminen LaTeX-dokumentin yhteydessä

Työn johdannossa, sekä kappaleessa 3.4 ”Tutkimusmenetelmä: lopputuloksen arviointi”, mainittiin yhden työn tavoitteista olevan helppo laajennettavuus ja ylläpidettävyys. Vaikka mahdollisuus Git-pohjaisen versionhallinnan käyttöön LaTeX-dokumenttia kehitettäessä ei vaikuttanut päätökseen ohjelmointioppaan lopullisesta toteutustavasta, LaTeX’in koodimainen tiedostorakenne mahdollisti Git’in käyttämisen oppaan versionhallintaan, tuoden mukanaan versionhallinnan edut.

Ohjelmointioppaan LaTeX-projektin tiedostorakenne suunniteltiin täten Git-pohjainen versionhallinta edellä. Ideana oli eristää oppaan esimerkkikoodit erilliseen repositorioon, joka voitaisiin jakaa opiskelijoille oppaan kanssa. Täten opiskelijoilla on mahdollisuus käydä itsenäisesti tutkimassa tai ajamassa oppaan esimerkkikoodia. Lopputuloksena oli Kuvassa 1 esiteltyä tiedostopuuta vastaava kansiorakenne, josta esimerkkikoodirepositorio oli versionhallinnan silmissä piilotettu erilliseksi itsenäisesti hallittavaksi repositorioksi.

Vastaavasti esimerkkikoodirepositorion kansiorakenteessa pystyttiin LaTeX-kielen tarjoaman helpon koodien integroimisen vuoksi mallintamaan kurssin aikataulu niin, että jokainen oppaan luku käytti erillistä kansiota, jonka alla luvun koodit olivat järjesteltyjä aihekohtaisiin alikansioihin. Tämä kansiorakenne on havainnollistettu Kuvassa 2.

Versionhallintaan valittiin Git-ohjelmiston tueksi käyttöön GitHub-pilvipalvelu repositorion säilymiseen. Palvelun käytön yhteydessä oppaan pääasiallisena palautekanavana päätettiin koittaa käyttää GitHubin issues-ominaisuutta, jolla ulkopuoliset osallistujat voivat luoda esimerkiksi bugiraportteja tai ehdottaa parannuksia repositorioon [iss, 2022].

```

Juuri
|___ Luku 1
|   |___ Aihe 1
|   |   |___ Koodiesimerkki 1
|   |   |___ Koodiesimerkki 2
|   |___ Aihe 2
|   |___ Aihe n
|___ Luku 2
|___ Luku n

```

Kuva 2. Havainnollistus esimerkkikoodirepositorion kansiorakenteesta

4.7 Koodin esittäminen oppaassa

Esimerkkikoodi päätettiin esittää värillisesti korostettuna käyttäen Minted-kirjastoa, joka korostaa koodin annetulla kielellä dokumentin kääntämisen yhteydessä [Rudolph, 2011]. Kirjasto löytyi Overleaf-palvelun dokumentaatiosta suositeltuna kirjastona käytettäväksi koodin korostamiseen [ove, 2022]. Kirjastolle voi tarjota koodia joko tekstinä LaTeX-tiedostossa, tai kooditiedostosta luettuna antamalla polun kyseiseen tiedostoon. Kirjaston tuottamaa koodia voisi lisäksi muokata komennoilla esimerkiksi fontin tai korostuksen väriteeman suhteen, mutta oppaassa ei tehty näin. Kuvat 3, 4, 5, 6, 7 ja 8 esittelevät kirjaston eri tavat koodin esittämiseen.

```

\begin{minted}{java}
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
\end{minted}

```

Kuva 3. LaTeX-syntaksi esimerkkikoodin kirjoittamiseen tekstinä LaTeX-tiedostossa

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Kuva 4. Kuvan 3 komennon lopputulos

```
\inputminted{java}{HelloWorld.java}
```

Kuva 5. LaTeX-syntaksi esimerkkikoodin tuomiseen tiedostosta "HelloWorld.java"

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Kuva 6. Kuvan 5 komennon lopputulos

Yhden rivin mittaisen koodinpalan kirjoittaminen suoraan LaTeX-tiedostoon tapahtuu kuvan 7 esittämällä syntaksilla. Tämä tuottaa kuvassa 8 esitellyn lopputuloksen.

```
\mint{java}|public static void main(String[] args)|
```

Kuva 7. LaTeX-syntaksi yksirivisen esimerkkikoodin kirjoittamiseen tekstinä LaTeX-tiedostossa

```
public static void main(String[] args)
```

Kuva 8. Kuvan 7 komennon lopputulos

4.8 LaTeXin käyttö oppaassa

Oppaan kirjoittaminen aloitettiin Tufte-book LaTeX-pohjan päälle, koska pohja tarjosi valmiin muotoiluavun kirjalle ja näytti ulkoisesti houkuttevalta [tuf, 2022]. Pohja hylättiin myöhemmin, koska se ei tukenut kahta yhtä leveää marginaalia. Lopulliseksi dokumenttiluokaksi päättyi tämän jälkeen LaTeX:in oletusluokka kirjalle. Luokka muokattiin näyttämään marginaalit yhtä leveinä paketilla `geometry`.

Oppaan sanastot on tuotettu `glossaries`-kirjastolla. Oletussanaston lisäksi oppaaseen on lisätty toinen sanasto paketin komennolla `\newglossary`. Tämä mahdollisti teknisellä tasolla sanastoesineiden jakamisen kahteen eri sanastoon, jotka voitiin esittää erillisinä kokonaisuuksina omilla otsikoillaan ja erillisillä automaattisesti luoduilla sisällysluetteloviittauksilla kuvassa 9 esitellyllä koodinpätkällä. Kirjaston käytön kanssa ilmeni pieni ongelma, jossa ääkkösmerkit eivät toimineet sanastoesineiden nimissä. Tämä korjattiin korvaamalla merkit aakkosvastineilla, niin että esimerkiksi sana `ilmentymä` korvattiin sanalla `ilmentyma`.

```
\printglossary[title=Sanasto, toctitle=Sanasto]
\printglossary[type=java, title=Javan avainsanat, toctitle=Javan avainsanat]
```

Kuva 9. LaTeX-komento sanastojen esittämiseen oppaan lopussa

Oppaan LaTeX-projektin rakenne kärsi teknologian vieraudesta. Hyvässä projektirakenteessa kirjamaisessa projektissa olisi yksi päätiedosto, jossa määritellään projektin ulkoasu ja muoto ja johon tuodaan leipäteksti erillisistä kappalekohtaisista tiedostoista. Tämän sijaan projekti koostuu yhdestä isosta tiedostosta, joka vastaa kaikesta, paitsi sanastojen sisällöstä. Tämä ei aiheuttanut kehityksen aikana suuria ongelmia, mutta voi hidastaa jatkokehitystä.

Ylipäätään projektin kirjoittaminen LaTeX-kielillä haittasi kehitystä vähemmän kuin aluksi arveltiin, mutta tarjosi silti huomattavia etuja. Teknologiavalinnan suhteen tehtiin siis vähintään hyvä päätös, joskin ilman kontrollia vaihtoehtoisilla teknologioilla toteutetuista vaihtoehtoista ei ole mahdollista sanoa, olisiko esimerkiksi TIM tarjonnut yhtä laadukkaan kehitysympäristön.

5 Oppaan sisällön suunnittelu

Kun oppaan tekninen toteutus oli selvillä, suunnittelutyö siirtyi oppaan rakenteeseen. Tämä vaihe oli edellistä selkeämpi, koska oppaan luonne tiukasti kurssiin kytkeytyneenä materiaalina määräsi suuren osan oppaan asiasisällöstä ja sen järjestyksestä.

5.1 Oppaan yleinen rakenne

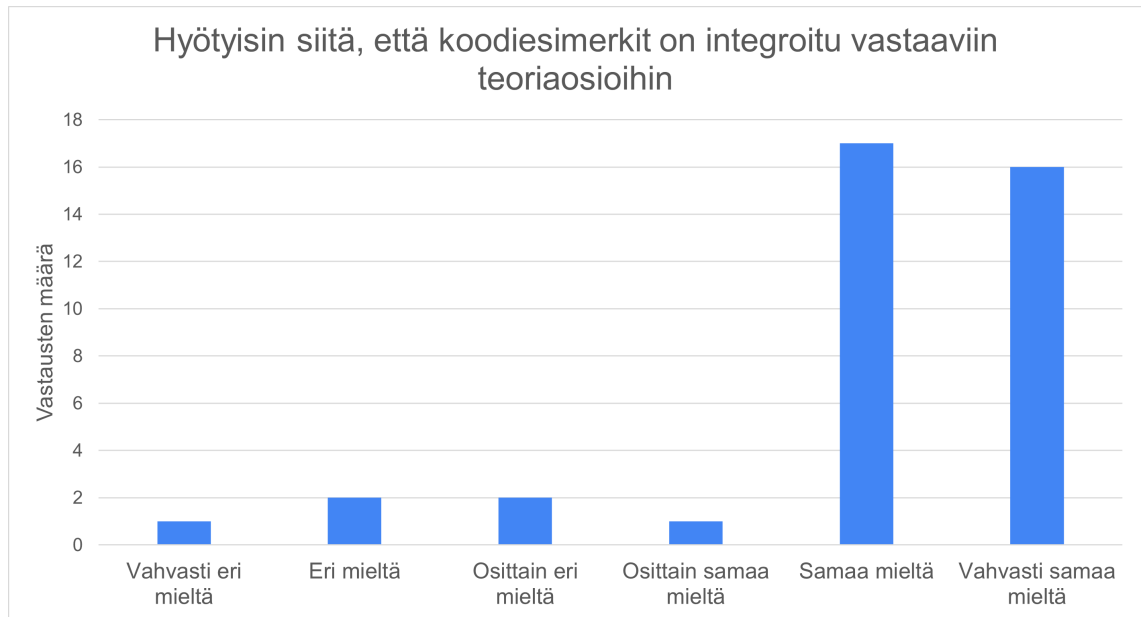
Oppaan sisällön toivottiin vastaavan kurssilla käsiteltäviä kokonaisuuksia mahdollisimman tarkasti, jotta ei ajauduttaisi tilanteeseen, jossa luennolla käsitellään yhtä aihealuetta, oppaassa toista, ja tehtävissä kolmatta samalla viikolla. Jotta opiskelijoiden olisi mahdollisimman helppoa seurata kurssin aikataulua oppaasta, päätettiin opas jakaa lukuihin kurssin viikkojen mukaan. Esittelyluvusta tehtiin luku 0, jotta viikon 1 asiasisältö saataisiin lukuun 1.

Tämän lisäksi oppaan suunnitteluvaiheessa mietittiin koodiesimerkkien sijaintia. Edellisessä kurssin ohjelmointioppaassa jokainen kappale oli koostunut ensin teoriaosuudesta, jota oli kappaleen lopussa seurannut esimerkkikoodia. Kyselyn perusteella esimerkkikoodin sijainti lähempänä teoriavastinetta kyseiselle koodille oli opiskelijoiden keskuudessa toivotumpaa kuin kaiken esimerkkikoodin kasaaminen luvun loppuun (Kaavio 6) (Kaavio 7). Tätä lähestymistapaa tukee myös muun muassa Kaspersenin ja Bennedsenin artikkeli ohjelmointikurssin suunnittelusta [Caspersen and Bennedsen, 2007].

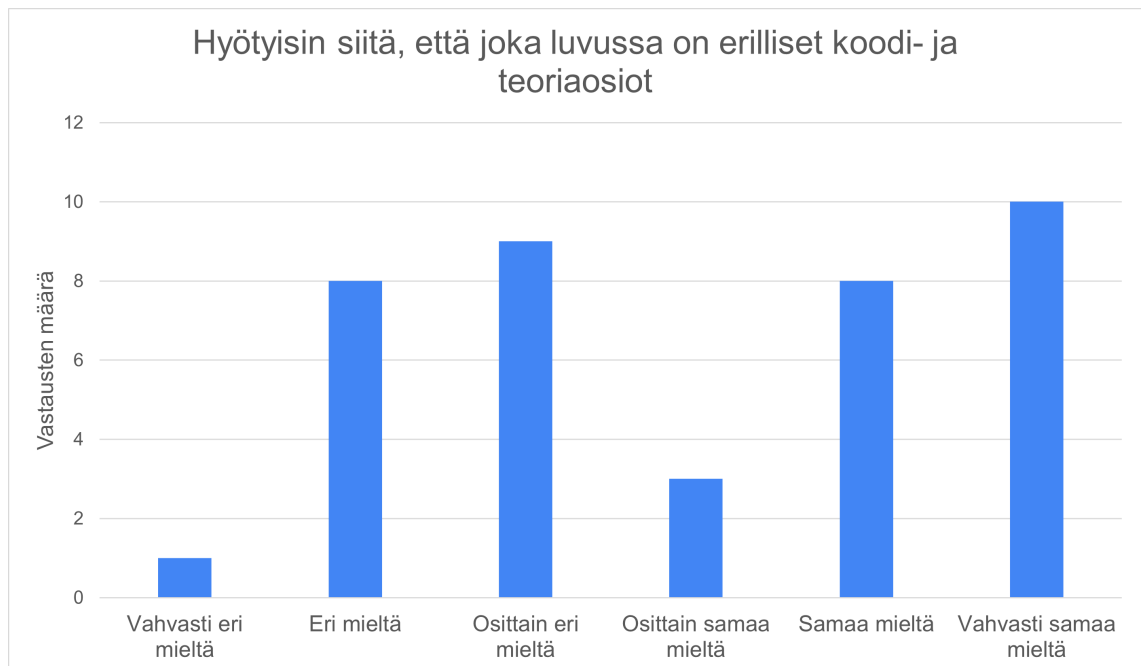
5.2 Sanastot

Opas päätettiin toteuttaa suomeksi opiskelijapalautteen perusteella (Kaavio 1). Kuitenkin suurin osa ohjelmointiresursseista internetissä käyttää pääkielenään englantia. Lisäksi oppaan yhteydessä haluttiin tarjota nopea tapa hakea selityksiä kurssilla esitellyille uusille termeille, sekä kurssilla käytetyille Javan avainsanoille. Tästä syystä oppaaseen päätettiin toteuttaa sanasto-osuus. Sanasto päätettiin jakaa kahteen osaan: normaaliin termistöön ja Javan avainsanoihin. Näin pyrittiin helpottamaan sanaston käyttöä tiedonhakuun.

Sanastoja käytetään oppaassa tästä syystä paitsi avaamaan termistöä, myös tarjoamaan englanninkieliset käännökset termistölle itsenäistä tiedonhakua varten. Normaalin termistösanaston lisäksi oppaaseen sisällytetty Javan avainsanoja ja standardikirjaston luok-



Kaavio 6. Opiskelijakyselyn tulokset koodiesimerkkien ja teoriaosuuksien yhdistämisen koetusta vaikutuksesta oppaan käyttöön



Kaavio 7. Opiskelijakyselyn tulokset koodiesimerkkien ja teoriaosuuksien erottamisen koetusta vaikutuksesta oppaan käyttöön

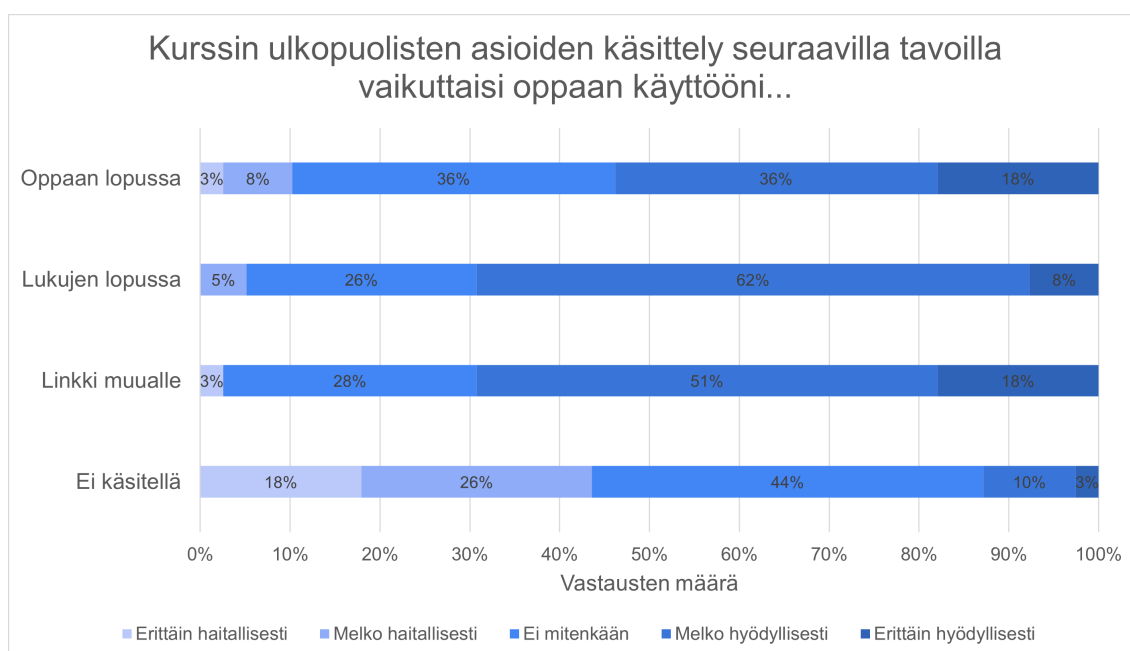
kia sisältävä sanasto pyrkii toimimaan nopeana referenssinä avainsanojen toimintaan, mikäli opiskelija ei halua hakea avainsanan tai luokan toiminnan ja käytön perusteita esimerkiksi Javan standardikirjastosta. Opas korostaa jokaisen uuden viittauksen kummastakin sanastosta, jotta opiskelija osaisi kiinnittää huomiota tuntemattomiin termeihin ja Java-konsepteihin.

5.3 Android-API:n käsittely

Yksi suurimmista kysymyksistä oppaan sisällön rakenteen suunnittelun aikana oli, miten kurssin Android-API painotteinen loppupuolisko esitetään oppaassa. Opiskelijakyselyssä ei kysytty erikseen opiskelijoiden tyytyväisyydestä tähän osuuteen kurssia, mutta saadussa vapaassa palautteessa oli nostettu Android-API:n käyttö esiin muun muassa seuraavilla tavoilla.

- Mikä on ollut huonoa/mitä toivoisit parannettavan nykyisissä oppaissa?
 - *”Tieto on osittain vanhentunutta / puutteelliset esimerkit Androidista.”*
- Vapaa sana kurssimateriaalista ja ohjelmointioppaasta:
 - *”Kurssilla ei ole tarpeeksi opasteista android studion viikkotehtäviin.”*
 - *”muun kurssilla käytettävän materiaalin opetus on minimalistista. (GitHub ja android studio). Tuntuu, tuntuu että kurssi on suunniteltu sen pohjalta, että oppilaalla on jo pohjaa Javaan, GitHubin ja androidstudion käyttöön. Jos kurssilla on liikaa materiaalia opetettavaksi tulisi se leikellä pienempiin osiin. Paljon tehtävää ja itseopiskelua 6 nopan eteen.”*

Vastausten mukaan kurssi hyötyisi erillisestä Android-API:n käytössä opiskelijoita ohjaavasta oppimateriaalista. Vastauksissa on kuitenkin nostettu hyvin esille myös tällaisen materiaalin ohjelmointioppaaseen lisäämisen huono puoli tiedon vanhentumisen suhteen. API:t muuttuvat osana niiden luonnollista kehitystä. Esimerkiksi Android API:n versioiden 30 ja 31 välillä yhdeksän toiminnallisuutta julistettiin vanhentuneiksi [and, 2021]. Vaikka oppaan on tarkoitus olla helposti laajennettavissa, ei sen ole tarkoitus olla riippuvainen kurssista ulkopuolisen tekijän päättämistä muutoksista. Android API:a koskevan oppimateriaalin tuottaminen päätettiin täten jättää oppaan tuottamisesta erilliseksi projektiiksi, johon saatetaan joskus myöhemmin palata osana kurssin kehitystä.



Kaavio 8. Opiskelijakyselyn tulokset erilaisten oppaan ulkopuolisten aihealueiden esittämistapojen koetuista vaikutuksista oppaan käyttöön

Tämän sijaan oppaan lukuun seitsemän harkittiin esimerkkityyppistä tutustumista johonkin vakaampaan kirjastoon. Tämän esimerkin tarkoituksena olisi opettaa opiskelijoille, kuinka API-dokumentaatiota luetaan. Ideaa pidettiin hyvänä, mutta se päätettiin jättää varalle niin, että opas toteutetaan ilman sitä, mikäli kehitysaika loppuu kesken.

5.4 Kurssitavoitteiden ulkopuolisen materiaalin käsittely

Oppaan suunnitteluvaiheessa nousi esille idea kurssin aiheita sivuavien, mutta kurssin oppimistavoitteiden ulkopuolisten aiheiden esittelystä oppaassa. Tämän idean pohjana oli sisällyttää oppaaseen ylimääräisiä lukuja näistä aiheista, yksi per oppaan normaali luku. Bonuslukujen ehdotettuja aiheita olivat muun muassa versionhallinta Git-ohjelmistolla ja GitHub-palvelulla, siistin koodin tuottaminen, Javan stream-rajapinta kokoelmien käsittelylle, sekä yksikkötestaamisen perusteet.

Idean järkevyyttä arvioitiin osittain palautekyselyn avulla. Suurin osa opiskelijoista oli verrattain neutraaleja tai lievästi myönteisiä idean suhteen (Kaavio 8). Koska tämäntyyppiselle toteutukselle ei löytynyt suurempaa kysyntää tai vastustusta, päätettiin idea jättää takalalle ja toteuttaa, mikäli aikaa riittäisi.

6 Oppaan sisällön esittely

Oppaan beta-versio valmistui ennen kevään 2022 Olio-ohjelmointi -kurssin alkua. Aikataulun tiukkuuden vuoksi sekä ylimääräiset kappaleet, että tutustuminen API-dokumentaation lukemiseen jäivät toteuttamatta oppaassa. Beta-versio sisälsi kuitenkin kaiken muun suunnitellun sisällön, käyden läpi kurssin koko oppimateriaalin Android-API:n käyttöä lukuun ottamatta. Raportti ei käy tarkkaan läpi oppaan jokaista lukua oppaan verrattain huomattavan pituuden vuoksi. Itse opas koostuu tällä hetkellä 76 sivusta, joita seuraa vielä 12 sivua sanastoja. Oppaan luvut ja niiden aihealueet ovat seuraavat:

- **0: Oppaasta ja sen käytöstä** - Oppaan filosofian, rakenteen, käytön ja muotoiluseikkojen esittely
- **1: Oliopohjainen ajattelu** - Perehdytys oliopohjaiseen ajatteluun
- **2: Javan perusteet** - Javan perusteet: perustietotyypit, luokan luonti, rakentajat, näkyvyysmääreet, peruslogiikka, tulostaminen ja syötteen vastaanottaminen
- **3: Kokoelmarakenteet, toistorakenteet ja lisää oliopohjaista ajattelua** - Lisää javan peruskäsitteitä: kokoelmarakenteet, toistorakenteet, erinäisiä avainsanoja kuten switch-case ja enum
- **4: Tiedostonhallinta ja virheenkäsittely** - Virhetilat ja try-catch-finally -rakenne, tiedostojen lukeminen ja tiedostoon kirjoittaminen
- **5: Oliopohjainen suunnittelu** - Aggregaatio ja kompositio, periytymisen alkeet, UML-kaaviot
- **6: Periytyminen** - Periytyminen Javassa, rajapintaluokat ja abstraktit luokat
- *(7: Kappale tiputettu oppaasta, koska kurssi käsittelee tällä viikolla ainoastaan Android-rajapintaa, joka on rajattu oppaan aihealueen ulkopuolelle)*
- **8: Luokkamuuttujat ja luokkametodit** - Avainsana static ja singleton-suunnittelumalli
- **9: SOLID-periaatteet, käyttäjän määrittelemät virheluokat** - SOLID-periaatteet, käyttäjän määrittelemät virheluokat ja raise-avainsana
- **10: Geneerisyys ja iteraattorit** - Geneerisyyden käsite ja Javan Iterator-rajapinta
- **11: Sisäiset luokat ja anonyymit luokat, kopiointi** - Sisäisten ja anonyymien luokkien käyttö ja erot. Kopiointi oliopohjaisissa kielissä: viitekopiointi, arvon kopiointi ja syväkopiointi

6.1 Oppaan alkumateriaali ja luvut 0 ja 1

Oppaan alkuun on normaalin sisällysluettelon lisäksi sijoitettu luettelot oppaan koodiesimerkeistä, UML-kaavioista ja taulukoista, jotta opiskelijat voivat hakea esimerkkejä nopeammin. Luku nolla esittelee oppaan idean, tavoitteet ja korostuvat Javan avainsanojen ja sanastotermien suhteen. Ensimmäisellä viikolla kurssilla ei vielä käsitellä Javaa, joten opas puhuu hieman oliopohjaisen ohjelmoinnin filosofiasta, ja esittelee kapseloinnin, koheesion ja pariutumisen käsitteet.

```

package week2.basicexamples;

// class-avainsana aloittaa luokan määritelmän. Sitä seuraa luokan nimi ja aaltosulkeisiin suljettu
// luokan runko. Pääluokan nimeä ei ole etukäteen määritelty ja onkin suositeltavaa nimetä se
// kuvailevasti, esimerkiksi koko ohjelman nimen mukaisesti.
class HelloWorld {

    // static-avainsana käsitellään oppaassa myöhemmin. Toistaiseksi riittää
    // että tiedostaa main-metodin vaadittavan olevan muotoa "public static void"
    public static void main(String[] args){

        // Kutsu println-metodiin tulostaa annetun merkkijonon ja rivinvaihdon
        System.out.println("Hello world!");
    }
}

```

Kuva 10. Oppaan koodiesimerkkien ulkoasu

Esimerkki 1.1. Ohjelma lähdekooditiedostossa

```
print("Hello World!")
```

Kuva 11. LUT-yliopiston ”Ohjelmoinnin perusteet-kurssin ohjelmointioppaan koodiesimerkkien ulkoasu

6.2 Javan perusteet: oppaan luvut 2, 3 ja 4

Oppaan luvut kahdesta neljään keskittyvät Javan perusteiden opettamiseen opiskelijalle. Käytännössä luvuissa käydään kolmessa viikossa läpi mikä LUT-yliopiston ohjelmoinnin perusteet -kursilla opiskeltavia python-ominaisuuksia vastaavat Javan ominaisuudet, sekä rakentajat ja main-metodi [Vanhala and Nikula, 2020]. Luvut ovat tästä syystä pitkiä. Tätä pidettiin hyväksyttävänä esitettyjen konseptien yksinkertaisuuden ja tuttuuden vuoksi. Luvuissa nähdään myös paljon koodiesimerkkejä syntaksin opetteluun helpottamiseksi. Kuvat 10 ja 11 esittelevät eron uuden ohjelmointioppaan koodiesimerkkien ulkoasun ja LUT-yliopiston ohjelmoinnin perusteet kurssin ohjelmointioppaan koodiesimerkkien ulkoasun välillä.

6.3 Perehtyminen periytymiseen ja oliopohjaiseen suunnitteluun: oppaan luvut 5 ja 6

Opiskelijoiden saatua perustyökälyt Javan kirjoittamiseen, luvut 5 ja 6 esittelevät periytymisen ensin konseptitasolla luvussa 5 ja sitten Javalla toteutettuna luvussa 6. Luvussa 5 esitellään myös UML-standardin luokkakaavio. Luku 5 ei sisällä lainkaan koodiesimerkkejä, ainoastaan UML-kaaviopohjaisia havainnollistavia esimerkkejä erilaisista luokkarakenteista.

Luvussa 6 käydään läpi periytymisen perusteiden lisäksi abstraktin luokan ja rajapin-

taluokan käsitteet ja serialisaatio. Serialisaatio on kurssimateriaalissa sijoitettu viikolle neljä, mutta sen täysi selittäminen opiskelijalle vaatii opiskelijan ymmärtävän periytymissen käsitteen, joten se päätettiin siirtää oppaassa sopivampaan kohtaan. Opiskelijat voivat edelleen löytää käsitteen hakemistosta tai sanaston avulla, mikäli tarvitsevat sitä aikaisemmin. Luvun 6 yhteydessä käsitellään myös rajapintaluokan ja abstraktin luokan käyttötapojen eroavaisuuksia.

6.4 Suunnittelumallien maailman oven raottamista: oppaan luvut 8 ja 9

Luku 7 on jätetty oppaasta kokonaan pois. Kuten kappaleessa 5.1 mainittiin, opas on suunniteltu niin, että jokaisen sisältöluvun lukunumero vastaa kurssiviikkoa, jolla kyseisen luvun sisältö käsitellään. Kurssiviikko 7 on viikko, jolloin Android-API:in tutustuminen kurssilla alkaa. Koska viikolla ei käydä uusia Java-käsitteitä ja opas ei käsittele Android-API:a, jätettiin luku kirjoittamatta. Lukunumeroa inkrementoidaan lukujen 6 ja 8 välissä LaTeX-komennolla.

Oppaan luvun 8 nimi on ”Luokkamuuttujat ja luokkametodit” ja luvun 9 nimi on ”SOLID-periaatteet, käyttäjän määrittelemät virheluokat” ja luvut käsittelevät näin hieman eri aihealueita. Molemmissa luvuissa on kuitenkin esitelty ajatusmalleja laadukkaamman koodin tuottamisen ympäriltä. Lisäksi molemmat luvut esittelevät yhden suunnittelumallin. Tässä kohtaa opasta luotetaan jo oppaan lukijan taitoon tuottaa toimivaa koodia ja tavoitteena on ohjata lukijaa pikkuhiljaa kohti laadukkaamman koodin tuottamista. Esimerkiksi suunnittelumallien käytön on todettu korreloivan myönteisesti koodin ylläpidettävyyteen [Bala and Kaswan, 2014].

Luku 8 esittelee luokkamuuttujien ja luokkametodien lisäksi kurssin etenemisen mukaisesti singleton-suunnittelumallin, jonka tavoitteena on varmistaa, että luokasta on olemassa vain yksi instanssi koko ohjelmiston laajuisesti. Malli ei ole täysin ongelmaton, eikä oppaassa pyritä esittämään sitä tällaisena. Esimerkiksi singleton-luokan periyttäminen tai monisäikeinen käyttö on ongelmallista [Lyon and Castellanos, 2007].

Luvussa 9 esitellään SOLID-periaatteet. Ne ovat alun perin Robert C. Martinin esittelemä ryhmä, joka sisältää viisi periaatetta, joiden käyttö tekee ohjelmistosta skaalautuvamman ja ylläpidettävämmän [Singh and Hassan, 2015]. Näiden lisäksi luku esittelee strategia-suunnittelumallin idean osana avoin/suljettu -mallin koodiesimerkkiä. Tämä suunnittelumalli ei kuulu kurssin osaamistavoitteisiin, joten opiskelijan huomiota ei kiinnitetä mal-

liin, vaan se jätetään taka-alalle SOLID-periaatteiden taustalle. Näin innokkaat opiskelijat voivat hyötyä mallin teoriapohjan ymmärtämisestä, mutta muuten malli sulautuu osaksi ohjelmointiesimerkkiä vetämättä huomiota puoleensa.

6.5 Lopun kurssin aihealueen käsittely: oppaan luvut 10 ja 11, sekä sanastot

Oppaan kaksi viimeistä lukua ovat oppaan vähiten koherentteja kokonaisuuksia. Tässä vaiheessa kurssia kurssimateriaali käsittelee lähinnä erilaisia kehittyneempiä Javan ominaisuuksia ja näiden kahden luvun tavoitteena on vain käsitellä kyseiset konseptit. Opas sitoo geneerisyyden ja iteraattorit löyhästi yhtenäiseksi kokonaisuudeksi luvussa 10 ja esittelee anonyymit luokat ja sisäluokat luvussa 11.

Oppaan lopusta löytyvät lyhyet loppusanat, joiden tavoitteena on tuoda pieni pala persoonallisuutta muuten asiapainotteiseen tekstiin. Tämän jälkeen oppaassa on kaksi erillistä sanastoa, kuten kappaleessa 5.2: Sanastot mainittiin. LaTeX-kielen mahdollistama helppo viittaussyntaksi ja automaattinen sanastojen luontityökalu mahdollistaa sanastojen automaattisen päivittämisen ja tekstinvälisten linkkien luonnin. Kaikki maininnat sanastotermeistä löytyvät sivunumeroon upotettuina linkkeinä sanastoista ja sisältävät itse linkin kyseiseen sanastokohteeseen. Lisäksi sanastot sisältävät katso myös -viittauksia toisiin, asiaan liittyviin sanastokohteisiin.

7 Pohdintaa

7.1 Oppaan jatkokehitys

Opas on luotu helposti muokattavaksi ja laajennettavaksi. Tämä korostuu oppaan ohjelmistoprojektimaisessa rakenteessa. Iterointi ja pienet muutokset on tehty helpoksi valitsemalla oppaalle oikeantyyppinen tekninen toteutus. Koodiesimerkkejä voi muokata erillään muusta oppaasta ja kaikki oppaan viittaukset niin toisiin kappaleisiin kuin sanastoon on luotu LaTeX-komennoilla, jolloin esimerkiksi kappaleiden otsikoiden muuttaminen on vaivatonta. Täten oppaan jatkokehitys ja hiominen tulevaisuudessa on tehty helpoksi.

Mikäli opasta halutaan joskus kehittää pidemmälle, suosittelisin laajentamisen aloitettavan kappaleessa 6: Oppaan sisällön esittely mainituista pois jätetyistä suunnitelluista aiheista. Erityisesti API-dokumentaation lukemiseen perehtyvä kappale viikolle seitsemän, sekä mahdollinen bonuskappale Git-ohjelmiston ja GitHub-palvelun käytöstä olisivat tärkeitä. Lisäksi oppaaseen voisi kirjoittaa reilusti enemmän esimerkkikoodia ja toimivia ohjelmistoja opastavalla kommentointityylillä. Oppaan teknisen rakenteen parantaminen olisi myös tärkeää jatkokehityksessä. Varmistamalla ettei projektiin kerry teknistä velkaa mahdollistetaan sujuva jatkokehitys myös tulevaisuudessa.

7.2 GitHub issues

Osassa 4.5: Versionhallinnan käyttäminen LaTeX-dokumentin yhteydessä mainittiin GitHub-palvelun käyttämisen mahdollistavan oppaan jatkokehityksessä GitHubin issues -ominaisuuden käytön oppaan pääasiallisena palautekanavana. Issues mahdollistaa repositorion kehitysideoiden scrum-metodologian tyylisinä product backlog item (PBI) -kokonaisuuksina [iss, 2022] [James and Walter, 2010].

Kun oppaan betaversio julkaistiin vuoden 2022 Olio-ohjelmointi -kurssin käyttöön tehtiin samalla päätös rohkaista opiskelijoita ilmoittamaan oppaasta löytyvät kirjoitusvirheet, faktavirheet ja parannusideat tällä systeemillä sen sijaan että palautetta esimerkiksi lähetettäisiin sähköpostilla kurssin vastaavalle opettajalle.

Kehittäjän näkökulmasta GitHub issues tarjoaa hyvän mahdollisuuden keskittää palaute oppaasta ja samalla rytmittää oppaan jatkokehitystä. Opiskelijoiden näkökulmasta työkalu

ei kuitenkaan ilmeisesti ole ollut erityisen helposti lähestyttävä, sillä yksikään opiskelija ei ole tehnyt issueta oppaan käytöstä selkeistä ohjeista ja rohkaisusta huolimatta. Itseasiassa yksi sanallinen opiskelijapalaute oppaasta oli, että opiskelija ei osannut tehdä issueta löytämästään kirjoitusvirheestä, vaan pyysi oppaan tekijää toteuttamaan tämän. Tähän riskiiriitaan olisi mielenkiintoista paneutua tarkemmin ja selvittää muun muassa voitaisiinko oppaan repositorion issuepohjia kehittää helpompikäyttöisiksi.

7.3 Arvio työn onnistumisesta

Työn tutkimuskysymys oli ”*Kuinka parantaa Olio-ohjelmointi -kurssin ohjelmointiopasta?*” jaettuna kahteen alakysymykseen: ”*Millä tekniikalla opas tulisi kasata ja ylläpitää?*” ja ”*Mitä oppaan tulisi sisältää?*”. Oppaan parantaminen toteutettiin luomalla uusi ohjelmointiopas käyttäen LaTeX-taittokieltä, git-ohjelmistoa ja GitHub-palvelua. Oppaan sisällön suhteen tärkein päätös oli Android-API:n käsittelyn tiputtaminen. Android-API on yleensä kurssilaisille kurssin haastavimpia asioita, mutta oppaan laajuutta rajoitti työn luonne kandidaatintyönä.

Kappaleessa 3.4 esitettyjen tavoitteiden pohjalta opas tuntui onnistuneelta. Alustava suullinen palaute kurssilaisilta on vaihdellut varovaisen positiivisen ja huomattavan positiivisen välillä. Lisäksi valittu tekninen toteutustapa saavutti tavoitteensa. Oppaan kehittäminen ja muokkaaminen oli mukavaa ja vaivatonta. Erityisesti esimerkkikoodin jakaminen erilliseen repositorioon ja mutkaton integroidun kehitysympäristön käyttö osana kehitysprosessia Minted-LaTeX-kirjaston ansiosta helpotti ja nopeutti kehitystyötä suunnattomasti. Suosittelisin käytettyä kehitystyöliä lämpimästi kaikkiin tämän tyyppisiin ohjelmointiaiheisiin pidempiin teksteihin, joissa koodin sisällyttäminen tekstiin on tärkeää.

7.4 Jatkotutkimus

Työn toteutuksen ympärille olisi mukava järjestää jatkotutkimusta kahdelta eri kannalta. Aiemmin mainittu päätös käyttää GitHub issues -palvelua pääasiallisena palautekanavana vaikutti Jiraan ja scrumiin töissä tottuneen oppaan pääasiallisen kehittäjän näkökulmasta hyvältä idealta. Kurssilaisilta saatu alustava suullinen palaute kuitenkin on antanut aiheita epäillä järjestelmän näennäisen monimutkaisuuden tai vaan vierauden hankaloittavan palautteen antamista. Tämä eri palautekanavien hyvien ja huonojen puolien vertailu oppaan jatkokehityksessä olisi erittäin mielenkiintoinen aihe tutkimukselle tulevaisuudessa.

Toinen, ehkä ilmeisempi jatkotutkimuksen aihe olisi oppaan julkaisun vaikutus kurssin opettamiseen. Työhön ei nyt sisällytetty minkäänlaista vertailua esimerkiksi vanhan ja uuden ohjelmointioppaan käytön suhteen, tai muutenkaan palautteen keräämistä opiskelijoilta työn jo valmiiksi melko laajan luonteen vuoksi.

8 Yhteenveto työstä

Työn tavoitteena oli tuottaa uusi ohjelmointiopas LUT-yliopiston Olio-ohjelmointi -kurssille. Esitettyä tutkimuskysymystä, ”Kuinka parantaa Olio-ohjelmointi -kurssin kurssiopasta?” lähestyttiin erityisesti oppaan teknisen toteutuksen näkökulmasta vertailemalla Jyväskylän yliopiston TIM-alustaa, LaTeX-taitokielellä kirjoitettua pdf-dokumenttia ja perinteisellä tekstinkäsittelyohjelmalla, esimerkiksi Microsoft Wordilla kirjoitettua pdf-dokumenttia. Näiden välillä päädyttiin LaTeX-kielellä toteutettuun pdf-dokumenttiin, hyödyntäen erityisesti Minted-kirjastoa osana kehitystä.

Oppaan sisältö suunniteltiin kevään 2021 kurssin osallistujille jaetun opiskelijakyselyn, sekä kurssin nykyisen rakenteen perusteella. Opas toteutettiin osana työtä. Oppaan beta-versio valmistui kevään 2022 kurssille, mutta työssä ei toteutettu laajempaa tutkimusta oppaan vaikutuksesta kurssin opetukseen. Oppaan tekninen toteutus LaTeX-taittokielellä toteutettiin helppokäyttöiseksi ja toimivaksi. Oppaan pääasiallisena palautekanavana päätettiin koittaa käyttää GitHubin issues-ominaisuutta.

Työn pohjalta voitaisiin suorittaa ainakin kaksi jatkotutkimusta. Palautekanavan tehokkuuden suhteen on herännyt epäilyjä suullisen palautteen takia ja olisi mielenkiintoista selvittää, onko uudistetusta tavasta antaa palautetta oppaasta enemmän haittaa vai hyötyä oppaan jatkokehityksessä. Toinen tutkittava aihe on oppaan vaikutus kurssin opetukseen ja täten oppaan toteutuksen onnistumisen arviointi. Tämä jätettiin aihealueen rajaamisen vuoksi pois tästä työstä.

VIITTAUKSET

[and, 2021] (2021). Android api deprecated features from version 30 to version 31. https://developer.android.com/sdk/api_diff/31/changes. Accessed: 2022-05-04.

[iss, 2022] (2022). Github issues. <https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues>. Accessed: 2022-05-04.

[int, 2022] (2022). IntelliJ idea. <https://www.jetbrains.com/idea/>. Accessed: 2022-05-04.

[tim, 2022] (2022). The interactive material, jyu. <https://tim.jyu.fi/velp/users/vesal/pedal7/vahemmanonenamman#ZxBESR6k3fTh>. Accessed: 2022-05-04.

[jav, 2022] (2022). Java standard library documentation. <https://docs.oracle.com/javase/8/docs/>. Accessed: 2022-05-04.

[opa, 2022] (2022). Ohjelmointiopas. <https://Github.com/EddieTheCubeHead/LUT-ohjelmoinnin-perusteet-ohjelmointiopas>. Accessed: 2022-05-04.

[esi, 2022] (2022). Ohjelmointioppaan esimerkkikoodit. <https://Github.com/EddieTheCubeHead/OhjelmointiopasEsimerkit>. Accessed: 2022-05-04.

[ove, 2022] (2022). Overleaf. <https://www.overleaf.com/>. Accessed: 2022-05-04.

[sou, 2022] (2022). Sourcetree. <https://www.sourcetreeapp.com/>. Accessed: 2022-05-04.

[tex, 2022] (2022). Texmaker. <https://www.xmlmath.net/texmaker/>. Accessed: 2022-05-04.

[tuf, 2022] (2022). Tufte-latex template. <https://tufte-latex.github.io/tufte-latex/>. Accessed: 2022-05-04.

[April, 2015] April, J. L. B. (2015). Word or latex typesetting: which one is more productive? finally, scientifically assessed. <https://mappingignorance.org/2015/04/06/word-or-latex-typesetting-which-one-is-more-productive-finally-sc>
Online article.

- [Bala and Kaswan, 2014] Bala, M. R. and Kaswan, M. K. K. (2014). Strategy design pattern. *Global Journal of Computer Science and Technology*.
- [Brischoux and Legagneux, 2009] Brischoux, F. and Legagneux, P. (2009). Don't format manuscripts. *The Scientist*, 23(7):24.
- [Browder and Cross, 2018] Browder, R. and Cross, C. (2018). Welcome to overleaf: A brief overview of opportunities. https://vtechworks.lib.vt.edu/bitstream/handle/10919/85838/Welcome_to_Overleaf_web%20version.pdf. Presentation.
- [Caspersen and Bennedsen, 2007] Caspersen, M. E. and Bennedsen, J. (2007). Instructional design of a programming course: a learning theoretic approach. In *Proceedings of the third international workshop on Computing education research*, pages 111–122.
- [Eckel, 2003] Eckel, B. (2003). *Thinking in JAVA*. Prentice Hall Professional.
- [Haines, 2014] Haines, C. (2014). *Understanding the Four Rules of Simple Design*. Leanpub.
- [Hakala et al., 2006] Hakala, T., Nykyri, P., and Sajaniemi, J. (2006). An experiment on the effects of program code highlighting on visual search for local patterns. In *PPIG*, page 10. Citeseer.
- [Hannebauer et al., 2018] Hannebauer, C., Hesenius, M., and Gruhn, V. (2018). Does syntax highlighting help programming novices? *Empirical Software Engineering*, 23(5):2795–2828.
- [Herala et al., 2015] Herala, A., Vanhala, E., and Nikula, U. (2015). Object-oriented programming course revisited. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, pages 23–32.
- [James and Walter, 2010] James, M. and Walter, L. (2010). Scrum reference card. https://www.collab.net/sites/default/files/uploads/CollabNet_scrumreferencecard.pdf. Short manual.
- [Kasurinen and Nikula, 2007a] Kasurinen, J. and Nikula, U. (2007a). Lower dropout rates and better grades through revised course infrastructure. In *Proceedings of the 10th International Conference on Computers and Advanced Technology in Education*, pages 152–157.
- [Kasurinen and Nikula, 2007b] Kasurinen, J. and Nikula, U. (2007b). Revising the first programming course-the second round. *ReflekTori 2007*, pages 92–101.

- [Knutas et al., 2016] Knutas, A., Herala, A., Vanhala, E., and Ikonen, J. (2016). The flipped classroom method: Lessons learned from flipping two programming courses. In *Proceedings of the 17th International Conference on Computer Systems and Technologies 2016*, pages 423–430.
- [Kölling, 1999a] Kölling, M. (1999a). The problem of teaching object-oriented programming, part 1: Languages. *Journal of Object-oriented programming*, 11(8):8–15.
- [Kölling, 1999b] Kölling, M. (1999b). The problem of teaching object-oriented programming, part 2: Environments. *Journal of Object-Oriented Programming*, 11(9):6–12.
- [Koulouri et al., 2014] Koulouri, T., Lauria, S., and Macredie, R. D. (2014). Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education (TOCE)*, 14(4):1–28.
- [León, 2010] León, X. (2010). Introduction to latex. <https://personals.ac.upc.edu/xleon/sodx/LatexIntroduction.pdf>. Short manual.
- [Lyon and Castellanos, 2007] Lyon, D. A. and Castellanos, F. (2007). The parametric singleton design pattern. *Journal of Object Technology*, 6(3).
- [Martin, 2009] Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.
- [Martin and Martin, 2006] Martin, R. C. and Martin, M. (2006). *Agile principles, patterns, and practices in C# (Robert C. Martin)*. Prentice Hall PTR.
- [McCracken et al., 2001] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 125–180.
- [Oyegoke, 2011] Oyegoke, A. (2011). The constructive research approach in project management research. *International Journal of Managing Projects in Business*.
- [Rudolph, 2011] Rudolph, K. (2011). The minted package: Highlighted source code in latex. <https://google-code-archive-downloads.storage.googleapis.com/v2/code.google.com/minted/minted.pdf>. Short manual.
- [Shmallo and Ragonis, 2021] Shmallo, R. and Ragonis, N. (2021). Understanding the “this” reference in object oriented programming: Misconceptions, conceptions, and teaching recommendations. *Education and Information Technologies*, 26(1):733–762.

- [Singh and Hassan, 2015] Singh, H. and Hassan, S. I. (2015). Effect of solid design principles on quality of software: An empirical assessment. *International Journal of Scientific & Engineering Research*, 6(4).
- [Vanhala and Kasurinen, 2018] Vanhala, E. and Kasurinen, J. (2018). Goals and principles for the redesign of a programming course. In *Workshop on PhD Software Engineering Education*. CEUR-WS.
- [Vanhala and Nikula, 2020] Vanhala, E. and Nikula, U. (2020). Python 3–ohjelmointiopas versio 1.2. 1.