TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

OPPIMISPÄIVÄKIRJA

PLA-32820 2018 Mobiiliohjelmointi

Iiro Loukkalahti

Tarkastaja: Mika Saari
Tarkastaja ja aihe hyväksytty
10. tammikuuta 2018

TIIVISTELMÄ

TOIMI KUNTA: Tampereen teknillisen yliopiston opinnäytepohja
Tampereen teknillinen yliopisto
Oppimispäiväkirja, XX sivua, YY liitesivua
Tammikuu 2018
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Ohjelmistotuotanto
Tarkastaja: Mika Saari

Avainsanat: oppimispäiväkirja, mobiiliohjelmointi, java, android, älypuhelin

Tämä dokumentti on oppimispäiväkirja liittyen kurssin ’Mobiiliohjelmointi’ suorittamiseen. Sisältö on jaettu lukuihin harjoituskohtaisesti, jossa jokaiseen on erikseen listattu tekemiset ja suunnittelemiset päiväkohtaisesti.

Tässä pohjassa tiivistelmää varten 2 omaa tekstityyppiä: tunnistiedoille tyyli BibInfo ja tiivistelmätekstille *Abstract*, jossa riviväli on 1.0. Otsikkotyyppi on *Heading (no number)*, joka tekee automaattsesti sivunvaihdon (Page break before). Samaa otsikkotyyppiä käytetään mm. sisällysluettelossa. Lähdeluettelossa on identtinen tyyppi hieman eri nimellä, jolloin se voidaan poimia sisällysluetteloon. Sivunumeroja varten etusivun lopussa pitää olla *Section Break* ja tiivistelmän yläotsakkeen (header) asetus *Link to Previous* pois päältä, ja lisäksi sivunumeron muotoilusta *Start at i* (eikä *Continue*).

ABSTRACT

TOIMIKUNTA: Thesis template of Tampere University of Technology
Tampere University of Technology
Study journal, XX pages, YY Appendix pages
January 2018
Master's Degree Programme in Information Technology
Major: Software Engineering
Examiner: Mika Saari

Keywords: study journal, diary, mobile programming, java, android, smartphone

This document is a required study journal related to course 'Mobiiliohjelmointi'. All exercises have been separated to their own sections with daily documentation. The documentation will contain all steps taken to carry out the tasks from start to finish.

ALKUSANAT

Tämä dokumenttipohja on laadittu TTY:n opinnäytetyöohjeen vuoden 2014 version mukaan edellistä pohjaa muokkaamalla. Työryhmä haluaa kiittää kaikkia ohjeen päivitykseen osallistuneita.

Alkusanoissa esitetään opinnäytetyön tekemiseen liittyvät yleiset tiedot. Tapana on myös esittää kiitokset työn tekemiseen vaikuttaneille henkilöille ja yhteisöille. Alkusanat eivät kuulu arvioinnin piriin, mutta niissä ei silti ole sopivaa moittia tai kritisoida ketään. Alkusanojen pituus on enintään 1 sivu. Alkusanojen lopussa on päivämäärä, jonka jälkeen työhön ei ole enää tehty korjauksia.

Tampereella, 25.8.2014

Toimi Kunta

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

CC-lisenssi	Creative Commons -lisenssi
LaTeX	ladontajärjestelmä tieteelliseen kirjoittamiseen
SI-järjestelmä	ransk. Système international d'unités, kansainvälinen mittayksikköjärjestelmä
TTY	Tampereen teknillinen yliopisto
URL	engl. Uniform Resource Locator, verkkosivun osoite

a	kiihtyvyys
$\mathbf{F}$	voima
m	massa

Työssä käytetyt lyhenteet ja merkinnät määritellään ja selitetään kootusti aakkosjärjestyksessä työn alussa ja kun ne esiintyvät tekstissä ensimmäisen kerran Lyhenteiden kanssa käytetään tällöin sulkeita. Selitetekstin tyyli on tässä *Symbol description*. Tämän sivun lopussa on *Section Break*, jotta sivunumerointi menee oikein. Lisäksi johdannon yläotsakkeen (header) asetus *Link to Previous* on pois päältä, ja lisäksi sivunumeron muotoilusta on valittu *Start at 1* (eikä *Continue*).

.

1. JOHDANTO

Tämä mallipohja liittyy Tampereen teknillisen yliopiston (TTY) opinnäytteen kirjoitusohjeeseen **Virhe. Viitteen lähdettä ei löytynyt.**. Opinnäyte tai raportti koostuu tyypillisesti seuraavista osista:

Nimiölehti
Tiivistelmä
Abstract (englanninkielinen tiivistelmä)
Alkusanat
Sisällys
Lyhenteet ja merkinnät
1. Johdanto
2. Teoreettinen tausta, lähtökohdat tai ongelman asettelu
3. Tutkimusmenetelmät ja aineisto
4. Tulokset ja niiden tarkastelu (mahdollisesti eri luvuissa)
5. Yhteenveto tai päätelmät
Lähteet
Liitteet (eivät pakollisia)

Jokainen yllämainituista osista aloitetaan uudelta sivulta. Osioiden 1–5 nimet ovat tässä ainoastaan esimerkkejä. Käytä työssäsi paremmin sisältöä kuvaavia nimityksiä. Nimiölehdelle tulee yliopiston logo (leveys n. 8 cm), tekijä, työn nimi ja työn tyyppi (diplomi-, kandidaatin-, harjoitustyö). Lisäksi oikeaan alareunaan tulee tarkastajan nimi. Sisällysluetteloon kootaan kaikki sitä seuraavat otsikot, erityisesti numeroidut. Aina siihen ei laiteta osia ennen sisällysluetteloa. MS Wordin automaattinen toiminto löytyy *References > Table of Contents*.

Johdannossa herätetään lukijan mielenkiinto, perehdytetään hänet tutkimuksen aihepiiriin ja jäsennetään tutkimus. Seuraavaksi esitellään opinnäytetyön taustatiedot, jotka ovat välttämättömiä työn ongelman ymmärtämiselle. Toisinaan tässä kuvataan myös käytetyt menetelmät ja –aineisto eli miten tutkimus on toteutettu.

Sen jälkeen esitellään työssä saavutetut tulokset, niiden merkitys, virhelähteet, poikkeamat oletetuista tuloksista ja tulosten luotettavuus. Yhteenveto on työn tärkein osio. Siinä ei enää toisteta yksityiskohtaisen tarkkoja tuloksia, vaan päätulokset kootaan yhteen ja pohditaan niiden merkitystä. Lähdeluettelon antaa kuvan työn teoreettisesta ja empiirisestä pohjasta sekä toimii kirjallisuusluettelona. Siinä esitetään kaikki tarvittavat tiedot kunkin julkaisun löytämiseksi.

Tämän pohjan luvussa 2 käsitellään esityyliin perussäännöt liittyen kuviin, taulukoihin ja matemaattisiin merkintöihin. Luvuissa 3 ja 4 esitellään viittaustekniikat ja lyhyt yhteenveto. Liitteenä on pohjan tekijän huomioita MS Wordin tekstityyleistä.

2. HARJOITUS 1 – TUTUSTUMINEN MOBIILIYM-PÄRISTÖIHIN

Ensimmäisenä tehtävänä on tutustua haluamaansa kiinnostavaan mobiililaitteeseen. Tarkoitus on tutustua sen ominaisuuksiin ja tapoihin, miten laitetta voidaan ohjelmoida.

2.1 Aloitus (11.1.2018)

Aloitin tänään varsinaisen kurssin suorittamisen asentamalla yliopistolta saamillani tunnuksilla Microsoft Office 365 ProPlus-ohjelmiston. Latasin koulun sivuilta valmiin opinnäytetyöpohjan, jota käyttäisin tämän dokumentin runkona. Koitin aivan aluksi editoida sitä Microsoftin Word-ohjelman selainversiolla, siinä kuitenkaan onnistumatta. Päädyin täten asentamaan koko Office-paketin tietokoneelleni.

2.2 Laitteen valinta (12.1.2018)

Valitsin laitteekseni oman älypuhelimeni, Samsung Galaxy S8 Plus. Tämä laite julkaistiin 29. maaliskuuta 2017. Se on Samsungin lippulaivapuhelin, joka edustaa Android-käyttöjärjestelmään pohjautuvia älypuhelimia. Samsungin sivuilta löytää laitteen tarkemmat ominaisuudet ja mm. käyttöoppaan.

Kuva 1. *Samsung Galaxy S8 Plus*

2.3 Käyttöjärjestelmän ominaisuudet (13.1.2018)

Puhelimessa on tällä hetkellä asennettuna Android 7.0 ”Nougat”. Androidin omilla kotisivuilla on kattava esittely ja listaus sen kaikista ominaisuuksista. Androidista voi lukea myös yleispätevän ominaisuuslistauksen Wikipedian artikkelista.

2.4 Ohjelmointi (14.1.2018)

Laitteiden valmistajat sekä Androidin kehittäjä, Google, tarjoaa verkossa kattavat materiaalit ja ohjeet järjestelmän ohjelmointiin. Tämä lisää erityisesti ohjelmien määrää Googlen ja valmistajien omissa ohjelmakaupoissa, mikä taas lisää asiakkaiden kiinnostusta alustaa kohtaan. Materiaalien avulla pääsee nopeasti kehityksen alkuun. Viime kädessä tästä hyötyvät kaikki osapuolet.

Eri valmistajat tuottavat omia räätälöityjä versioitaan Androidista omiin tuotteisiinsa. Google on yksi valmistajista, joka käyttää ”referenssi”-laitteissaan ”puhdasta Androidia”. Esimerkiksi Samsungin laitteiden käyttöjärjestelmän pohjalla käytetään Androidia, mutta Samsung on itse tehnyt siihen mm. korjauksia, mitä virallisessa versiossa ei ole. Räätälöidyssä versiossa voi olla mukana täysin erilaiset oletussovellukset mm. tekstiviesteille, kuville tai vaikkapa internetin selaamiselle. Eri valmistajien tuottamat räätälöidyt käyttöliittymät voivat joskus näyttää jopa täysin eri käyttöjärjestelmiltä. Androidia käytettäessä tähän kaikkeen on mahdollisuus.

Android Developers sivu sisältää Googlen tuottamat viralliset materiaalit, jotka pätevät kaikkiin Android-pohjaisiin käyttöjärjestelmiin. Näitä ohjeita ja materiaaleja voidaan siis soveltaa valmistajasta riippumatta, jos alustan pohjalla on Android.

Samsung Developers on yksi esimerkki valmistaja- ja laitekohtaisista Android-kehitysmateriaaleista. Valmistaja on mm. listannut ohjeita Galaxy-sarjan mobiililaitteiden kameran ohjelmoinnista. Se tarjoaa käytettäväksi mm. Samsung Camera SDK-kirjastoa, jonka kautta voidaan paremmin ohjata ja käyttää laitteen kameraa ja sen ominaisuuksia. Tätä kirjastoa ei mitä luultavammin voi käyttää millään muilla kuin tämän valmistajan Galaxy-sarjan laitteilla.

2.5 Mahdolliset ohjelmointikielet (15.1.2018)

Androidin ohjelmointiin voidaan käyttää monia eri ohjelmointikieliä. Ohessa on katsaus suosituimpiin vaihtoehtoihin.

2.5.1 Java

Virallinen ohjelmointikieli, jolla suurin osa alustan ohjelmista on tehty. Tämä on turvallinen ja hyvä valinta mm. ensimmäisen oman ohjelman tekemiseen.

2.5.2 Kotlin

Melko uusi kieli, jota myös Google tukee. Koodi voidaan kääntää suoraan ajettavaksi Javan virtuaalikoneessa (JVM). Kieli kehitettiin olemaan ”parempi kuin Java”, kuitenkin niin, että yhteensopivuus Javaan säilytettiin. Tällä koitetaan edesauttaa kehittäjien siirtymistä Javasta Kotliniin helpommin. Lisätietoja täältä.

2.5.3 C ja C++

Kaksi vanhempaa ja erittäin käytettyä kieltä. Kielet antavat kehittäjälle vapaammat kädet mm. muistinhallintaan, jolloin laitteista voidaan ottaa ”kaikki mahdollinen mehu irti”. Kieliä voidaan käyttää ”Native Development Kit” (NDK) kautta Android-ohjelman osissa. Nämä osat voivat olla esimerkiksi vanhoja kirjastoja, joita halutaan käyttää mobiiliohjelman kehityksessä. Lisätietoja täältä.

2.5.4 Python ja Bash

Python on melko uusi alustariippumaton ja tulkattava ohjelmointikieli. Osa sen suosiosta perustuu myös sen helppoon syntaksiin, jonka myötä sitä myös suositellaan opeteltavaksi ensimmäisenä ohjelmointikielenä.

Bash on ohjelmointikieleksi miellettävä kieli sekä komentotulkki, jolla voidaan käyttää ja esimerkiksi ohjelmoida tietokonetta. Monissa Linux-jakeluissa Bash on oletuskomentotulkki.

Kumpaakin kieltä voidaan käyttää Androidissa ”Scripting Layer for Android (SL4A)”-avulla. Lisätietoja täältä.

2.5.5 LUA

Kevyt ja kooltaan pieni skriptikieli. Käytetään usein ohjelmien ”sisäisenä” sktriptikielenä (esim. pluginien kielenä jne.). Käytettävä koodi ajetaan LUA:n omassa virtuaalikoneessa. Nykyään kieli on myös erittäin suosittu mobiilipelien kehityksessä.

LUA:aa voidaan käyttää Androidilla “Corona SDK”-sovelluskehityspaketin kautta. Lisätietoja täältä.

2.5.6 HTML5, JavaScript, CSS:

Android-sovellus voidaan myös tehdä käyttäen apuna perinteisiä web-teknologioita. Adoben ”Phonegap”-teknologialla on mahdollista rakentaa mobiiliohjelma verkkosivuna, käyttäen kuitenkin hyödyksi mm. laitteen sensoreita jne. Lisätietoja täältä.

2.5.7 C#

Tällä Microsoftin kehittämällä kielellä on vahva jalansija Windows-työpöytäohjelmien ja verkkopalveluiden kehityksessä. Se on tyypiltään hyvin samantapainen kuin Java. Kieltä voidaan käyttää ”Xamarin”-työkalun avulla, jolla voidaan kääntää sama ja yksi ohjelma monelle eri alustalle (mm. Windows Phone, iOS, Android). Lisätietoja täältä.

2.6 Ohjelmointiin tarvittavat työkalut (18.1.2018)

Developer.com-sivusto on listannut kaikki parhaimmat kehitysympäristöt Android-kehitykseen. Voit lukea koko listan täältä.

Helpoin tapa toteuttaa Android-kehitystä on tehdä se esimerkiksi Windows-tietokoneella. Ohjelmien testaukseen ja ajamiseen voi käyttää oikean mobiililaitteen sijaan myös emulaattoreita. Emulaattori on ohjelma, joka ”matkii” oikeaa laitteen toimintaa. Voimme esimerkiksi asentaa Windowsiin muutamia eri valmistajien emulaattoreita, ja ajaa niitä ”ohjelmina”. Emulaattorit tukevat koodien debuggausta, mutta ne voivat olla erittäin raskaita. Täten nopein ja paras tapa on käyttää testaukseen oikeaa mobiililaitett (mahdollisesti muutamaa erilaista).

2.7 Laitteesta löytyvät ominaisuudet (24.1.2018)

Android Developer-sivulla on materiaalit ja ohjeet sensorien käyttöön. Koko aineiston voi lukea täältä. Materiaalissa oletetaan, että ohjelmistokehitys toteutetaan Java-kielellä, mutta sensoreita voi käyttää myös muilla kielillä.

Android-järjestelmässä ohjelmat joutuvat pyytämään käyttäjältä luvat tiettyihin asioihin. Yksi tällaisista asioista on esimerkiksi sijaintitiedot, jossa käytetään GPS-sensoria. Jos lupaa ei saada, on kehittäjän käsiteltävä tilanne. Ohjelmiin ei tällöin kannata lisätä turhia oikeusvaatimuksia (esim. sijaintitiedolle), jos niille ei ole oikeasti tarvetta. Loppukäyttäjä voi siis rajata ohjelman käytössä olevien API:en, eli rajapintojen, määrää.

3. HARJOITUS 2 – GIT VERSIONHALLINTA

3.1 Uusi repositorio (24.1 ja 9.2.2018 muokattu)

Tein uuden projektin GitHub-palveluun nimellä ”harj2”. Projekti on nähtävillä julkisesti profiilini kautta osoitteessa:

https://github.com/W0lfw00ds/harj2

Valitsin lisäasetuksen, joka lisää repositorioon suoraan oletusarvoisen ”README.md”-tiedoston, jonka sisältö näytetään etusivulla.

3.2 Projektin tuonti omalle koneelle

Kopioin projektin ”Clone or download”-valinnan kautta seuraavan osoitteen:

https://github.com/W0lfw00ds/harj2.git

Avasin tämän jälkeen Windows PowerShell-komentorivin. Ajoin seuraavat komennot:

```
PS C:\Users\iirol> cd C:\var\temp\
PS C:\var\temp> git clone https://github.com/W0lfw00ds/harj2.git
Cloning into 'harj2'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

Siirryin siis aluksi pohjakansioon, jonne haluan projektin kloonata (C:\var\temp). Tämän jälkeen ajoin vain Git-komennon ”clone”, joka hakee ja kloonaa projektin paikalliselle koneelle. Tiedot siirretään siihen kansioon, mikä on valittuna komentorivin työkansioksi.

3.3 Koodin lisäys repositorioon

Avasin sitten vanhan tutun työkalun, Microsoft Notepadin. Loin uuden tiedoston, johon kirjoitin lyhyen ”Hello world”-skriptin PowerShell-kielellä. Tiedoston nimi oli ”hello-world.ps1”. Tallensin tiedoston juuri kloonaamani projektin juureen, kansioon ”C:\var\temp\harj2\”.

Otin sitten PowerShell-komentorivityökalun jälleen esiin. Tällä kertaa ajoin sille seuraavat komennot:

```
PS C:\var\temp> cd harj2
PS C:\var\temp\harj2> git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        helloworld.ps1

nothing added to commit but untracked files present (use "git add" to track)
PS C:\var\temp\harj2> git add *
PS C:\var\temp\harj2> git commit -m "Hello World-skripti lisätty!"
[master d24cdef] Hello World-skripti lisätty!
 1 file changed, 1 insertion(+)
 create mode 100644 helloworld.ps1
PS C:\var\temp\harj2> git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 329 bytes | 329.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/W0lfw00ds/harj2.git
   93a9301..d24cdef  master -> master
```

Vaihdoin siis aluksi projektin kansioon, jonka Git loi kloonauksessa. Tämän jälkeen tarkastin tilanteen, ja Git ilmoitti uudesta tiedostosta, jota ei ole vielä lisätty versionhallintaan. Lisäsin kaikki vielä versionhallinnassa olemattomat tiedostot komennolla ”git add *”, jonka jälkeen loin ensimmäisen commitin. Aivan lopuksi ajoin ”git push”, joka työnsi juuri luomani commitin GitHub-palvelimelle.

Harjoitus 2 oli nyt valmis.

4. HARJOITUS 3

4.1 Android-työkalujen asennus (6.1.2018)

Latasin uusimman version Windows 10 Pro 64bit käyttöjärjestelmälle. Kehitysympäristön asennukseen meni hetki, mutta ilman ongelmia. Asennus latasi automaattisesti puuttuvia komponentteja, ja laittoi asioita kuntoon.

4.2 Uuden projektin luominen (6.2.2018)

Loin uuden Android-projektin, jossa käytin valmista ja tyhjää ”Empty Activity”-pohjaa. Projekti käyttää työkalunaan Gradlea, ja jostakin syystä ympäristö valitti koko Gradlen puuttumisesta. Nopeasti netistä luettuani ymmärsin, että Gradlen pitäisi tulla Android Studion mukana valmiiksi säädettynä. Jouduin asettamaan asetuksista Gradlen ”home”-kansion, joka löytyi Android Studion asennuskansion sisältä. Tämän jälkeen ympäristö herjasi vielä muista ongelmista, mutta se osasi korjata ne itsekseen, lataamalla puuttuvia komponentteja verkosta.

Käytettäessä ”Empty Activity”-pohjaa, on ulkoasuun (activity_main.xml) jo valmiiksi lisätty yksi ”TextView”-elementti, joka tulostaa näytölle tekstin ”Hello World!”. Toisin sanoin, projekti tarvitsi pohjalle ainoastaan sijainnin sekä projektin nimen, ja ensimmäinen ohjelma oli valmiina ajettavaksi!

4.3 Ohjelman testaaminen Android-laitteella (6.2.2018)

Ennen projektin ajamista oikealla laitteella, oli laitteen asetuksista laitettava ”USB”-debuggaus päälle. Tämä onnistui seuraavasti:

Asetukset > Tietoja puhelimesta > Ohjelmiston tiedot

Tämän jälkeen kohtaa ”Koontiversio” oli napautettava sarjassa 7 kertaa. Tämän seurauksena asetuksiin tuli esille aiemmin piilotettu asetusryhmä, ”Sovelluskehittäjien asetukset”. Näistä asetuksista oli laitettava päälle valinta ”USB-virheenkorjaus”. Tämän jälkeen laite oli periaatteessa valmis yhdistettäväksi tietokoneeseen USB-kaapelilla.

Muistin aiemmin, että eri laitteille oli ladattava erillinen USB-ajuri, ennen kuin testattavia ohjelmia voitiin ajaa. Sitä ei kuitenkaan tällä kertaa tarvittu, vaan Android Studio osasi heti listata laitteen painettaessa ”Run..”-komentoa.

Puhelimen ruudulle tuli vielä erillinen valinta, jossa piti antaa oikeus USB-virheenkorjaukseen. Tämän jälkeen laite ja ympäristö oli viimein yhdistetty kokeilua varten.

Ajoin uudelleen ”Run…”-komennon, jonka jälkeen Android Studio avasi ohjelman puhelimeen ensimmäistä kertaa. Hello world!

Android Studio huomautti, että siihen pitää asentaa päivitys, jonka kautta ohjelman voi ”päivittää puhelimeen suoraan”, käyttäen komentoa ”Apply Changes”. Ilmeisesti tämä on nopeampi tapa muokata ja ajaa ohjelmaa uudelleen. En kuitenkaan vielä tässä vaiheessa asentanut pakettia, vaan ajoin ohjelman puhelimeen perinteisellä ”Run”-komennolla.

4.4 Lisäys Git-versionhallintaan (6.2. ja 9.2 muokattu)

Loin uuden projektin GitHubiin, nimellä ”harj3”. Se löytyy osoitteella:

https://github.com/W0lfw00ds/harj3

Siirryin sitten PowerShell-komentorivillä Android Studiolla luotuun projektikansioon, ja ajoin seuraavat Git-komennot:

```
PS C:\var\temp\harj2> cd C:\Users\iirol\StudioProjects
PS C:\Users\iirol\StudioProjects> cd .\Harjoitus3\
PS C:\Users\iirol\StudioProjects\Harjoitus3> echo "# harj3" >> README.md
PS C:\Users\iirol\StudioProjects\Harjoitus3> git init
PS C:\Users\iirol\StudioProjects\Harjoitus3> git status
PS C:\Users\iirol\StudioProjects\Harjoitus3> git add *
PS C:\Users\iirol\StudioProjects\Harjoitus3> git commit -m "eka committi!"
PS C:\Users\iirol\StudioProjects\Harjoitus3> git remote add origin
https://github.com/W0lfw00ds/harj3.git
PS C:\Users\iirol\StudioProjects\Harjoitus3> git push -u origin master
```

Ohjelman koodit oli nyt lisätty versionhallintaan onnistuneesti!

4.5 Versionhallinnan työkalut Android Studiossa

Android Studio itsessään vaatii GitHub-käyttäjätunnusten syöttämisen. Tämä ei kuitenkaan riittänyt, vaan GitHub-pyysi vielä omalla ikkunallaan tunnusten syöttämisen uudelleen. Palveluun oli nyt kuitenkin kirjauduttu Android Studion kautta, jolloin ”Version Control”-välilehti pääsi käytettäväksi.

En käyttänyt näitä työkaluja sen enempää, koska koen että komentorivipohjainen Git on paljon yksinkertaisempi, kuin Android Studion oma käyttöliittymällinen työkalu. Kokeilin tätä työkalua kuitenkin myöhemmin, ja sen napeista saa ajettua samoja toimintoja kuin mitä kirjoitetaan komentorivin kautta (esim. ”add”, ”commit”, ”push” jne). Ulkoasu on kuitenkin logiikkaansa nähden todella monimutkaisesti suunniteltu. Omasta mielestäni paras versionhallinta-työkalu on ollut Eclipse-kehitysympäristössä.

5. HARJOITUS 4 - LASKUKONE

5.1 Uuden projektin aloitus (8.2.2018)

Aloitin työn luomalla projektille uuden repositorion ”harj4” GitHubiin, osoitteeseen:

https://github.com/W0lfw00ds/harj4

5.2 Uusi projekti Android Studiossa (8.2.2018)

Tein uuden projektin Android Studiossa. Annoin projektin nimeksi ”Harjoitus4”, sekä käytin pohjana valmista tyhjää Acitivity-pohjaa. En tehnyt vielä mitään muuta tässä kohtaa, kuin tallensin projektin.

5.3 Lisäys versionhallintaan (8.2.2018)

Olin asentanut Git-ohjelman tietokoneelleni jo aiemmin, joten pääsin suoraan lisäämään projektia versionhallintaan.

Avasin Windows 10-käyttöjärjestelmän mukana tulevan ”Windows PowerShell”-komentorivityökalun. Tämä työkalu toimii miltei samoin kuin vanha tuttu komentokehote-ohjelma, mutta siinä on uusia tuettuja ominaisuuksia, esimerkiksi PowerShell-skriptojen ajaminen.

Siirryin aivan aluksi projektin sisältävään kansioon seuraavasti:

```
PS C:\Users\iirol>
PS C:\Users\iirol> cd .\StudioProjects\Harjoitus4\
```

Sitten ajoin PowerShellilla seuraavat komennot, jotka löytyvät GitHubin "Quick Setup”-sivulta. Tämä koodi luo uuden ”README.md” tiedoston, joka näytetään repositorion etusivulla, sekä tekee ensimmäisen commitin. Tämä osuus ei vielä lisää projektia:

```
PS C:\Users\iirol\StudioProjects\Harjoitus4> echo "# harj4" >> README.md
PS C:\Users\iirol\StudioProjects\Harjoitus4> git init
Initialized empty Git repository in C:/Users/iirol/StudioProjects/Harjo-
itus4/.git/
PS C:\Users\iirol\StudioProjects\Harjoitus4> git add README.md
PS C:\Users\iirol\StudioProjects\Harjoitus4> git commit -m "first commit"
[master (root-commit) beb5255] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
PS C:\Users\iirol\StudioProjects\Harjoitus4> git remote add origin
https://github.com/W0lfw00ds/harj4.git
PS C:\Users\iirol\StudioProjects\Harjoitus4> git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 237 bytes | 237.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
```

```
To https://github.com/W0lfw00ds/harj4.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Lisäsin tämän jälkeen Android Studiossa luodut projektitiedostot versionhallintaan seuraavilla komennoilla:

```
PS C:\Users\iirol\StudioProjects\Harjoitus4> git add *
PS C:\Users\iirol\StudioProjects\Harjoitus4> git commit -m "added files"
PS C:\Users\iirol\StudioProjects\Harjoitus4> git push
```

Nyt projektin kaikki tiedostot oli lisätty, ja ne näkyivät repositorion kotisivuilla.

5.4 Laskuri-ohjelman kehitys (8.2.2018)

Avasin projektin nyt uudelleen Android Studiossa. Aloitin muokkaamalla ensin pohjaohjelmasta pois ylimääräisiä asioita.

5.4.1 Etusivun ulkoasu (8.2.2018)

Tyhjensin ensimmäisen sivun aktiviteetin (activity_main.cml) tiedoston sisältämään vain pohjaelementin, ”ConstraintLayout”. Lisäsin sen jälkeen sen sisälle yhden lapsen ”TableLayout”, jolla oli asetettuna ”ID”. Tämä melko simppeli pohjapaneeli on helppo tapa asetella napit ja muut sisällöt oikeille paikoilleen.

Koska jokainen rivi tulisi sisältämään miltei saman logiikan, olisi ulkoasun voinut rakentaa dynaamisesti, mutta olen oppinut kantapään kautta, että se tekee asioita hankalampia. Tämän takia rakensin ulkoasun perinteiseen tapaan, staattisesti XML-tiedoston sisälle.

Ideana oli luoda jokaiselle eri laskulle oma ”TableRow”, jolla ne rivitettäisiin omille riveilleen. Jokainen laskuoperaatiota tekevä rivi koostui seuraavista elementeistä:

1. **EditText:** normaali syötettävä tekstikenttä, jossa on oletus, ”hint”-, eli vihjeenä arvo ”0”. Tämä arvo katoaa heti, kun käyttäjä kirjoittaa kenttään jotain. Käyttäjän syöte on rajoitettu vain kokonaislukuihin, käyttämällä ominaisuutta ”inputType=’number’”. Tähän täytettiin ensimmäinen luku.
2. **TextView:** pelkkä teksti, jota ei voi valita. Tämä keskitettiin ”textAlignment”-ominaisuuden avulla. Tähän kirjoitettiin operaattorin merkki.
3. **EditText:** toinen samanlainen elementti kuin ensimmäinen. Tähän käyttäjä kirjoittaa toisen luvun.
4. **Button:** nappi, jota painamalla ohjelma suorittaa laskun operaattorin mukaisesti. Tulos asetettiin seuraavaksi esiteltävään elementtiin.
5. **EditText:** normaalista poiketen, tämä elementti on ”disabloitu”, joten siihen kirjoittaminen on estetty. Tekstin valinta ja kopiointi on kuitenkin mahdollista. Tähän listataan laskennan tulos, kun nappia painetaan.

Kaikkien rivien elementtien leveydet asetettiin suhteellisesti toisiinsa, käyttäen ”layout_weight”-ominaisuutta, ja asettamalla normaali leveys ”layout_width=’0dp’”. Tässä käytettiin suhteita: 25%, 5%, 25%, 25%, 20%.

Laskenta-rivien alapuolelle lisättiin vielä erillainen työkalurivi, jossa olivat napit ”Tyhjennä kaikki” ja ”Näytä logi”. Nämä luotiin samanlaisella menetelmällä kuin aiemmat rivit, käyttäen nyt leveyssuhteita 50% ja 50%.

5.5 Logi-sivun ulkoasu (8.2.2018)

Valinnainen lisätehtävä vaati toisen sivun, jossa näytettäisiin kaikki tehdyt laskuoperaatiot.

Aloitin tämän samalla tavalla kuin edellisen Activityn tekemisen. Lisäsin tällä kertaa ainoaksi lapseksi ”TableLayout”-elementin, jolla ei ollut itsellään yhtäkään lasta. Nämä lapset tultaisiin tekemään dynaamisesti.

5.6 Etusivun ohjelma (8.2.2018)

Etusivun ohjelman logiikka sijaitsee ”MainActivity”-luokassa. Aloitin tekemisen, listaamalla privaatit oliot jokaselle ulkoasun elementille (esim. TableLayout, TableRow, Button jne), jotka hain ”findViewById”-metodilla.

Olioiden listauksen jälkeen, asetin mahdolliset nappien kuuntelijat. Jokainen laskuoperaatioita tekevä rivi käyttäisi painettaessa samaa metodia:

```
private void laske(int luku1, int luku2, String operaattori, TextView tulos-
Kentta);
```

Metodin parametrit luetaan sen rivin oikeista kentistä, operaattori ”TextView”-elementin tekstistä, sekä tulosta varten annettiin viite rivin tuloksen listaavaan ”EditText”-elementtiin. Tämä oli helppoa ja mahdollista jo senkin takia, että jokainen rivi oli periaatteessa täysin samanlainen. Ainoastaan näytettävä operaattori vaihtui. Alla esimerkki napin toiminnallisen lisäämisestä, käyttäen apuna anonyymiä luokkaa:

```
this.plus_nappi.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    laske(
      Integer.valueOf(MainActivity.this.plus_luku1.getText().toString()),
      Integer.valueOf(MainActivity.this.plus_luku2.getText().toString()),
      MainActivity.this.plus_teksti.getText().toString(),
      MainActivity.this.plus_tulos
    );
  }
});
```

Työkalurivin napeille luotiin erilliset metodit ”tyhjenna()”, sekä ”naytalogi()”. Toiminnallisuus lisättiin samaan tapaan anonyymejä luokkia käyttäen, kutsuen ”onClick(View view)”-metodin sisällä vain vastaavaa metodia.

5.6.1 ”laske”-metodin logiikka

Metodi lukee parametreista molemmat luvut, sekä merkkijonona käytetyn operaattorin. Se valitsee sitten ”switch”-lohkon sisällä oikean logiikan operaattorin perustella, ja laskee tuloksen uuteen muuttujaan.

Tämän jälkeen haetaan nykyinen päivämäärä muodossa:

```
dd-MM-yyyy  HH:mm:ss
```

Sitten päivämäärän merkkijono ja laskettu tulos yhdistetään, jolloin saadaan seuraavanlainen merkkijono:

```
dd-MM-yyyy  HH:mm:ss: tulos
```

Tämä jono lisätään paikalliseen ”ArrayList<String>”-listaan, jossa on listattuna kaikki tehdyt operaatiot.

Lopuksi laskettu tulos asetetaan parametrina saadun ”EditText”-kentän tekstiksi.

5.6.2 ”tyhjenna”-metodin logiikka

Tämän metodin sisällä käydään läpi kaikki lukujen syöttämiseen luodut ”EditText”-elementit, joiden arvoksi asetetaan tyhjä. Tähän käytettiin apuna olion metodia ”.setText(””);”.

5.6.3 ”naytalogi”-metodin logiikka

Tämä metodi luo uuden ”Intent”-olion, joita käytetään siirryttäessä sivulta toiselle. Tämä on yksi normaaleista tavoista vaihtaa Activity-oliota Androidissa. ”Intent”-olion mukaan laitetaan ”Bundle”-olio, johon voidaan listata erilaisia parametreja. Tässä käytettiin metodia ”.putStringArrayList("logit", this.logi);”, jolla saatiin annettua mukaan ”laske”-metodissa täytetty lista merkkijonoja.

5.7 Logi-sivun ohjelma

Tämän sivun logiikkaan kuului ainoastaan käyttäjän tekemien laskujen tulostus. Logiikka alkaa ylikirjoitetusta metodista:

```
protected void onCreate(Bundle savedInstanceState)
```

Kuten edellisen sivun logiikassa mainittiin, saa tämä sivu avattaessa ”ArrayList<String>”-tyyppisen parametrin, joka sisältää laskuhistorian. Parametri kaivetaan tämän ylikirjoitetun metodin sisällä, hakemalla ”Intent”-olio, jolla sivulle tulliin. Tähän käytettiin seuraavia pätkiä:

```
// Lue lähetetyt logit
this.logit = getIntent().getStringArrayListExtra("logit");

// Luo logeista rivit
this.listaaLogit();
```

Alussa siis haetaan ”Intent”-olio metodilla ”getIntent()”, josta kaivetaan ”Bundle”-olion sisällä lähetetty merkkijono lista metodilla ”getStringArrayListExtra(”logit”)”. Tämän jälkeen lista talletetaan paikalliseen muuttujaan myöhempää käyttöä varten.

5.8 ”listaaLogit”-metodi

Kun rivit ollaan saatu tuotua onnistuneesti, ajetaan sen jälkeen tämä metodi. Metodi toimii kahdella tapaa:

- Tulosta erityisviesti, jos yhtään laskua ei ole vielä laskettu
- Tulosta kaikki laskut omina riveinään näytölle

Yksinkertaisuudessaan katsotaan siis parametrina saadun merkkijono listan koko. Jos lista on tyhjä, tulostetaan erityisviesti. Muussa tapauksessa taas tulostetaan rivit jokaista laskua kohden.

Pientä ulkoasun viilausta ajatellen, logiikkaan lisättiin oma ”boldaaTeksti”-Boolean muuttuja. Tämän muuttujan tila vaihdetaan jokaisen rivin kohdalla joko ”TRUE”- tai ”FALSE”-arvoiseksi vuorotellen. Varsinainen rivi luotiin käyttöliittymälle ”lisaaRivi”-metodilla, josta on lisää tietoja seuraavaksi.

5.9 ”lisaaRivi”-metodi

Tätä yhtä ja samaa metodia käytetään kaikkien rivien listaukseen (eli erityisviestin sekä laskettujen laskujen näyttämiseen käyttöliittymällä). Se luo uuden ”TextView”-olion, joka näyttää riville tulevan tekstin. Jos parametrina tullut ”boldaaTeksti”-muuttuja on arvoa ”TRUE”, muunnetaan teksti vielä lihavoituun muotuun.

Tämä tekstin esittävä olio lisätään uuden ”TableRow”-olion lapseksi, jolla on asetettuna kevyet paksunnukset, joka saa tekstin hieman irti näytön reunoilta.

Loppu viimeksi tämä ”TableRow” olio listataan koko käyttöliittymän pohjalla olevaan ”TableLayout” olioon, jolloin näytölle ilmestyy uusi rivi.

6. HARJOITUS 5-6 – TIETOKANTAA HYÖDYNTÄVÄ OHJELMA

TBA.

7. MOOC-HARJOITUKSET

Lisätään myöhemmin.

7.1 MOOC-Harjoitus 1 -

TBA.

8. LAAJA HARJOITUSTYÖ

8.1 Idea

Ajatuksenani on luoda yksinkertainen valuuttojen kurssien vertailun mahdollistava ohjelma. Käyttäjä valitsisi ensin oletusvaluutan, johon muita valuuttoja voisi verrata.

Käyttäisin tässä apuna avointa valuuttakurssien rajapintaa, jonka kautta koittaisin päivittää ohjelman paikallisen tietokannan (jos mahdollista) verkon kautta. Kurssi haettaisiin ainoastaan kerran päivissä, jonka jälkeen se luettaisiin päivän loppuun asti laitteen tietokannasta. Yksinkertaisuudessaan rajapintaan tehdään kysely, joka palauttaa JSON-tyyppisen listan kaikista sen tarjoamista valuutoista ja niiden arvosta suhteessa johonkin valittuun valuuttaan (esim. $).

Ohjelmaa avattaessa se tarkastaa tietokannan tilanteen, ja hakee mahdolliset päivitykset verkon kautta. Tämän jälkeen se avaa uuden sivun, jossa on listattuna oletus- tai käyttäjän viimeksi valitsema valuutta. Tämän alapuolella on kaikki muut valuutat ja niiden arvo suhteessa käyttäjän valitsemaan valuuttaan.

Käyttäjä voi ainoastaan vaihtaa omaa oletusvaluuttaansa, eikä muita asetuksia ole.