Mika Anttonen

MOBIILIOHJELMOINNIN PERUSOPETUKSEEN SOVELTUVAN OHJELMOINTI-KIELEN VALINTA

MOBIILIOHJELMOINNIN PERUSOPETUKSEEN SOVELTUVAN OHJELMOINTI-KIELEN VALINTA

Mika Anttonen
Opinnäytetyö
Syksy 2015
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Tekijä: Mika Matti Fredrik Anttonen
Opinnäytetyön nimi: Mobiiliohjelmoinnin perusopetukseen soveltuvan ohjelmointikielen valinta
Työn ohjaaja: Lehtori Juha Alakärppä (Oulun ammattikorkeakoulu)
Työn valmistumiskuukausi ja -vuosi: Syksy 2015
Sivumäärä: 38 + 2 liitettä

Tässä opinnäytetyössä tehtiin itsenäinen tutkimustyö. Työn tavoitteena oli määritellä mobiiliohjelmointia aloittelevalle opiskelijalle suositeltavin ohjelmointikieli, jonka avulla pääsee kaikkein helpoimmin alkuun mobiiliohjelmoinnissa.

Työn alussa valittiin kolme yleisimmin käytettyä ohjelmointikieltä sekä käytettiin myös kolmea eri ohjelmointiympäristöä, joihin oli integroituna emulaattorit simuloimaan älypuhelinta tietokoneen näytöllä.

Näiden pohjalta työtä lähdettiin rakentamaan eteenpäin toteuttamalla jokaisella kolmella ohjelmointiympäristöllä omat mobiilisovelluksensa, omine ohjelmointikielineen sekä emulaattoreineen. Lopuksi siirrettiin myös yksi opinnäytetyössä läpikäyty koodiesimerkki fyysiseen laitteeseen ja todettiin sovellus toimivaksi myös oikeassa laitteessa.

Työn päätyttyä saatiin määriteltyä ohjelmointikieli, joka tämän tutkimustyön lopullisena tuloksena on suositeltava ja looginen vaihtoehto ja josta on hyvä jatkaa muihin – ehkä vaativimpiin ohjelmointikieliin.

ASIASANAT: ohjelmointikielet, mobiiliohjelmointi, kehitysympäristö, C#, Java, Objective - C, SDK

ALKULAUSE

Haluan kiittää työni ohjaajaa Juha Alakärppää sekä tekstinohjaaja Tuula Hopeavuorta tuesta ja avusta tehdessäni tätä opinnäytetyötä.

Muutoin opiskeluani ajatellen haluan lausua kiitokseni työnantajaani joustavuudesta työaikojen suhteen niin, että se mahdollisti minulle aktiivisen osallistumisen luennoille sekä luonnollisesti myös kokeisiin.

Oulussa 30.10.2015

Mika Anttonen

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aihe päätettiin aloituspalaverisissa Oulun ammattikorkeakoulun tekniikan ja luonnonvara-alan yksikössä Oulussa syksyllä 2014. Opinnäytetyön aiheena oli mobiiliohjelmoinnin perusopetukseen soveltuvan ohjelmointikielen valinta (liite 1). Työ oli jo heti alkujaan mielenkiintoinen vaikkakin kaikki alkuvalmistelut, kuten materiaalin hankkiminen aiheeseen monine ohjelmointiympäristöineen antoivat heti ymmärtää, että tiedossa tulee olemaan paljon tehtävää, ennen kuin työ kokonaisuudessaan alkaa olla valmiina.

Ohjelmointikielen valinnassa taulukossa 1 esitetyt kriteerit haluttiin perustella kiinnittämällä huomiota kielen lauserakenteisiin, eli kielen haluttiin olevani mahdollisimman selkeä syntaksiltaan. Toisin sanoen kielen opiskelun aloittaminen ilman aikaisempaa kokemusta ohjelmoinnista olisi lauseopiltaan sellaista, jonka perusrakenteet olisi helppo oppia ja sisäistää.

TAULUKKO 1. Tärkeimmät kriteerit ohjelmointikielen valintaan

✓	yksinkertainen
✓	selkeä syntaksiltaan
✓	tuki eri alustoille
✓	hyödyllinen
✓	ohjelmointiympäristö (IDE)

Lisäksi kiinnitettiin huomiota kielen soveltuvuuteen eri laitealustoille, eli voidaanko tehdä sovelluksia useille alustoille samaa kieltä käyttäen. Tämä tulisi lisäämään kielen opiskelun hyödyllisyyttä. Työssä kiinnitettiin huomiota myös ohjelmointiympäristöjen toimivuuteen.

Tutkimussuunnitelman pohjalta toteutettiin sama sovellus kaikissa ympäristöissä eri alustoille, jotta kielten vertailusta saatiin selkeämpää. Tämän opinnäytetyön tarkoituksena on myös herättää mielenkiintoa mobiililaitteiden käyttäjiä kohtaan, eli millaista on tähän alaan liittyvä työ ohjelmoijan näkökulmasta katsoen. Toivottavasti tämä myös antaa myös ideoita ja vinkkejä opettajille, jotka tulevaisuudessa tulevat opettamaan mobiiliohjelmointia eri oppilaitoksissa.

2 ENSIMMÄISIÄ OHJELMOINTIKIELIÄ

SHORT CODE

Mitä ilmeisimmin ensimmäinen toteutettu korkean tason ohjelmointikieli oli John Mauchlyn suunnittelema SHORT CODE, joka ilmestyi vuonna 1949. Korkean tason ohjelmointikielellä tarkoitetaan kieltä, joka koodataan käyttäen ihmiselle tuttuja termejä, jonka jälkeen koodi käännetään tietokoneen ymmärtämään muotoon ns. konekielelle. SHORT CODE oli ensimmäinen tietokonekieli, jolla ohjattiin elektronisia laitteita, ja se edellytti ohjelmoijaa vaihtamaan tilat 0 ja 1 käsin. Toisin sanoen jännitteitä jouduttiin vaihtelemaan 0 ja 5 voltin välillä todennäköisesti yksinkertaisten painokytkinten avulla. Kuitenkin se oli ensimmäinen askel nykyaikaisiin ohjelmointikieliin. (1.)

FORTRAN

Ensimmäinen varsinaisen läpimurron saavuttanut korkean tason ohjelmointikieli oli FORTRAN, joka tulee sanoista FORmula TRANslator. Kielen kehitti IBM:llä työskentelevä John Backus työryhmineen vuosien 1954 ja 1957 välisenä aikana. FORTRANin merkitys ohjelmointikielten kehityksessä onkin merkittävä koska havaittiin, että korkean tason ohjelmointikieliä ylipäätään mahdollista toteuttaa. Esimerkkinä mainittakoon monelle tuttu numeeriseen laskentaan tarkoitettu MATLAB-ohjelmisto, jonka ensimmäinen versio on kirjoitettu FORTRAN-ohjelmointikielellä. FORTRANin koodin sanotaankin olevan nopeaa sekä selkeälukuista ja kieltä käytetään edelleen kehittyneenä muunnelmana laajasti sen alkuperäiseen tarkoitukseen - eli tieteelliseen ja numeeriseen ohjelmointiin. (2.)

COBOL

COBOL (Common Business Oriented Language) on kirjoitettu 1950-luvulla Pentagonissa eli Yhdysvaltain Puolustusministeriössä lähinnä yrityskäyttöön sekä hallinnollisiin tietojärjestelmiin tarkoitettu ohjelmointikieli, jota käytetään vielä tänäkin päivänä esimerkiksi rahoitusalalla. Ehkä enemmän kuin missään muussa ohjelmointikielessä COBOLin syntaksi muistuttaa paljon englannin kieltä, josta voidaan jo helposti päätellä olevan kyseessä korkean tason ohjelmointikieli. (3.)

ALGOL

ALGOL (ALGOtrithmic Language) kehitettiin 1950-luvun loppuvuosina lähinnä tieteelliseen laskentaan. Kielen syntaksissa on huomattavissa samankaltaisuutta kuin vielä nykyäänkin monien muiden

käytössä olevien ohjelmointikielten lauserakenteissa kuten Pascal ja C, jotka ovat osaksi periytyneet ALGOLista. ALGOLin osalta voidaankin puhua ensimmäisestä korkean tason ohjelmointikielestä, jossa ohjelmakoodin kirjoittaminen alkoi jo selkeämmin etääntyä varsinaisesta konekielestä ihmislähemmäksi. (4.)

LISP

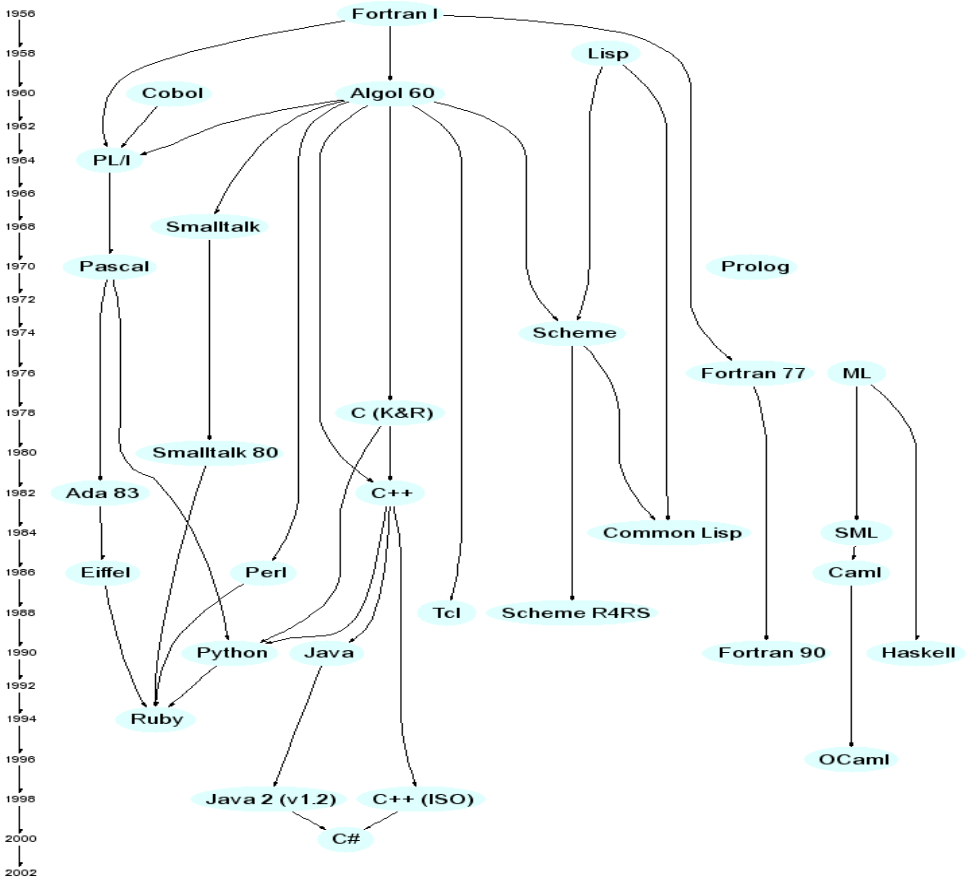
LISPin ensimmäinen versio kirjoitettiin vuonna 1958. Kieltä hyödynnetään myös tietokoneavusteisessa suunnittelussa. Esimerkiksi AutoCAD-ohjelmistoissa on mahdollista käyttää AutoLISP-ominaisuutta, joka helpottaa monimutkaisten mallinnusten piirtämistä. LISP oli ensimmäinen ohjelmointikieli, jossa oli ns. roskienkeruuominaisuus, kuten nykyisin esimerkiksi Javassa. (5; 6.)

OHJELMOINTIKIELTEN KEHITYS

- 1949
  - Ohjelmointi tapahtui reikäkorteilla. Short Code
- 1950 - 1967
  - Ensimmäiset modernit ohjelmointikielet
  - LISP, COBOL, FORTRAN, ALGOL
- 1967 - 1978
  - Olio-ohjelmointi, logiikkaohjelmointi, funktionaalinen ohjelmointi
  - Simula, Smalltalk, C, Prolog, ML, SQL (tietokantahaut)
- 1980-luku
  - Yhtenäistämistä, standardisointia
  - C++, Ada, Perl, Eiffel, FL
- 1990-luku
  - Nopea ohjelmointi, Internet-ohjelmat
  - Haskell, Python, Java, Ruby, PHP
- 2000-luku
  - C#, Swift.

(7.)

Kielikartta ohjelmointikielten kehityksestä on nähtävissä kuvassa 1.



KUVA 1. Kielikartta (1)

3 TYÖSSÄ KÄYTETYT OHJELMOINTIKIELET

3.1 Java

Java on tunnetuimpia ja käytetyimpiä ohjelmointikieliä tänä päivänä ympäri maailmaa. Java-kielellä voidaan kirjoittaa ohjelmia useimpiin tunnettuihin ympäristöihin kuten Windows-, Linux- ja Mac-ympäristöt. Kaikkiin näihin ympäristöihin on myös saatavilla työkalut Java-ohjelmistokehitystyötä varten. Mobiilipuolella paljon suositulle Android-alustalle tarkoitetut sovellukset kirjoitetaan käytännössä pelkästään Javalla, mikä kertoo jo paljon kielen tärkeydestä tänä päivänä. (8.)

3.2 C#

Microsoftin kehittämä C# (”C-sharp”) on pelkästään Windows-sovellusten ohjelmointiin soveltuva ohjelmointikieli. Toisin sanoen esimerkiksi Windows Phone -puhelimien sovellukset voidaan kirjoittaa ainoastaan C#-kielellä. C#:n syntaksissa on huomattavissa piirteitä sekä Java- että C++-kielistä. (8.)

3.3 Objective-C

Applen Objective-C muistuttaa paljon normaalia C-kieltä, mutta sisältää kuitenkin joitain lisäominaisuuksia kuten oliomallin. Objective-C:tä käytetään Applen iOS-laitteiden sovelluskehityksessä kuten iPhone-puhelimet, iPod Touch -musiikkisoittimet ja iPad-taulutietokoneet. Applen tarkoituksena on kuitenkin korvata lähitulevaisuudessa Objective-C kokonaan uudella Swift-ohjelmointikielellä. (9; 10.)

3.4 HTML-kieli

HTML on pääasiassa web-sivujen koodaamiseen tarkoitettu kieli. HTML:llä vaikkapa tekstin rakennetta voidaan merkitä niin, että tietty osa tekstissä on otsikkoon kuuluvaa ja toinen varsinaista tekstiä. Tekstin fonttikokoa voidaan vaihdella sekä sivuille voidaan koodata ns. hyperlinkkejä toisille sivuille siirtymistä varten jne. HTML-koodiin kuuluu aina ns. header-osa, johon kirjoitetaan muun muassa sivun otsikko sekä viittaukset mahdollisiin aputiedostoihin, kuten dokumentin ulkoasuun tarvittavat tiedostot. HTML:n uusin versio on nimeltään HTML5. (11.)

3.5 JavaScript

JavaScript on skriptikieli eli se perustuu komentosarjojen käyttöön. Kieltä käytetään suurimmaksi osaksi apuna verkkosivujen luomisessa. Jos halutaan web-sivulle esimerkiksi lisätä kellonaika ja päivämäärä näkyville, niin tämä on lyhyen JavaScript-koodin avulla mahdollista toteuttaa. Koodi voidaan sijoittaa HTML-dokumentin johonkin haluttuun osaan tai vaihtoehtoisesti kokonaan omaan tiedostoonsa. (12.)

3.6 CSS

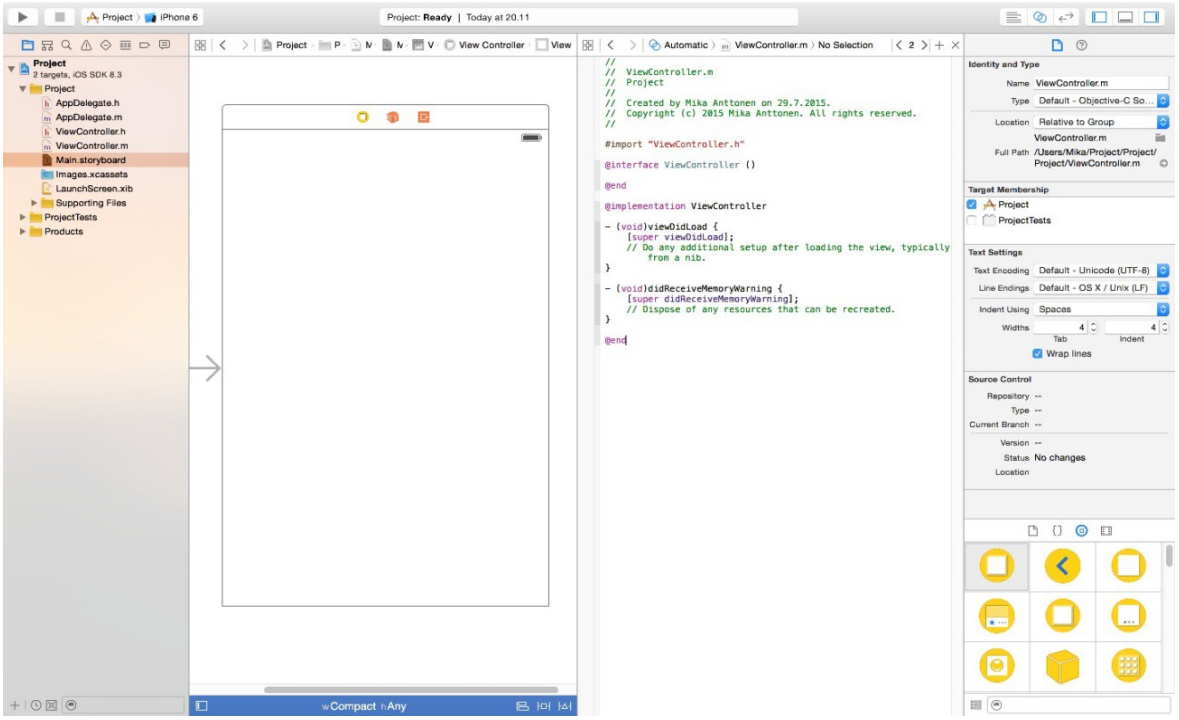
CSS:ää (Cascading Style Sheets) käytetään yleensä HTML-sivun tyyliohjekielenä, jota nimensä mukaisesti käytetään web-sivuilla määrittelemässä värejä, elementtien kokoa ja ulkonäköä. Myös CSS-koodi on suositeltavaa kirjoittaa omaan tiedostoonsa, jotta sen myöhemmät mahdolliset muutokset olisi selkeämpi tarvittaessa toteuttaa. Silloin luonnollisesti CSS-tiedostoonkin on annettava viittaus HTML-dokumentissa. CSS3 on viimeisin versio CSS-tyyliohjekielestä, johon on lisätty paljon uusia ominaisuuksia kuten responsiivisuus, jolla saadaan web-sivu paremmin skaalautumaan vaikkapa puhelimen pienelle näytölle. (13.)

4 IDE-YMPÄRISTÖT

IDE (integradet development environment) on ohjelmointiympäristö, joka sisältää vähintään tekstieditorin koodin kirjoittamista varten sekä kääntäjän, jolla ohjelmoijan kirjoittama lähdekoodi saadaan tietokoneen ymmärtämään muotoon.

4.1 XCode

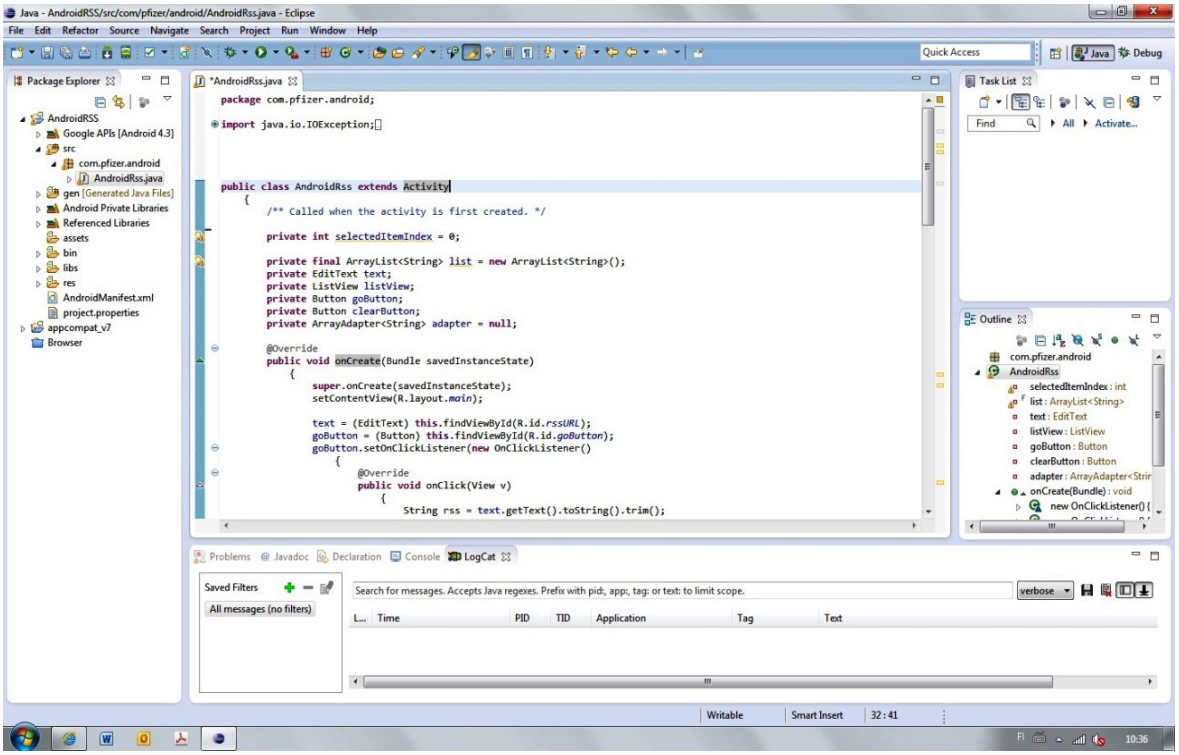
XCode on Applen OS X 10.10:lle tai myöhemmille käyttöjärjestelmäversioille suunnattu ohjelmointityökalu ja sisältää kaiken tarvittavan kehitystyötä varten, kuten iOS-simulaattorin iPhonelle ja iPadille sekä editorin ja kääntäjän (kuva 2). XCoden avulla voidaan kehittää sovelluksia Mac-, iPhone- ja iPad-alustoille. Ohjelmointi tapahtuu joko Objective-C:llä tai Applen uudella Swift-kielellä. OS X käyttäjillä on mahdollista ladata ohjelma Applen iOS Dev Centeristä Mac -tietokoneeseen. (14.)



KUVA 2. XCode-ohjelmointiympäristö

4.2 Eclipse

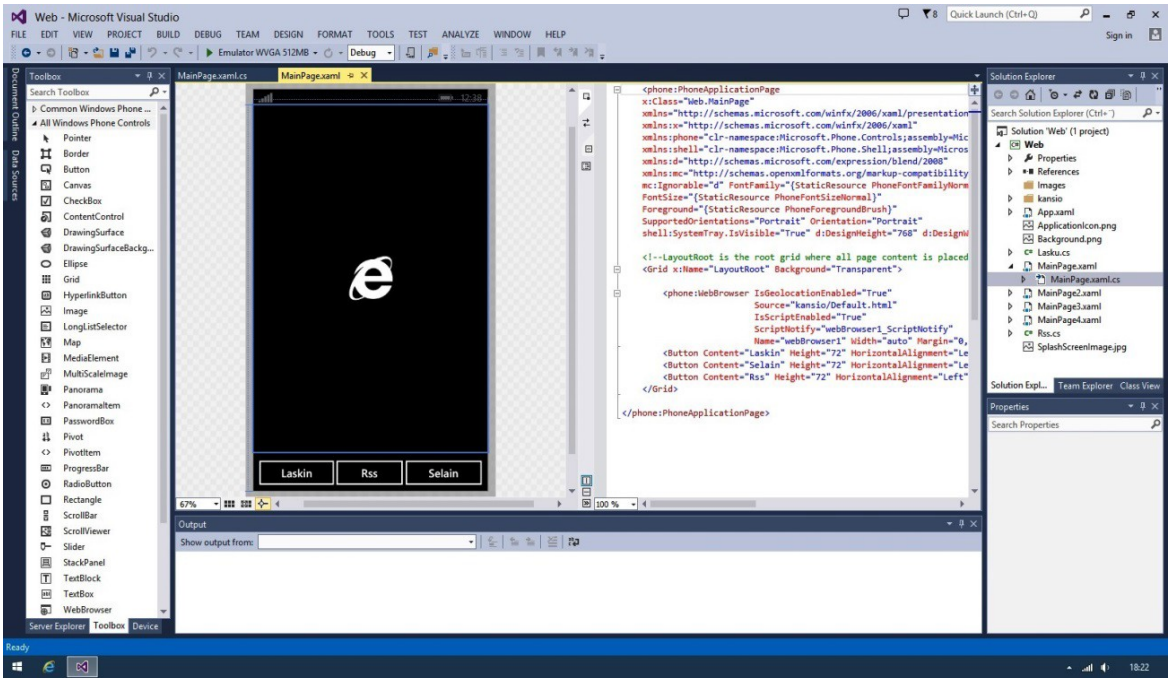
Eclipsen kehitysympäristöllä (kuva 3) voidaan tehdä sovelluksia Java sekä C/C++-kielillä esimerkiksi Android-laitteille. Alustoina voidaan käyttää esimerkiksi MAC OS X- tai Windows-alustaa. Eclipse IDE käyttää kuitenkin omaa käyttöliittymäkirjastoa, joten täysin alustariippumaton se ei ole. Kehitysympäristöstä on saatavilla useita eri jakelupaketteja kehittäjien tarpeen mukaan. (15.)



KUVA 3. Eclipse

4.3 Visual Studio

Microsoft Visual Studio (kuva 4) on Microsoftin kehittämä ohjelmankehitysympäristö, jossa voi käyttää useita ohjelmointikieliä, kuten Visual Basicia, C++:aa, C#:a ja F#:a. Ympäristössä voidaan tehdä esimerkiksi Windows-työpöytä-, web- ja mobiilisovelluksia mutta vain Windows-alustoille, kuten Windows Phonelle. Visual Studiolla on myös mahdollista luoda mobiilisovelluksia Microsoftin Azure-pilvipalveluun. (16.)



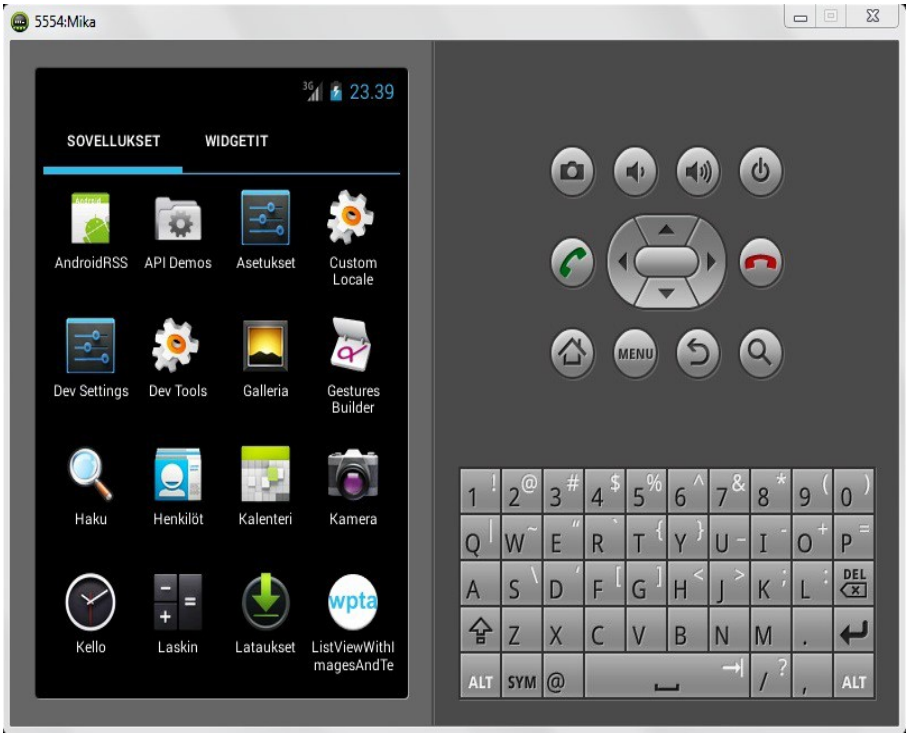
KUVA 4. Microsoft Visual Studio

5 SDK-EMULAATTORIT JA -SIMULAATTORIT

SDK (Software Development Kit) ohjelmistokehitystyökaluun sisältyy emulaattori tai simulaattori, joka on eräänlainen tietokoneohjelma. SDK-ohjelmistokehitystyökalu sisältää rajapinnan, joka voi kommunikoida toisen tietokoneohjelman kanssa, kuten tässä työssä käytetyt ohjelmointiympäristöt. Emulaattorilla voidaan myös jäljitellä oikean laitteen ominaisuuksia, kuitenkaan soittaminen ja tekstiviestien lähettäminen ei ymmärrettävästi ole mahdollista toteuttaa tämän avulla. Työkalua käytetäänkin pääsääntöisesti ohjelmistokehittäjien tekemien sovellusten testaamiseen ja toimivaksi toteamiseen, ennen kuin ne ajetaan oikeaan laitteeseen.

5.1 Android

Android-emulaattori on työkalu, joka emuloi Android-laitetta kuten älypuhelinta. Androidin emulaattoria käytetään usein Eclipse-ympäristössä. (Kuva 5.)



KUVA 5. Android-emulaattori

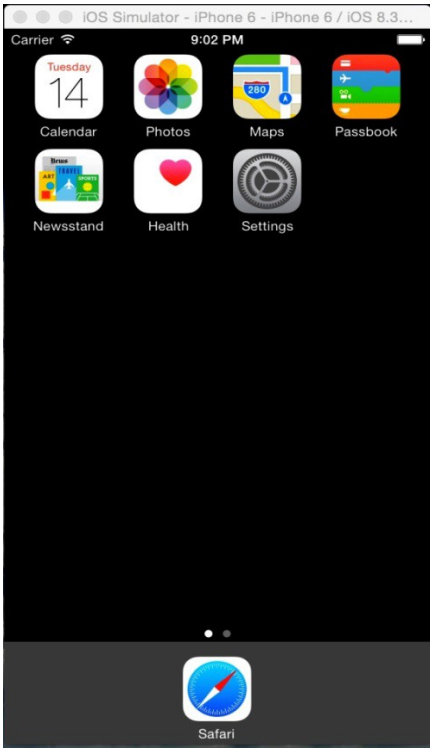
5.2 Windows Phone 8.1

Windows Phone -emulaattori sisältyy WP 8.1 SDK:hon. Emulaattorin toimintaedellytyksenä on Windows 8 Pro -käyttöjärjestelmäversio PC-tietokoneessa. Lisäksi laitteistopuolelta vaaditaan uudehko PC-tietokone, jonka prosessorissa on ns. Hyper-V -tuki. Tämä tulee kytkeä päälle PC:n bios-asetuksissa, sillä muutoin Windows Phone 8.1 -emulaattorin käyttö ei ole mahdollista. (Kuva 6.)

KUVA 6. Windows Phone 8.1 -emulaattori

5.3 Apple iPhone 6

Applen iPhone 6 -simulaattorilla voidaan simuloida Objective-C:llä sekä uudella Swift-kielellä tehtyjä sovelluksia. Kuvassa 7 esitetty simulaattori sisältyy XCode-kehitysympäristöön.



KUVA 7. Apple iPhone 6 -simulaattori

6 MOBIILISOVELLUSTEN TOTEUTUS JA TESTAUS ERI OHJELMOINTIKIELILLÄ SEKÄ YMPÄRISTÖILLÄ

Ohjelmointia, oli se sitten mobiili- tai mitä tahansa ohjelmointia, ei opi mitenkään muuten kun itse konkreettisesti tekemällä, riippumatta kielestä, jolla aloitetaan. Lisäksi tarvitaan hyvä käyttöjärjestelmä ja hyvä kehitysympäristö, jossa sovelluksia tullaan tekemään. Ainoastaan mielekäs ohjelmointikieli ei siis riitä.

Tämän opinnäytetyön tueksi haettiin mobiiliohjelmointiin liittyviä aikaisempia tutkimuksia sekä kielistä sekä ohjelmointiympäristöistä. Nämä haettiin verkosta ja web-osoitteet lisättiin lähdeluetteloon (17; 18). Lisäksi tehtiin haastatteluja henkilöiltä joilla on kokemusta aiheeseen liittyen. Työssä saadut omat näkemykset ovat verrattavissa tehtyihin haastatteluihin suurimmaksi osaksi ehkä siltä osin, mitä ohjelmointikieltä ja -ympäristöä itse pitää miellyttävimpänä käyttää. Haastattelujen keskeisimmät asiat ovat luettavissa alapuolella lähdemerkintöineen.

”Mielestäni paras valinta on se setup joka on itselle kaikkein tutuin, sekä kielen että IDE:n suhteen. Jos tyhjästä lähdetään liikkeelle niin sitten joku generatiivinen ympäristö jossa itse ei vielä koodata vaan piirrellään graafisesti.” (19.)

”Ohjelmoimaan oppiminen on yhtä helppoa tai yhtä vaikeaa riippumatta niinkään kielestä jolla aloitetaan, oli se sitten Java tai C#. Microsoftin Visual Studio on ohjelmointiympäristöistä käytettävyydeltään kaikkein miellyttävin. Applen Objective-C kielestä ei ole kokemusta.” (20.)

”Minulla on kokemusta lähinnä Java ME:stä, joka ei enää ole ajankohtainen. Näin ollen en voi hyvin perustellen vertailla noita nykyisiä Android, Windows Phone ja iOS-alustojen työkaluja. Noissa kielissä ja niiden kirjastoissa ei ole sellaisia eroja, että ne tekisivät jonkin niistä toisia helpommaksi tai vaikeammaksi perusopetuksen kannalta. Kielet kuuluvat samaan kieliperheeseen, kirjastot sisältävät modernin mobiiliympäristön edellyttämät rajapinnat ja ohjelmointityökalut sisältävät samat toiminnot vaikka hiukan eri tavoin toteutettuina.

Mobiiliohjelmoinnin perusopetuksen ongelma, Java ME-kurssien kokemuksella, on aivan sama kuin kaikessa ohjelmoinnin perusopetuksessa. Se ei ole työkaluissa eikä kielissä tai kirjastoissa. Suurin ongelma on, että aloittavan ohjelmoijan on (1) vaikea hahmottaa, mistä annetussa tehtävässä on kyse, (2) kehitellä annetulle tehtävälle ratkaisumalli, joka on ohjelmoitavissa, (3) hallita nykyaikainen ohjelmointiympäristö ja kielen sekä kirjastojen sisällöt niin, että voi niitä hyödyntää lopulta ratkaisun ohjelmoinnissa, ja lopuksi, (4) pystyä lukemaan itse kirjoittamaansa ohjelmaa (ja toisten tekemiä) niin, että osaa päätellä täsmällisesti, mitä ohjelma tekee.” (21.)

Testausvaiheeseen käytettiin kolmea eri ohjelmointikieltä sekä asennettiin tarvittavat ohjelmointiympäristöt (liite 2). Tähän opinnäytetyön kirjalliseen osuuteen sisällytettiin vain keskeisimmät koodausvaiheet esimerkkisovelluksista. Havainnollisuuden vuoksi kaikissa ympäristöissä malliesimerkkinä toteutettiin peruskäyttöön soveltuva nelilaskin, jolla voitiin suorittaa tyypillisimmät laskutoimitukset. Taulukkoon 2 on lueteltu testausvaiheissa käytetyt kielet sekä muut tarvittavat työkalut.

TAULUKKO 2. Testausvaiheiden kielet ja työkalut

Kieli	IDE	SDK
Java	Eclipse	Android
C#	Microsoft Visual Studio	Windows 8.1
Objective-C	XCode	iOS

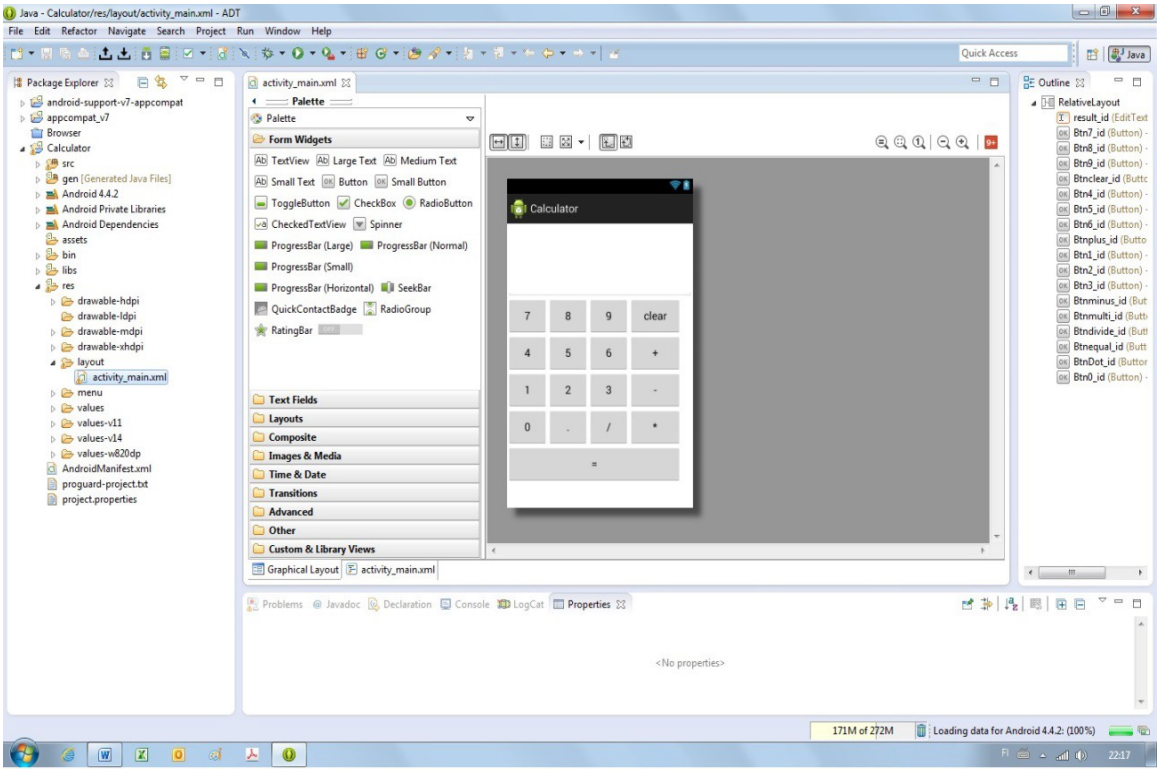
6.1 Java ja Eclipse

Ensimmäisenä lähdettiin luomaan Android-sovellusta Java-ohjelmointikielellä. Sovellus toteutettiin Eclipse-ympäristössä Androidin 4.4.2 KitKat -emulaattoria käyttäen. Javalla voidaan Androidin lisäksi tehdä ohjelmia useille eri alustoille kuten Windows Phonelle.

6.1.1 Graphical Layout

Projektin luomisen jälkeen ohjelmointi aloitettiin avaamalla activity_main-tiedosto, jossa Graphical layout -välilehdellä määriteltiin muun muassa tekstikenttä numeroiden ja merkkien syöttämistä varten. Lisäksi tarvittiin luonnollisesti myös painikkeet laskimelle.

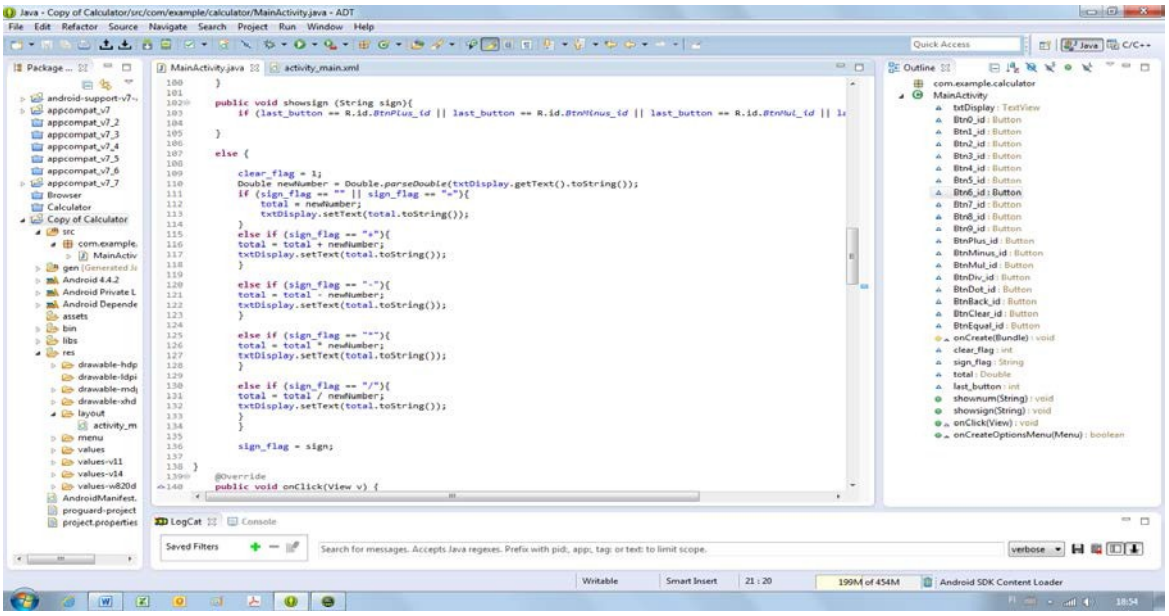
Tarvitut komponentit saatiin Palette-valikosta vetämällä layoutiin, tämän jälkeen ne nimettiin sopiviksi katsotuilla nimillä, sekä määriteltiin jokaiselle komponentille omat id-tunnuksensa varsinaista Java-koodia varten. Tarvittaessa käyttöliittymän komponentit olisi voitu kirjoittaa suoraan varsinaiseen activity_main.xml-tiedostoon. (Kuva 8.)



KUVA 8. activity_main.xml

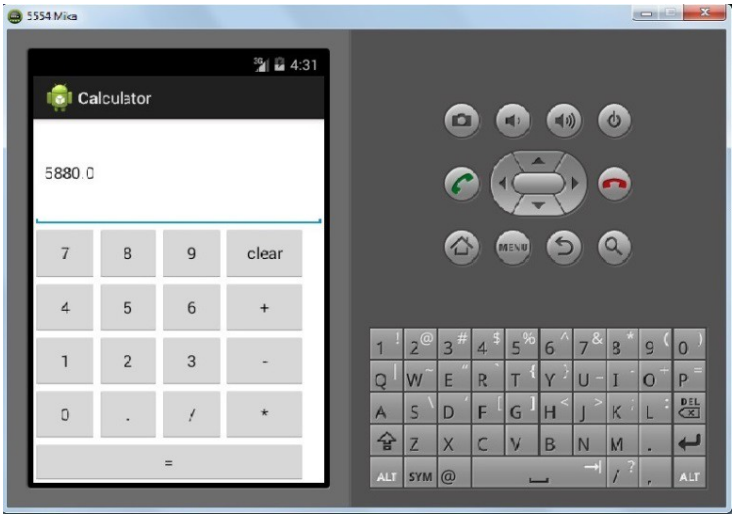
6.1.2 MainActivity-luokka

Sovelluksen ohjelmointia jatkettiin kirjoittamalla pääohjelmaan eli MainActivity-luokkaan varsinainen Java-koodi laskimen toiminnalle. Laskimen painikkeet kirjoitettiin metodirunkoineen vastaamaan niille activity_main-tiedostossa määriteltyjä toimintoja sekä kirjoittamalla laskuoperaattorit if-else-rakenteen sisään, jotta haluttu laskuoperaatio voitiin valita kolmen muun joukosta. Koodi kokonaisuudessaan kirjoitettiin pääohjelmaan, joka samalla toimii Javassa myös luokkana. (Kuva 9.)



KUVA 9. MainActivity-luokka

Java-sovellus testattiin ja todettiin toimivaksi Android-emulaattorissa (kuva 10).



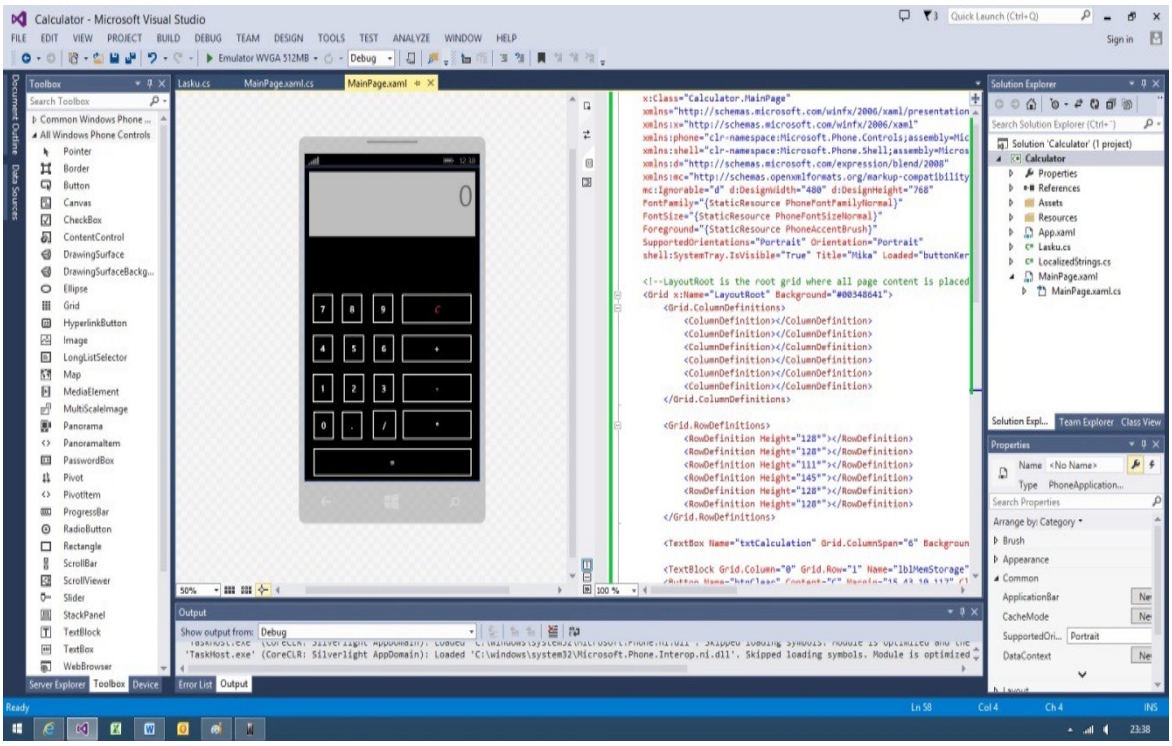
KUVA 10. Android Calculator

6.2 C# ja Microsoft Visual Studio

Toisessa testausosuudessa tehtiin sovellus Windows Phonelle Microsoft Visual Studio 2013 -kehitysympäristöä hyödyntämällä. Ohjelmointikielenä käytettiin tässä tapauksessa ainoata vaihtoehtoa eli C#:a.

6.2.1 MainPage

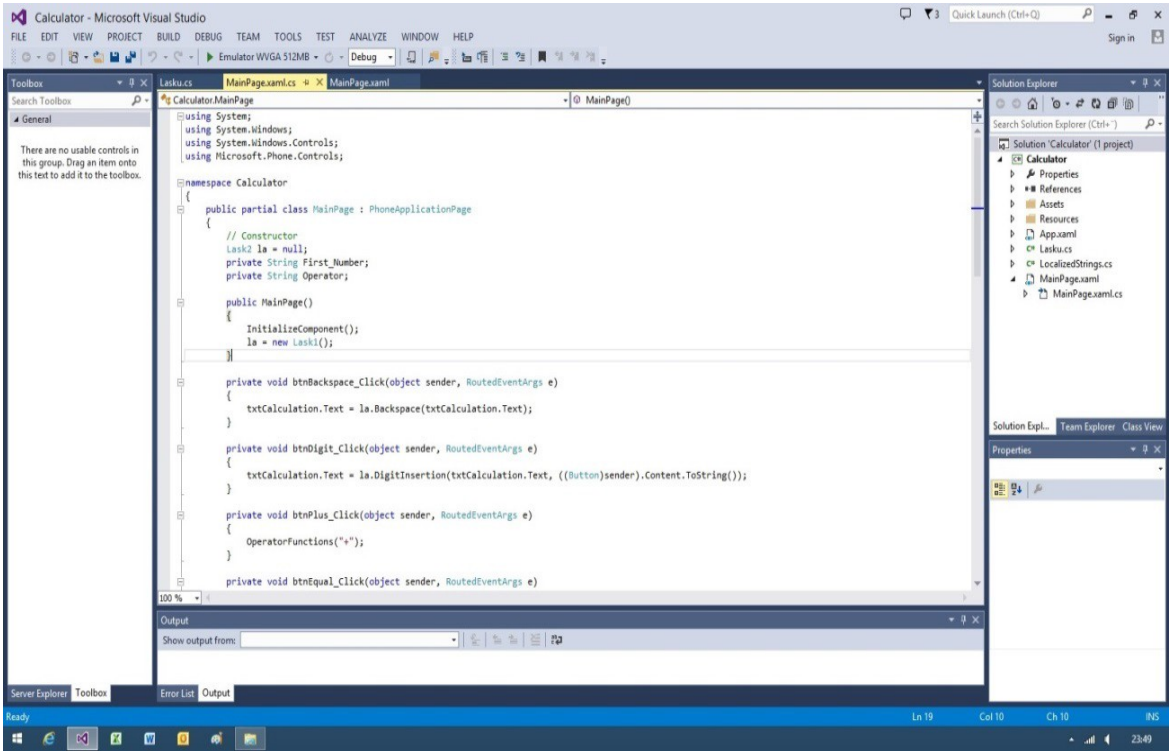
Projekti aloitettiin periaatteessa samalla tavoin kuin Eclipsessäkin eli valittiin nyt nimeltään MainPage.xaml-välilehti, jossa siirrettiin Toolbox-ikkunasta kaikki tarvitut painikkeet haluttuihin kohtiin. Lisäksi ikkunasta tarvittiin myös TextBox-komponentti toimimaan näyttönä laskimelle. Komponenteille annettiin halutut nimet sekä symbolit Properties-ikkunassa. Tämän jälkeen kaksoisklikkaamalla haluttuja komponentteja metodirungot generoituvat valmiiksi varsinaiseen C#-koodiin. Edellisessä Java-esimerkissä metodirungot jouduttiin kirjoittamaan käsin. (Kuva 11.)



KUVA 11. MainPage.xaml

6.2.2 MainPage.cs

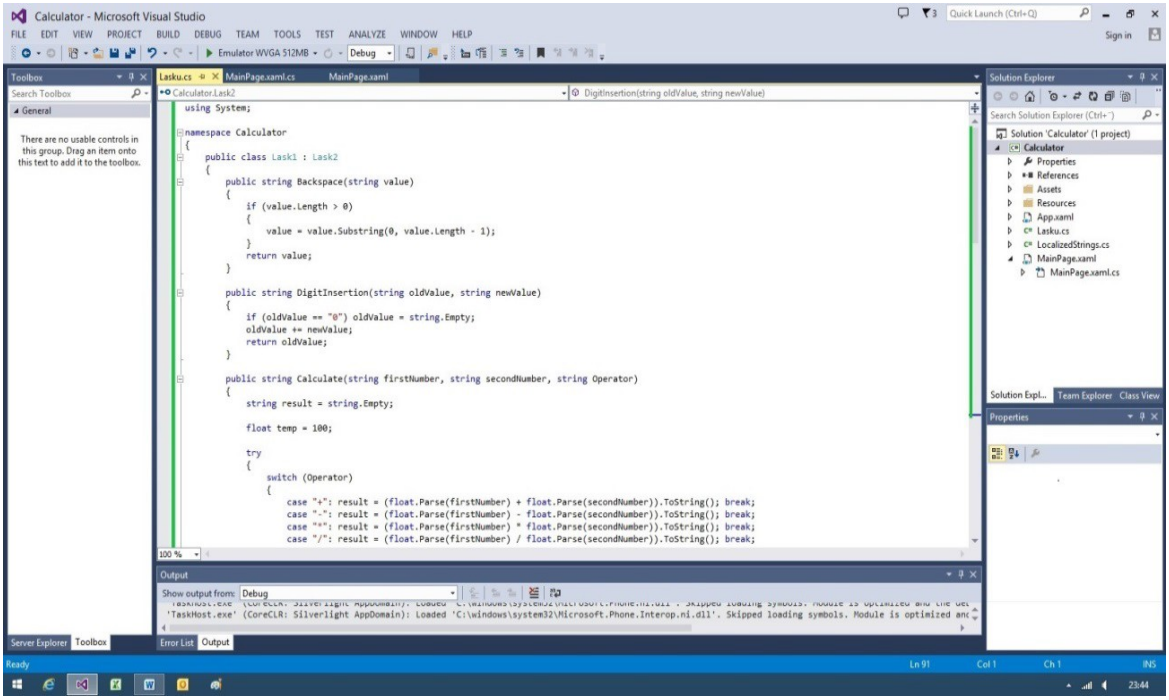
Seuraavaksi siirryttiin MainPage.cs-välilehdelle, johon luotiin muun muassa muodostinfunktio sekä luokkaolio. Lisäksi kirjoitettiin aikaisemmin luotujen metodirunkojen sisään tarvitut toiminnot painikkeita varten. (Kuva 12.)



KUVA 12. MainPage.cs

6.2.3 Lasku.cs

Viimeisessä vaiheessa projektiin lisättiin tiedosto, joka nimettiin Lasku.cs-tiedostoksi. Tähän tiedostoon koodattiin kaksi luokkaa varsinaisten laskuoperaatioiden suorittamiseksi sovelluksessa. Koodissa käytettiin nyt switch case -valintarakennetta tietyn laskuoperaation valintaan. (Kuva 13.)



KUVA 13. Lasku.cs

Lopuksi C#-sovellus ajettiin Windows Phone 8.1 -emulaattoriin ja todettiin toimivaksi (kuva 14).

KUVA 14. Windows Phone Calculator

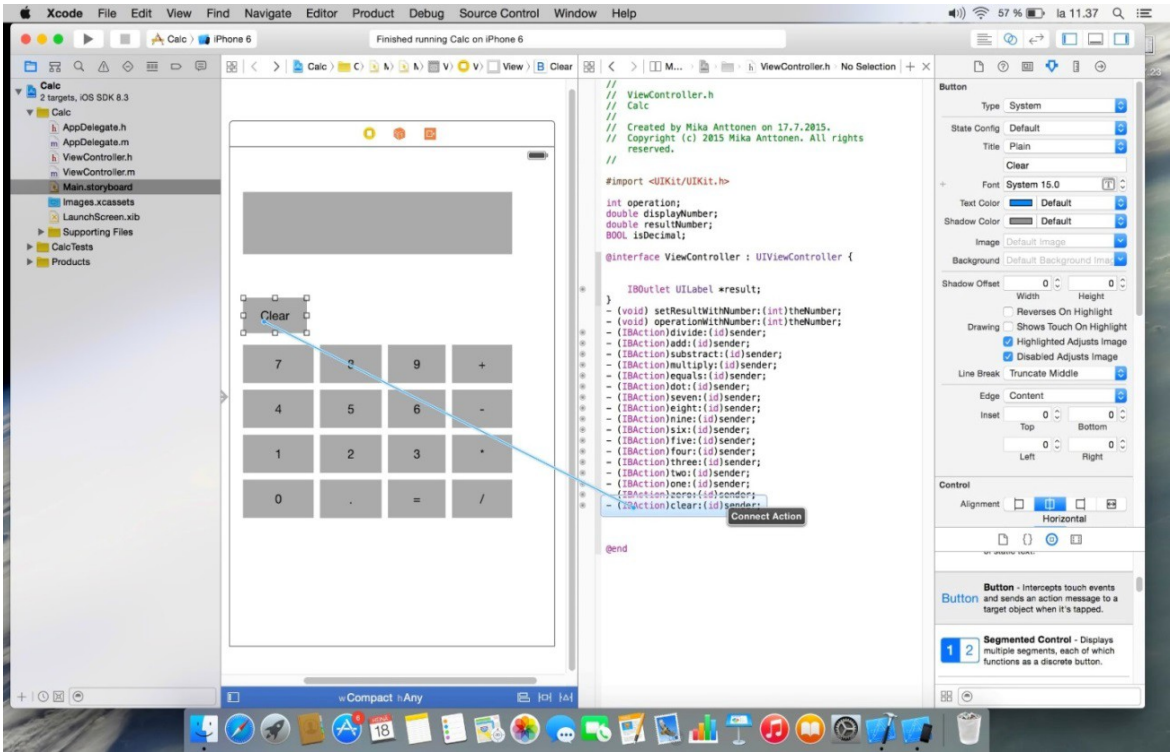
6.3 Objective-C ja XCode

Kolmantena tehtiin sovellus Applen XCode-ympäristössä. Ohjelmointikielenä käytettiin Objective-C-kieltä, jolla voidaan ohjelmoida sovelluksia ainoastaan Applen iOS-laitteille.

6.3.1 Main Storyboard

Sovelluksen luominen aloitettiin käyttöliittymän eli käyttäjärajapinnan luomisella. Main.storyboard-välilehdelle vedettiin hiiren avulla alhaalla olevasta ikkunasta painikkeet haluttuja toimintoja varten sekä label-kenttä toimimaan näyttönä laskimelle.

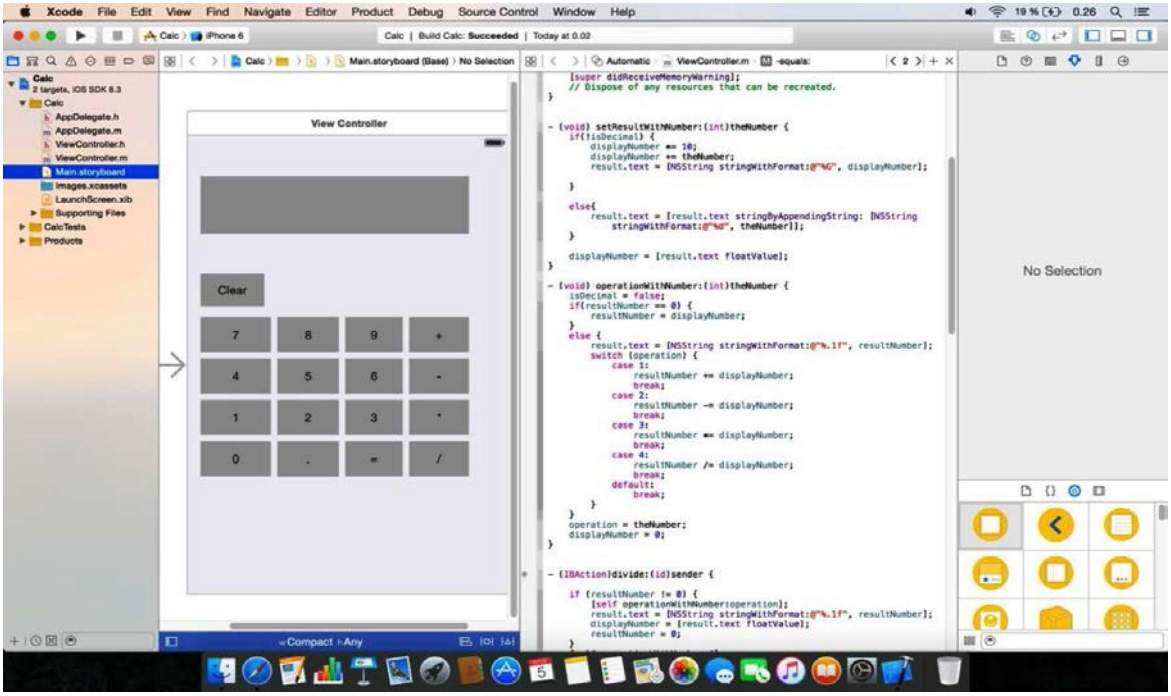
Seuraavaksi määriteltiin painikkeille sekä näytölle datatyypeiksi ”double” desimaaleja varten. Lopuksi nimettiin ja määriteltiin halutut toiminnot kaikille käyttöliittymän komponenteille. XCodessa tämä voitiin tehdä ehkä hieman totutusta poikkeavalla tavalla, eli vetämällä toiminto esimerkiksi ”clear”-painikkeesta suoraan ViewController.h-koodiin. (Kuva 15.)



KUVA 15. ViewController.h

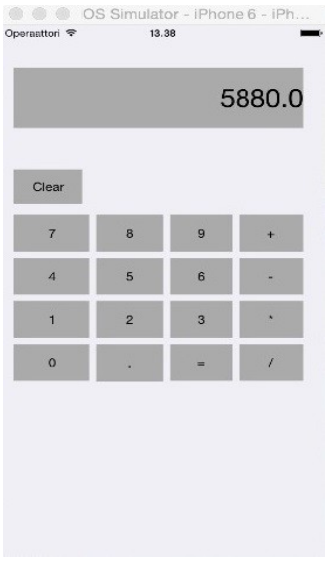
6.3.2 ViewController

Objective-C:n ViewController.m-tiedostoon kirjoitettiin tälläkin kertaa switch case -valintarakenteen sisään halutut laskuoperaatiot. Lisäksi koodiin muun muassa määriteltiin näyttämään laskutoimituksen tulos yhden desimaalin tarkkuudella. (Kuva 16.)



KUVA 16. ViewController.m

Objective-C-sovellus toimivana sovelluksena iPhone 6 -simulaattorissa (kuva 17).



KUVA 17. iPhone Calculator

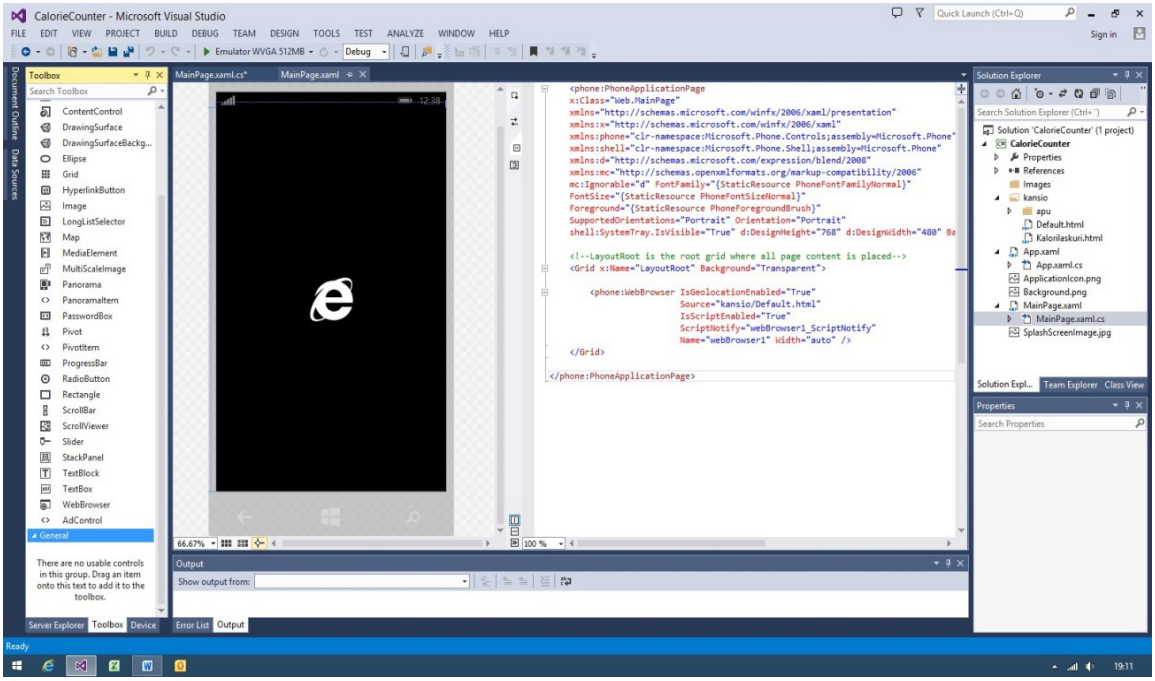
7 SOVELLUKSEN OHJELMOINTI JA SIIRTO FYYSISEEN LAITTEESEEN

Käytännön esimerkkinä kohdelaitteeseen siirtoa varten ohjelmoitiin Windows Phone -sovellus web-tekniikoita käyttäen. Tämä toteutettiin Visual Studio 2013 -ympäristössä C-Sharp-ohjelmointikielellä. Sovelluksessa ohjelmoitiin ns. hybridi web-sovellus, jossa hyödynnettiin HTML:ää, JavaSkriptiä sekä CSS:ää. Sovellusideana oli tehdä Ruokakalorilaskuri (Calorie Counter). Sovelluksen testaaminen oikealla laitealustalla nähtiin myös tärkeäksi sisällyttää tähän opinnäytetyöhön.

Windows Phone 7 oli ainoa käytettävissä oleva kohdealusta, johon sovellusesimerkki voitiin siirtää. Se vaati ensin rekisteröitymisen Microsoftin Windows Phone -kehittäjäksi. Windows Phone -kehittäjälisenssi saatiin opiskelukäyttöön ilmaiseksi.

7.1 MainPage.xaml

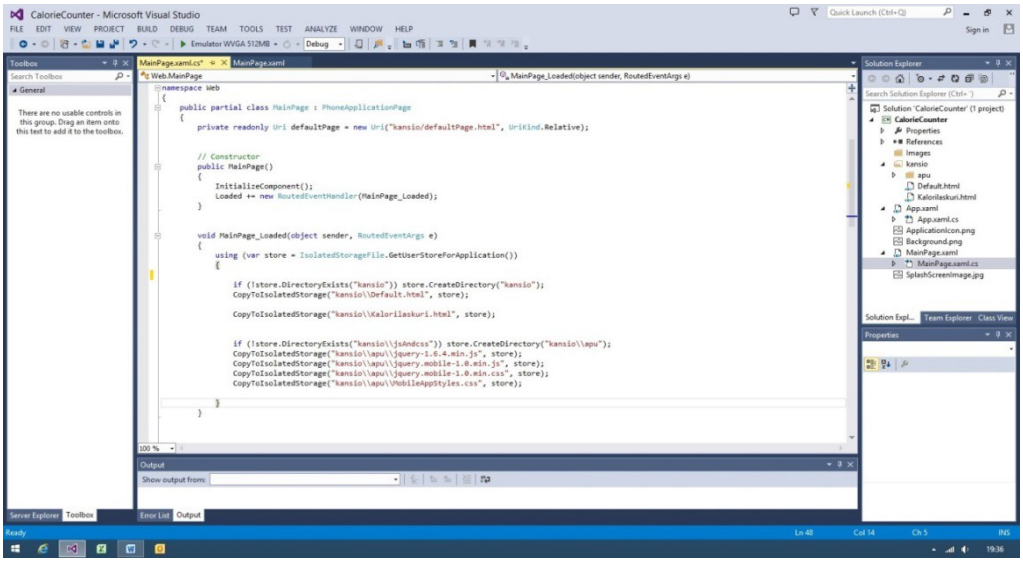
Kalorilaskurin ohjelmointi Windows Phonelle aloitettiin luomalla MainPage.xaml-näkymä, johon vedettiin Toolbox-ikkunasta ainoastaan WebBrowser-kontrolli, johon sovellus saadaan myöhemmin näkyviin. Halutessaan kaikki tarvittavat komponentit voidaan myös kirjoittaa suoraan kuvassa oikealla näkyvään xaml-koodiin, jonka jälkeen ne on käytettävissä sovelluksessa halutuissa kohdissa. (Kuva 18.)



KUVA 18. Calorie Counter MainPage.xaml

7.2 MainPage.xaml.cs

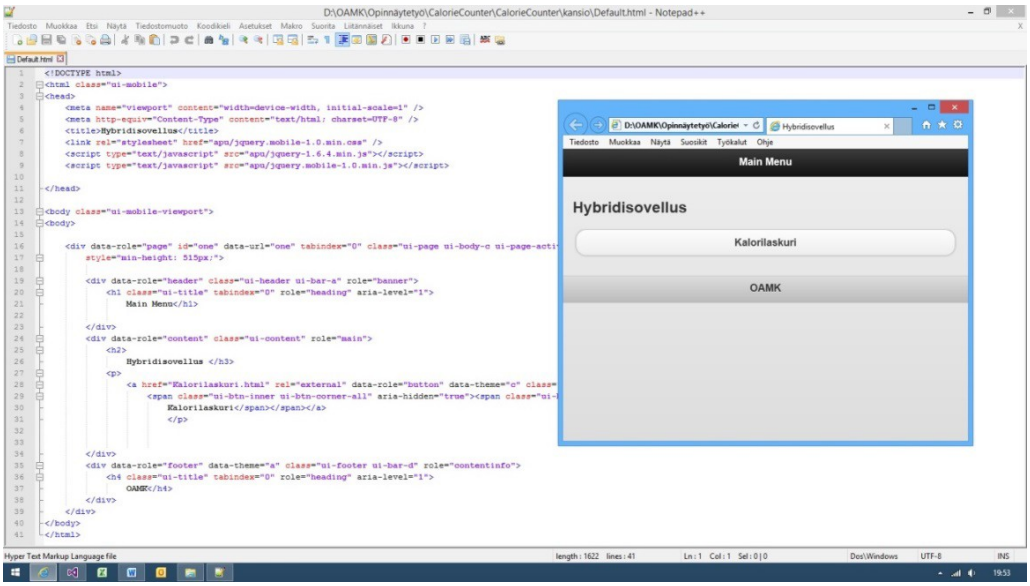
MainPage.xaml.cs-tiedostoon kirjoitettiin tarvittavat koodit html- ja JavaScript-tiedostoja varten. Näin sovellus osaa hakea ne projektiin määritellyistä kansioista ja tiedostoista. (Kuva 19.)



KUVA 19. Calorie Counter MainPage.xaml.cs

7.3 HTML default -sivu

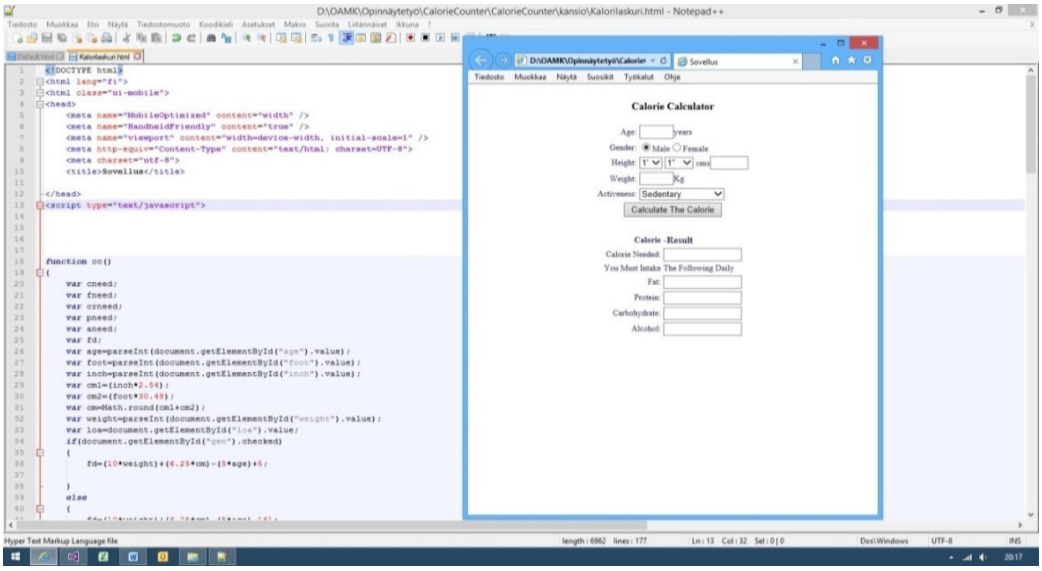
HTML Default -sivun koodi (sovelluksen aloitusnäkymä) tehtiin Notepad++-tekstieditorissa. Testausvaiheessa koodi ajettiin Internet Explorerin Web-selaimeen. (Kuva 20.)



KUVA 20. HTML Default -sivun koodi sekä Web-selain

7.4 Skriptikoodi

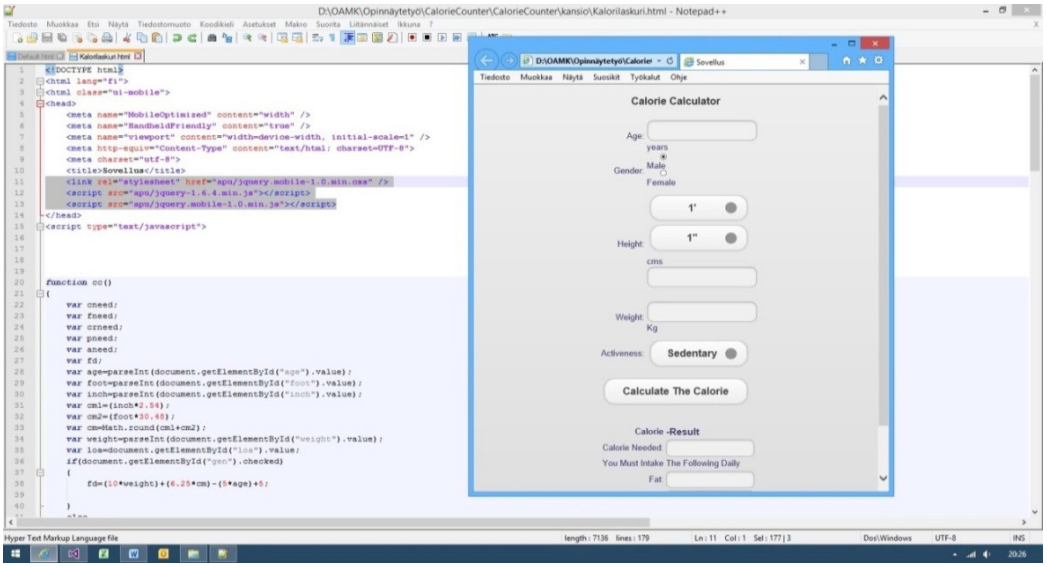
Skriptikoodi lisättiin mukaan koodiin ja käyttöliittymä ajettiin Web-selaimeen vielä ilman jQueryä. Tässä vaiheessa sovelluksen käyttöliittymä haluttiin ainoastaan testata ja todeta toimivaksi. (Kuva 21.)



KUVA 21. Skriptikoodi ja käyttöliittymä

7.5 Skriptikoodi jQueryn tiedostoilla

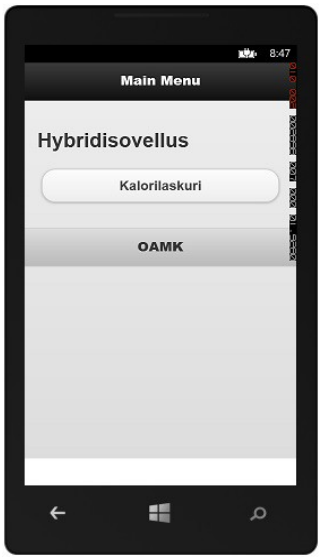
Seuraavaksi koodiin lisättiin viittaukset jQueryn tiedostoihin HTML-koodin ”head”-osassa. jQueryn avulla käyttöliittymä saatiin visuaalisesti näyttämään paremmalta. (Kuva 22.)



KUVA 22. Skriptikoodi ja käyttöliittymä jQueryllä

7.6 Main menu

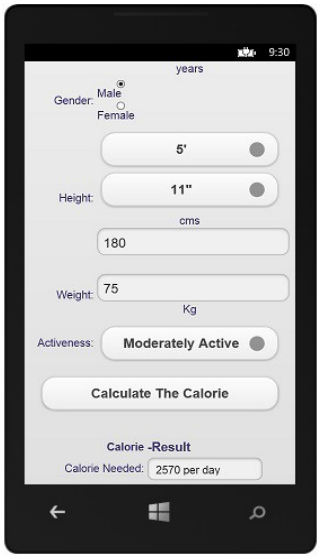
Valmis ohjelma testattiin Windows Phone 8 -emulaattorissa. Main Menu -näkymään koodattiin painike ”Kalorilaskuri”, jota painamalla sovellus käynnistettiin. (Kuva 23.)



KUVA 23. Main Menu

7.7 Calorie Counter

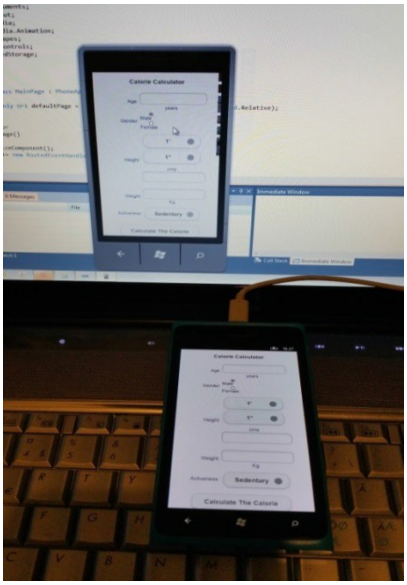
Sovelluksen käynnistyttyä kenttiin voitiin syöttää vaaditut tiedot. Tämän jälkeen painettiin Calculate The Calorie -painiketta, jolloin tarvittu päivittäinen kalorimäärä voitiin nähdä Calorie ded -kohdassa. (Kuva 24.)



KUVA 24. Calorie Counter

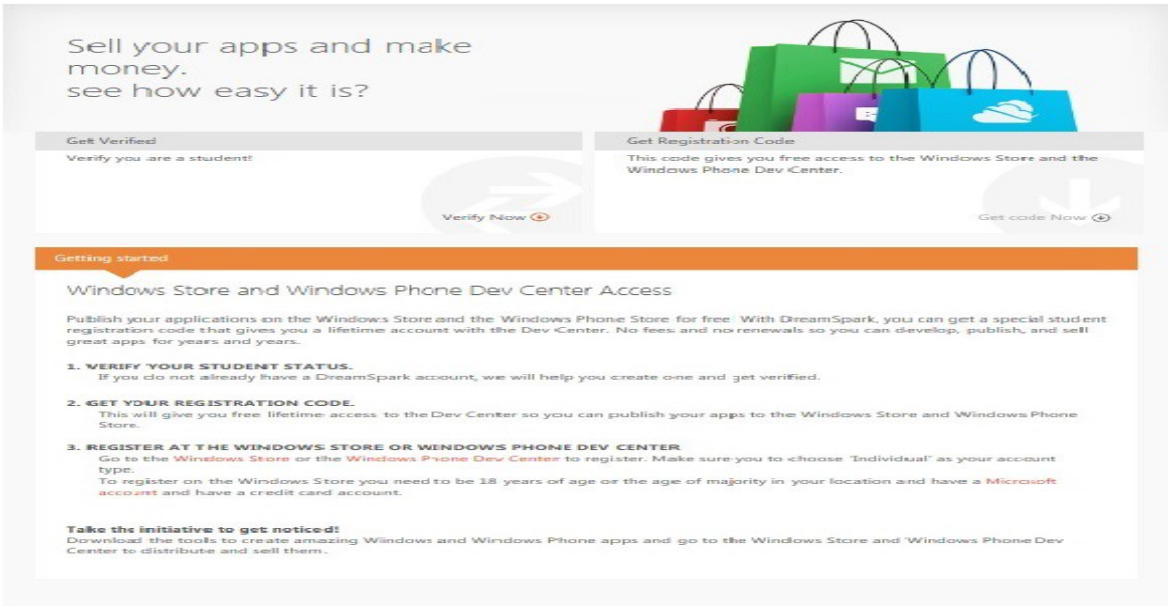
7.8 Windows Phone

Lopuksi sovellus ajettiin Windows Phone -laitteeseen ja havaittiin sen toimivan oikealla tavalla. Laitteena käytettiin vanhempaa Windows Phone 7 Nokia Lumia 900 -puhelinta. Kuvassa 25 on nähtävillä Calorie Counter emulaattorissa ja laitteessa.



KUVA 25. Calorie Counter

Windows Phone -sovellusten ajamiseen omalle laitteelle vaadittiin rekisteröityminen ohjelmistokehittäjäksi Microsoftin DreamSpark -sivustolla, josta saatiin myös selkeät ohjeet aiheeseen liittyen (kuva 26).



KUVA 26. Microsoft DreamSpark (22)

8 JOHTOPÄÄTÖKSET

Mobiilijärjestelmien ohjelmointiin suositeltavimmaksi ohjelmointikieleksi itse pitäisin varteenotettavana vaihtoehtona C#:a. Suurin syy siihen on, että C# on mielestäni helpoin omaksua, vaikkakin kieli on alkujaan kehitetty Javasta sekä C++-kielestä. Osittain se johtuu varmasti myös siitä, että Visual Studion ohjelmointiympäristö toimii suhteellisen luotettavasti, käyttämäni versiot 2010 sekä 2013. C#:n hyödyllisyydestä nimenomaan mobiiliohjelmoinnin näkökulmasta katsoen on kuitenkin muistettava, että kielellä voidaan tehdä sovelluskehitystä ainoastaan Windows Phone -alustoille, toisin kuin Java, jolla voidaan tehdä sovelluksia useille eri laitealustoille.

Kuitenkin C# voisi toimia myöhemmin hyvänä pohjana Javan ja C++:n opiskeluun, koska C# sisältää piirteitä molemmista kielistä. Kielien perussyntaksi kuten lauserakenteet, valintarakenteet, silmukat eli loopit sekä oliot luodaan hyvin samantyyppisesti molemmissa kielissä, sekä Javassa että C#:ssä, jotka on mielestäni myös helppo oppia jopa ulkoa.

Applen Objective-C:stä sekä XCode-kehitysympäristöstä minulla ei ollut ennen tätä opinnäytetyötä aikaisempaa kokemusta. Pienen tutustumisen ja kielen perusteiden sekä kehitysympäristöön tutustumisen jälkeen kuitenkin työskentely näillä työkaluilla oli verrattain sujuvaa ja mielenkiintoista. Pitäisinkin yhtenä vaihtoehtona Objective-C-kieltä ensimmäiseksi ohjelmointikieleksi, silloin mikäli Applen mobiililaitteiden sovelluskehitys muutoin tuntuu kiinnostavalta.

Tekemieni haastattelujen perusteella mielekkäimpänä ohjelmointikielenä useimmat pitävät sitä, jota on itse tottunut käyttämään. Kielessä itsessään erot eivät ole niin suuria, että joku kieli sopisi huonommin ensimmäiseksi kieleksi kuin joku toinen. Kuitenkin jokaisella kielellä päästään yleensä samaan lopputulokseen, joten ihan perustellusti voidaan sanoa ohjelmointikielen olevan hyvin paljon tottumiskysymys. Lisäksi painoarvoa on myös annettava kehitysympäristölle eli jos ympäristöä on miellyttävä käyttää, se osaltaan motivoi opiskelemaan ohjelmointia.

PhoneGap, jota ei tässä työssä käsitelty, on mielestäni kuitenkin mainitsemisen arvoinen sovelluskehys joka poikkeaa tyypillisistä kielistä. Periaatteessa PhoneGapissa on jo valmiina ”runko”, jonka ympärille sovellusta lähdetään jollakin tietyllä kielellä halutulle alustalle kehittämään. PhoneGap -ohjelmistokehityksessä käytetään HTML-kieltä, CSS-kielisiä tyylitiedostoja sivujen ulkoasun muotoiluun ja JavaScriptejä erinäisten toiminnallisuuksien lisäämiseksi sovelluksiin. Alustoja, joille PhoneGap -sovelluksia voidaan tehdä, ovat esimerkiksi iOS-, Android-, ja Windows Phone -alustat. Pho-

neGapilla voidaan siis tehdä kehitystyötä samoilla alustoilla sekä kielillä kuin tässä opinnäytetyössä omilla ympäristöillään tehdyt sovelluksetkin.

Miten mobiiliohjelmointi sitten eroaa perinteisestä ohjelmoinnista? Mobiiliohjelmoinnin opetukseen ja mobiiliohjelmointiin yleensä liittyy tiettyjä erityisvaatimuksia, jotka tulisi huomioida verrattuna perinteiseen ohjelmointiin eli ohjelmiin, jotka on tarkoitettu toimimaan työpöytäkäyttöön tarkoitetuissa tietokoneissa. Usein mobiilipuolella joudutaan tekemään samoja sovelluksia useille eri alustoille, jolloin tietty alusta voi vaatia tietyn ohjelmointiympäristön sekä kielen, jotta alustakohtainen ohjelmistokehitys olisi mahdollista.

Lisäksi mobiilisovelluksia kehitettäessä laitteen komponenttitasolla eli fyysisellä puolella on huomioitava laitteen suorituskyky, vaikkakin nykyiset mobiililaitteet ovat jo kuitenkin suorituskyvyltään perustietokoneen tasoa. Näytön tarkkuus eli resoluutio on myös usein otettava huomioon sovellusta kehitettäessä sekä unohtamatta virrankulutusta, koska laitetta joudutaan käyttämään pääsääntöisesti akkuvirralla. Myös laitteeseen integroidut anturit, kuten kiihtyvyys- ja ilmanpaineanturit sekä asennontunnistimet jne. tuovat myös omat lisähaasteensa mobiilijärjestelmien ohjelmointiin liittyvään suunnittelu- ja kehitystyöhön.

9 LOPPUSANAT

Tässä opinnäytetyössä tehtiin itsenäinen tutkimustyö, jonka tavoitteena oli määritellä mobiiliohjelmointia aloittelevalle opiskelijalle suositeltavin ohjelmointikieli, jonka avulla pääsee kaikkein sujuvimmin alkuun mobiiliohjelmoinnissa.

Kaiken kaikkiaan tämän opinnäytetyön suunnittelu sekä esimerkkien ohjelmointi ja varsinkin sovellusten testaaminen oikeassa laitteessa olivat erittäin mielenkiintoisia vaiheita työtä tehdessä. Toisaalta eri kehitysympäristöjen käyttö ehkä hankaloitti vertailua, mutta toisaalta se teki työstä vastaavasti mielenkiintoisempaa. Eclipsen ja Androidin kehitysympäristöltä olisin toivonut lisäksi enemmän nopeutta sekä luotettavuutta. Ympäristö on toivottoman hidas verrattuna Microsoftin Visual Studioon tai varsinkin Applen XCodeen.

Lisäksi oikeanlaisen tiedon etsiminen ja hankkiminen niin, että se oli sovellettavissa tähän työhön, oli yllättävän vaativaa työtä.

LÄHTEET

1. Javan historia ja muita ohjelmointikieliä. Kielikartta. Saatavissa: http://www.syreeni.huone.net/javakurssi/arkielama_ehdot_vertailu/arkielama/historia.html. Hakupäivä 30.10.2015

2. FORTRAN. 2015. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/Fortran. Hakupäivä 30.10.2015

3. COBOL. 2015. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/COBOL. Hakupäivä 30.10.2015

4. ALGOL. 2015. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/ALGOL. Hakupäivä 30.10.2015

5. LISP. 2015. Wikipedia. Saatavissa: http://fi.wikipedia.org/wiki/Lisp. Hakupäivä 30.10.2015

6. AutoLISP. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/AutoLISP. Hakupäivä 30.10.2015

7. Riekki, Jukka 2009. Hieman ohjelmointikielten historiaa. Raippa: Ohjelmoinnin alkeet ja ATK I Ohjelmoinnin perusteet - 521141P. Saatavissa: https://www.raippa.fi/elementary-programming/Ohjelmointikielten%20historia. Hakupäivä 30.10.2015.

8. Kovanen, Pasi 2013. Ohjelmointikielet vertailussa. Vincit/blog. Saatavissa: http://www.vincit.fi/?s=ohjelmointikielet+vertailussa. Hakupäivä 30.10.2015.

9. Objective-C. 2013. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/Objective-C. Hakupäivä 30.10.2015

10. Swift. A modern programming language that is safe, fast and interactive. 2015. Swift. Saatavissa: https://developer.apple.com/swift/. Hakupäivä 30.10.2015

11. Mikä on HTML5? Web-opas. Saatavissa: http://www.webopas.net/html5.html. Hakupäivä 30.10.2015

12. JavaScript-opas. 2015. 2kmediat.com. Saatavissa: http://www.2kmediat.com/jscript/upottaminen.asp. Hakupäivä 30.10.2015

13. CSS Tutorial. 2015. w3schools.com. Saatavissa: http://www.w3schools.com/css/. Hakupäivä 30.10.2015

14. XCode. 2015. Mac App Store Preview. Saatavissa: https://itunes.apple.com/us/app/xcode/id497799835?ls=1&mt=12. Hakupäivä 30.10.2015.

15. Eclipse (IDE). 2015. Wikipedia. Saatavissa: http://fi.wikipedia.org/wiki/Eclipse_(IDE). Hakupäivä 30.10.2015.

16. Microsoft Visual Studio. 2015. Wikipedia. Saatavissa: http://fi.wikipedia.org/wiki/Visual_Studio. Hakupäivä 30.10.2015.

17. Grønli, Tor Morten., Hansen, Jarle. & Ghinea, Gheorghita 2010. Android vs. Windows Mobile vs. Java ME: a comparative study of mobile development environments. 2015. ResearchGate: Conference Paper. Saatavissa: http://www.researchgate.net/publication/221410373_Android_vs_Windows_Mobile_vs_Java_ME_a_comparative_study_of_mobile_development_environments. Hakupäivä 30.10.2015

18. Sonmez, John 2010. Simple Programmer: C# vs Java Part 1: The Languages. Saatavissa: http://simpleprogrammer.com/2010/02/01/c-vs-java-part-1-the-languages/. Hakupäivä 30.10.2015

19. Mikkonen, Tommi 2015. Professori, Tampereen teknillisen yliopiston tietotekniikan laitos. Re: Mielipidekysymys mobiiliohjelmointikielistä. Sähköpostiviesti. Vastaanottaja: Mika Anttonen. 15.7.2015

20. Niemi, Eino 2015. Lehtori, Oulun ammattikorkeakoulun tekniikan ja luonnonvara-alan yksikkö. Mielipidekysymys mobiiliohjelmointikielistä. AC-haastattelu. 22.9.2015

21. Pirttiaho, Lauri 2015. Yliopettaja, Oulun ammattikorkeakoulun tekniikan ja luonnonvara-alan yksikkö. Mielipidekysymys mobiiliohjelmointikielistä. Sähköpostiviesti. Vastaanottaja: Mika Anttonen. 12.9.2015

22. Students. 2015. Microsoft Developer Network. Saatavissa: https://www.dreamspark.com/Student/Windows-Phone-8-App-Development.aspx. Hakupäivä 30.10.2015

LÄHTÖTIETOMUISTIO

Tekijä

ANTTONEN MIKA MATTI FREDRIK

Tilaaja

OULUN AMMATTIKORKEAKOULU

Tilaajan yhdyshenkilö ja yhteystiedot

JUHA ALAKÄRPPÄ

KOTKANTIE 1, 90250 OULU

Työn nimi

MOBIILIOHJELMOINNIN PERUSOPETUKSEEN SOVELTUVAN OHJELMOINTIKIELEN VALINTA

Työn kuvaus ja tavoitteet

Työ liittyy mobiiliohjelmointiin jossa opinnäytetyön tarkoituksena ja tavoitteena on tehdä ohjelmointikielen valintaan liittyvä tutkimustyö jolla ohjelmointia aloitteleva opiskelija pääsee kaikkein sujuvimmin alkuun mobiiliohjelmoinnissa sekä toivottavasti myös ideoita ja vinkkejä opettajille jotka tulevat opettamaan mobiiliohjelmointia oppilaitoksissa.

Tavoiteaikataulu

aloituspalaveri: Syksy 2014

työn suunnittelu: Syksy 2014

työhön tarvittavan materiaalin ja tiedon hankinta: Syksy 2014

opinnäytetyön toteuttaminen Syksy 2014 – Syksy 2015

työn esittely: syksy 2015

Päiväys ja allekirjoitukset

17.9.2014 Mika Anttonen _____

OPINNÄYTETYÖN OHJELMISTOASENNUKSET

Mobiilisovellusten ohjelmoimiseksi tässä opinnäytetyössä käytetyillä eri alustoilla ladattiin sekä asennettiin seuraavat ohjelmat verkosta, jotta kehitysympäristöt saatiin toimimaan tarkoitetulla tavalla.

Java Development KIT

- Java Development Kit: http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

Java Devlopment Kit (JDK) on ohjelmistopaketti, joka sisältää Java-ohjelmoinnissa välttämättömät sovellukset.

Eclipse ja Android SDK

- Eclipse: http://www.eclipse.org/downloads/
- Android SDK: http://developer.android.com/sdk/index.html

Lisäksi Eclipseen jouduttiin asentamaan vielä erillinen lisäosa eli ADT (Android Devlopment Tools). Tämän lisäosan asennus tuli tehdä Eclipsessä käynnistämällä ylävalikosta Help - Install New Software… Klikattiin Add ja syötettiin avautuvaan dialogiin ”ADT” ja osoitteeksi annettiin https://dl-ssl.google.com./android/eclipse/ ja painettiin OK.

Microsoft Visual Studio versiot ja Windows Phone 8.1 SDK

Windows Phone 8.1:n asennuksessa vaadittiin että Visual Studio 2013:een oli asennettu päivityspaketti 2 tai uudempi.

- Microsoft Visual Studio: http://www.visualstudio.com/downloads/download-visual-studio-vs
- Windows Phone SDK 8.1: http://www.microsoft.com/en-us/download/details.aspx?id=43719

XCode ja iOS 6

XCode:n ohjelmointiympäristö ladattiin ja asennettiin Mac - tietokoneeseen Applen kehittäjätyökalut -sivustolta. XCode IDE:n mukana saatiin valmiina myös SDK -paketti.

XCode IDE: https://itunes.apple.com/us/app/xcode/id497799835?ls=1&mt=12