Tallinna Ülikool

Informaatika Instituut

Rakenduse loomine iOS operatsioonisüsteemiga seadme jaoks.

Õppematerjal

Seminaritöö

Autor: Romil Rõbtšenkov

Juhendaja: Andrus Rinde

Autor: ....." ....."2014

Juhendaja: ....." ....."2014

Instituudi direktor: ....." ....."2014

Tallinn 2014

Autorideklaratsioon

Deklareerin, et käesolev seminaritöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.... (kuupäev) (autor)

Sisukord

Sissejuhatus

Nutitelefonid ja tahvelarvutid on viimastel aastatel muutunud järjest populaarsemaks, mistõttu on laienenud ka mobiilsete seadmete kasutusvaldkonnad. Tekkinud olukord on suurendanud vajadust uute mobiilsete rakenduste vastu ja ka nende loomist käsitleva õppematerjali järele. Eesti keeles on antud valdkonnas autorile teadaolevalt Android operatsioonisüsteemi käsitlevaid materjale aga antud seminaritöös käsitletava iOS operatsioonisüsteemi jaoks rakenduste loomise õppimiseks materjalid puuduvad.

Käesoleva töö teema valik tulenes vajadusest eestikeelse õppematerjali vastu, mille abiga oleks võimalik iseseisvalt õppida rakenduse loomisest iOS operatsioonisüsteemiga seadme jaoks. Töö valmimise hetkel oli võimalik Tallinna Ülikooli Informaatika Instituudis õppida ainet nimega Rakenduste programmeerimine, mis käsitles teiste teemade seas mobiilsete rakenduste loomist, kuid seda ainult teise populaarse mobiilse operatsioonisüsteemi - Androidi jaoks. Teema valikut mõjutas lisaks autori isiklik huvi õpetamise ning mobiilsete rakenduste loomise vastu.

Seminaritöö eesmärk on luua õppematerjal, mille abil programmeerimiskogemusega huvilised iseseisvalt omandada iOS operatsioonisüsteemi jaoks rakenduse loomise esmased oskused. Oodatav tulemus on luua õppematerjal, mis annaks ülevaate iOS operatsioonisüsteemist, arendajatele pakutavatest võimalustest, arendamiseks kasutatavast tarkvarast ja tekitaks õppijates huvi rakenduste loomise vastu antud mobiilsele operatsioonisüsteemile.

Eesmärkide saavutamiseks annab autor esimeses peatükis kirjanduse põhjal ülevaate iOS operatsioonisüsteemist, selle populaarsusest ja selle jaoks arendajatele pakutavatest võimalustest.

Teises peatükis tutvustatakse iOS operatsioonisüsteemile rakenduste arendamiseks vajalikku arenduskeskkonda Xcode, sinna kuuluvat rakendust iOS Simulator ja nõudeid arendaja arvutile, millega seda keskkonda kasutada saab.

Kolmandas peatükis analüüsitakse loodava õppematerjali nõudeid, ülesehitust ja tutvustatakse võimalikke õpiteid. Lisaks kirjeldatakse õppematerjali testimist Tallinna Ülikooli Informaatika Instituudi bakalaureuseõppe kolmanda kursuse üliõpilastega, antakse ülevaade testgrupilt saadud tagasisidest ja analüüsitakse tulemusi.

Seminaritöö lisades on õppematerjal, mida on võimalik pdf-failina allalaadida aadressilt http://romil.ee/seminaritoo/oppematerjal_rakenduse_loomiseks_iOS_operatsioonisusteemiga_seadme_jaoks.pdf (12MB). Lisaks sellele on lisades õppematerjali testimise tagaside küsitluse ankeet ja küsitluse tulemused.

1 iOS operatsioonisüsteem

iOS on Apple'i poolt loodud operatsioonisüsteem, mis on kasutusel seadmetel iPhone, iPad, ja iPod Touch. See haldab seadmete riistvara ja varustab tehnoloogiaga, mis on vajalik seadme jaoks loodavatele rakendustele. Operatsioonisüsteemi on eelpaigaldatud hulgaliselt Apple'i poolt loodud rakendusi nagu Telefon, Kalender, E-post, Safari, Kaamera, Muusika jpt, mis võimaldavad kasutajale nutitelefonile omaseid funktsionaalsusi (Apple Inc, 2013).

1.1 iOS kasutajad

Turuuuringutele, analüüsile ja konsulteerimisele keskendunud, Ammerika Ühendriikides asuv ettevõte, International Data Corporation (IDC) avaldas oma pressiväljaandes 2013. aasta kolmandas kvartalis populaarsemate operatsioonisüsteemide jaotuse ülemaailmsel mobiiliseadmete turul. Välja toodud andmete järgi on teiste operatsioonisüsteemidega võrreldes populaarsemad iOS, turuosaga 14.4 % ja Android turuosaga 74.9 % (IDC, 2013).

Eesti turundusfirma smartAD veebipõhise vidina nimega Addresstising raamat[1] andmete järgi on Eestis 65303 - 72559 Android operatsioonisüsteemi kasutavat mobiiliomanikku ning iOS operatsioonisüsteemi kasutavad vastavalt 38809 - 43133 iPad tahvelarvuti ja 37106 – 41229 iPhone'i omanikku. Android operatsioonisüsteemiga tahvelarvutite kasutajate arvu ei ole välja toodud. Nende andmete põhjal saab järeldada, et iPhone'i omanikke on Eestis palju ja suure kasutajaskonnaga iOS operatsioonisüsteemile rakenduste loomise oskus on väga oluline.

1.2 iOS 7

iOS 7 on õppematerjali loomise ajal kõige uuem iOS operatsioonisüsteemi versioon, mida tutvustati avalikkusele 2013. aasta juunis. Apple'i poolt läbi viidud uuringu järgi kasutatakse 2013. aasta detsembri seisuga operatsioonisüsteemi iOS 7 ülekaalukalt 78%, iOS 6 kasutatakse 18% ja kõiki vanemaid versioone kokku kasutatakse 4%. Antud statistika näitab, et üleminek uuele operatsioonisüsteemile on olnud kiire ja kasutajad on uue funktsionaalsuse omaks võtnud (Hughes, 2013).

[1] Veebipõhine vidin **Addresstising raamat** on saadaval aadressil http://smartad.eu/estonia/addresstising_book/

Võrreldes vanemate iOS operatsioonisüsteemidega, on iOS 7 kasutajaliidese uuendusliku kujundusega (Joonis 1) ja pakub kasutajatele uusi omadusi ja funktsionaalsust nagu Juhtimiskeskus (*Control Center*), Teavituskeskus (*Notification Center*) jpt (Apple Inc, 2013).

Joonis 1 iOS 7 (Apple Inc, 2013)

1.3 Arendamine iOS 7 operatsioonisüsteemi jaoks

Apple on loonud iOS 7 operatsioonisüsteemile rakenduste arendajatele eraldi veebikeskkonna, tööriistad ja materjalid, tehes sellega nii arendamise kui ka õppimise võimalikult lihtsaks. Teatud materjalidele ligipääsu saamiseks ja App Store'i kaudu valmisrakenduste levitamiseks, on vaja aga kindlasti liituda ühe arendajatele mõeldud tugiprogrammiga.

1.3.1 iOS 7 SDK

iOS Software Development Kit (SDK) sisaldab endas erinevaid tööriistu, mis on vajalikud *native* tüüpi rakenduste arendamiseks, seadmetele paigaldamiseks, käivitamiseks ja ka testimiseks. *Native* tüüpi rakendused on loodud kasutades iOS operatsioonisüsteemile mõeldud teeke ja Objective-C programmeerimiskeelt ning töötavad otseselt iOS operatsioonisüsteemil. Erinevalt veebipõhistest rakendustest, on *native* tüüpi rakendused paigaldatud füüsiliselt seadmele ja on alati kasutajale kätte saadavad, isegi juhul kui seade on Lennukirežiimis. Kõik rakendused salvestatakse koos võimalike andmetega kasutaja kohta arvutisse, kasutades selleks programmi iTunes (Apple Inc, 2013).

1.3.2 Reeglid rakenduste kasutajaliidese kujundamiseks

Selleks, et iOS operatsioonisüsteemi jaoks tehtud rakendustel oleks hästi toimiv kasutajaliides, on Apple loonud dokumendi nimega OS Human Interface Guidelines[2], kus on arendajate jaoks täpselt kirjeldatud kasutajaliidese disaini eeskirjad. Lisaks sisaldab see piisavalt nõuandeid hea kasutajaliidese loomiseks, seal on kirjeldatud kuidas tuleb integreerida sotsiaalmeediat ja milliseid valmisobjekte on võimalik kasutajaliidese loomisel kasutada, millest tuleb lähtuda ikoonide kujundamisel ning tutvustatakse ka kuidas rakenduste loomisel saab arvestada vaegnägijatega (Apple Inc, 2013).

1.3.3 Arendajate tugikeskkond

iOS Dev Center[3] on veebikeskkond, kus arendajatel on võimalik leida Apple'i poolt loodud dokumentatsiooni ja muid materjale, mis arendamisel kasuks tulevad. Samuti on alamlehtedena olemas arendajatele mõeldud foorum ja muud abilehed. Sarnased veebikeskkonnad on loodud Mac OS jaoks rakenduste ja Safari jaoks pistikprogrammide arendajatele.

1.3.4 iOS arendajatele mõeldud tugiprogrammid

Apple on loonud arendajatele mõeldud kolm erinevat programmi, mis võimaldavad arendajatele peale nendega liitumist erinevaid hüvesid.

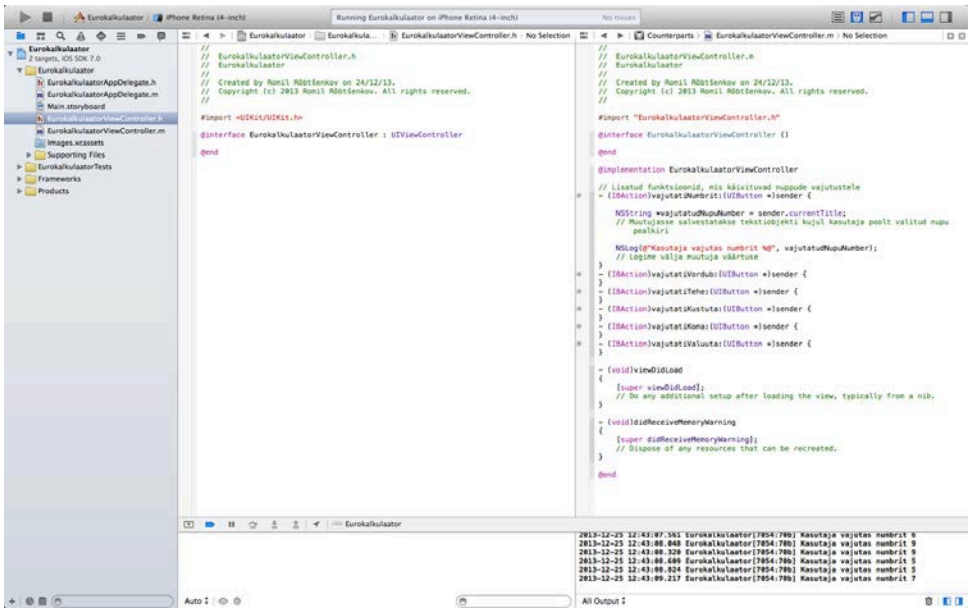
- **iOS Developer Program** on kõige tavalisem programm annab 99 dollari eest aastas võimaluse arendatud rakendusi testida päris seadmetel ja valmisrakendusi levitada App Store'i kaudu.
- **iOS Developer Enterprise Program** võimaldab 299 dollari eest aastas arendada ja levitada rakendusi ettevõte, avalike asutuste, õppeasutuste või muude asutuse siseselt.
- **iOS Developer University Program** on tasuta ja on mõeldud kraadiõpet või kõrgharidust võimaldavatele õppeasutustele, andes nende tudengitele võimaluse rakendusi päris seadmetel testida (Apple Inc, 2013).

[2] iOS Human Interface Guidelines on saadaval aadressil https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf
[3] iOS Dev Center on saadaval aadressil https://developer.apple.com/devcenter/ios/index.action

Viimase kirjeldatud programmiga, iOS Developer University Program, on liitunud Tallinna Ülikooli Informaatika Instituudi Haridustehnoloogiakeskus. Antud õppematerjali kasutamiseks ei ole vajalik ühegi arendajatele mõeldud programmiga liituda.

2 Arenduskeskkond Xcode

Xcode on Apple'i poolt loodud arenduskeskkond, mis võimaldab projektihaldust, koodi redigeerimist, kompileerimist, vigade otsimist, failide versioonihaldust, testimist jpm. Arenduskeskkonna keskse osa moodustab Xcode nimeline rakendus (Joonis 2), mis pakub koodi redigeerimise võimalust ja ligipääsu kõikidele teistele tööriistadele. Tähtsamad rakendused, mis Xcode arenduskeskkonda veel kuuluvad, on jõudluse ja teiste tegurite testimist võimaldav rakendus Instruments ja mobiilset seadet simuleeriv iOS Simulator. Xcode on tasuta allalaetav Mac arvutile App Store-ist (Apple Inc, 2013).



Joonis 2 Arenduskeskkond Xcode

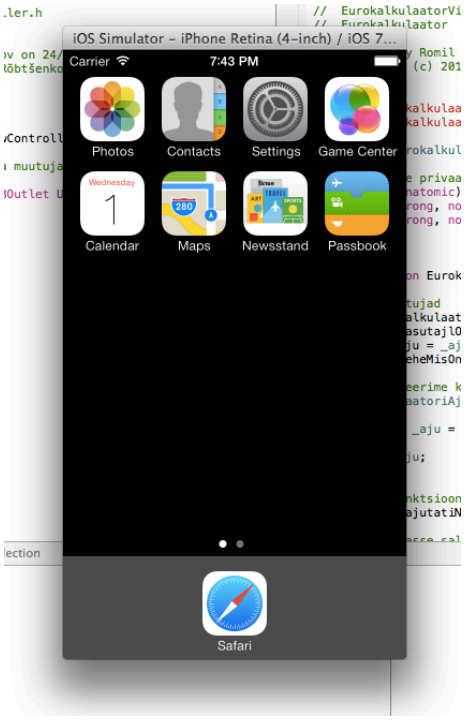
2.1 Süsteeminõuded arendaja arvutile

Arendaja arvuti ja operatsioonisüsteemi piirang on Apple'i poolt küllaltki range, nõudes arendaja arvutiks kindlasti Apple'i poolt loodud Mac arvutit ja OS X operatsioonisüsteemi. Ametlikult puuduvad nii arendustarkvara kui ka tugi, et arendamisel kasutada teisi arvuteid ja Microsoft Windows operatsioonisüsteemi. Selleks, et arendada rakendust iOS 7 operatsioonisüsteemile on vaja täita järgmised nõuded:

- Apple'i Mac arvutil peab olema operatsioonisüsteem OS X 10.7 (Lion) või uuem;
- arvutisse peab olema paigaldatud arenduskeskkond Xcode 5;
- arenduskeskkonnas peab olema iOS 7 SDK (Apple Inc, 2013).

2.2 iOS Simulator

iOS Simulator võimaldab kiiret prototüüpimist ja loodud rakenduste jooksvaks testimiseks kogu arendusprotsessi käigus (Joonis 3). Antud rakendus on üks Xcode arenduskeskkonna osa koos iOS SDK-ga ja töötab nagu iga teine Mac rakendus simuleerides iPhone või iPad seadet, võimaldades testida arendatavat rakendust ka erinevate iOS versioonidega (Apple Inc, 2013).



Joonis 3 iOS Simulator rakendus

3 Õppematerjal, nõuded ja ülesehitus

Õppematerjali loomise vajadus tuleneb eestikeelsete õppematerjalide puudumisest, mille abil oleks võimalik õppida rakenduste loomist iOS operatsioonisüsteemile. Tallinna Ülikooli Informaatika Instituudis oli õppematerjali loomise hetkeks võimalik õppida ainet "IFI6059 Rakenduste programmeerimine", mis käsitles teiste teemade seas mobiilsete rakenduste loomist, kuid seda ainult teise populaarse mobiilse operatsioonisüsteemi - Androidi jaoks. Üheks õppematerjali loomise põhjuseks on töö autori suur huvi nii õpetamise kui ka rakenduste loomise vastu iOS operatsioonisüsteemile.

Õppematerjali loomisel lähtub autor enda kogemustest iOS operatsioonisüsteemile rakenduse loomisest, olles eelnevalt kasutanud mitmeid veebipõhiseid õppematerjale ning läbinud õpikeskkonnas iTunesU mõned kõige populaarsemad Stanfordi Ülikooli kursused nimedega Developing apps for iOS ja Developing iOS 7 Apps for iPhone and iPad.

Enne seminaritöö raames koostatava õppematerjali loomist, määratleti selle õppematerjali muud nõuded ja ülesehituse.

3.1 Õppematerjali nõuded

iOS operatsioonisüsteemile rakenduste loomiseks ehk käesoleva seminaritöö raames loodava õppematerjali kasutamiseks, peavad õppuril lisaks mõningasele programmeerimiskogemusele olema soovitatavalt eelteadmised klassidest, pinudest ja programmeerimiskeelest C. Sellest lähtuvalt sobib antud õppematerjal eelkõige Tallinna Ülikooli Informaatika Instituudi bakalaureuseõppe 3. kursuse üliõpilastele, kellel on autori arvates piisav eelnev programmeerimiskogemus ja peaksid olema läbitud ained "IFI6069 Programmeerimise põhikursus" ja "IFI6083 Algoritmid ja andmestruktuurid", mis annavad vajalikud eelteadmised. Programmeerimise põhikursus lihtsustab loodavas õppematerjalis arusaamist objektorienteeritud programmeerimisest ja arendamise käigus loodavatest klassidest. iOS operatsioonisüsteemile loodava rakenduse arendamisel kasutatakse Objective-C programmeerimiskeelt, mida loetakse võrdlemisi keeruliseks ja varasem kokkupuude C keelega ja pinudega aines nimega Algoritmid ja andmestruktuurid tuleb igati kasuks.

Võttes arvesse sihtrühma kuuluva üliõpilase teadmisi õppematerjalis kasutatava terminoloogia vallas, ei peetud oluliseks teatud mõisteid eraldi selgitada ja tõlkida inglisekeelseid termineid.

iOS operatsioonisüsteemile rakenduse loomise õppimiseks kasutatakse õppematerjalis ametlikku, Apple'i poolt loodud, arenduskeskkonda nimega Xcode. Sellest lähtuvalt on vaja õppematerjaliga õppimiseks Mac arvutit ja soovitatavalt ka varasem kogemus seda tüüpi arvutitega. Õppematerjali saab lähiajal kasutada Tallinna Ülikooli Informaatika Instituudi Mac-laboris, kus plaanitakse uuendada operatsioonisüsteemi ja arenduseks paigalda vaja minev arenduskeskkond.

Õppematerjali abil põhioskuste omandamine peaks olema võimalik 4 akadeemilise tunniga, lisaks on pakutud hulk iseseisvaid ülesandeid. Autori enda kogemustest lähtudes on pikem järjestikune keskendumine programmeerimise õppimisele väsitav, ka ülikooli tunniplaanis on programmeerimine enamasti kuni 4 akadeemilist tundi järjest. Samuti peaks sellest ajast olema piisav, et anda lühike ülevaade iOS operatsioonisüsteemist, selle jaoks arendamisest ja Xcode arenduskeskkonnast ning seejärel õpetada sammhaaval rakenduse loomist iOS operatsioonisüsteemi jaoks. Käsitletavad teemad annavad aluse iOS operatsioonisüsteemi jaoks rakenduse loomiseks, andes õppuritele võimaluse soovi korral ise edasi õppida.

Õppematerjali käigus arendatakse üks iOS operatsioonisüsteemile mõeldud rakendus valikulise funktsionaalsusega, andes õpitu kinnistamiseks võimaluse rakendus iseseisvalt lõpetada, täites õppematerjalis antud iseseisvad ülesanded. Kindlasti ei saa õpiprotsessi tulemusel täiesti selgeks Objective-C programmeerimiskeelt ega omandata kõiki võimalikke teadmisi iOS operatsioonisüsteemile rakenduse loomisest, kuid tutvustatakse arenduskeskkonda ja saadakse algteadmisi rakenduse loomiseks. Samuti autor loodab tekitada suuremat huvi õppematerjali läbijal, rakenduste loomise vastu iOS operatsioonisüsteemile.

Õppematerjali läbimise käigus loodava rakenduse valik tulenes autori eelnevast kogemustest iTunesU õpikeskkonnas oleva Stanfordi Ülikooli kursuse raames kalkulaatori loomisest ja aines "Veebirakenduste kasutajaliidesed" Javascripti programmeerimiskeeles objektorienteeritusest lähtudes kalkulaatori loomisest. Õppematerjali käigus loodav rakendus on kalkulaator, mille funktsionaalsuse ja ülesehituse valik tulenes õppematerjali läbimise ajalisest piirangust ja rakenduse loomise protsessi võimalikult huvitavaks tegemises. Selle rakenduse loomise käigus saab hästi tutvuda arenduskeskkonna ja rakenduse loomise võimalusega. Loodava rakenduse nimi Eurokalkulaator ja kasutajaliidese kujundus, oli inspireeritud 2010. aastal kõigile leibkondadele Eestis saadetud eurokalkulaatorist.

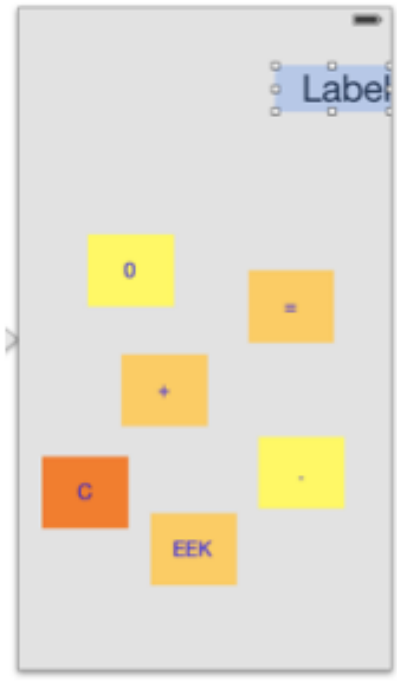
Õppematerjal luuakse pdf-failina, mida on võimalik vajadusel printida, kuid parem on seda kasutada arvutis. Pdf-faili saab võrdlemisi lihtsalt ning kiiresti tekstifailist tekitada ja selle avamiseks ei ole vaja erilist keskkonda või spetsiaalsed tarkvara nagu e-kursused. Samuti mõjutas valikut see, et peaaegu kõigis Tallinna Ülikooli Informaatika Instituudi bakalaureuseõppes programmeerimist õpetavates ainetes on õppematerjalid sellisel kujul ning samuti on nii Informaatika Instituudi bakalaureusetööde kui ka seminaritööde raames loodud eelnevalt mitmeid õppematerjale just sellisel kujul.

Õppematerjali sisu ülesehitamisel lähtutakse põhimõttest, et kõik uued osad on tehtud eraldi peatükkideks ja alampeatükkideks. Samuti üritatakse teha õppija jaoks kõik võimalikult lihtsaks ja vajadusel sisukorras üles leitavaks. Menüü punktid, tähtsamad mõisted ja failinimed on tekstist eristamiseks ja nendele tähelepanu juhtimiseks vormindati paksuks kirjaks. Õppematerjali käigus loodavate funktsioonide ja muutujate nimed on eesti keeles, et oleks võimalik paremini eristada muudetavat sisu Objective-C programmeerimiskeele osadest. Samuti peaks sihtrühm olema eestikeelsete muutujate ja funktsioonide kasutamisega juba tuttavad erinevatest programmeerimist õpetavatest ainetest Informaatika Instituudis. Selleks, et teha õppematerjal õppija jaoks lihtsamaks ja arusaadavamaks, on sinna lisatud palju koodinäiteid (Joonis 4) ja ekraanipildi jooniseid (Joonis 5). Koodinäited on lisatud tervikuna ühele lehele, et kopeerimisel leheküljenumbreid kaasa ei tuleks. Joonised on mahutatud teksti sisse, et materjal oleks mugavalt loetav ja kuna nad on väga heast kvaliteedist, saab neid vajadusel arvutist lähemalt vaadata. Joonistel olevad värvid võivad natuke erineda arenduskeskkonnas olevatest värvidest sõltuvalt kuvarite erinevusest.

```
@property (strong, nonatomic) NSString* teheMisOnMeeles;
```

Koodinäide 29 Uue muutuja teheMisOnMeeles defineerimine

Joonis 4 Koodinäide õppematerjalis



Joonis 29 Kitsama sildi lisamine

Joonis 5 Ekraanipilt õppematerjalis

Õppematerjali testitakse, et leida seal võimalikke vigu ja saada üldist tagasisidet õppematerjali kohta. Testijate valik on mugavusvalimi põhine, mis tugineb õppematerjali sihtrühma kirjeldusele. Testimine plaanitakse läbi viia kasutades õppematerjali autori arvutit. Tallinna Ülikooli Informaatika Instituudi Mac-laborit hetkel kasutada ei ole võimalik, sest Mac arvutitel on vanem operatsioonisüsteem ja arenduskeskkond, mida plaanitakse lähiajal uuendada.

3.2 Õppematerjali ülesehitus

Õppematerjali sissejuhatuses kirjeldatakse õppematerjali ülesehitust ja tähelepanekuid, kellele on õppematerjal mõeldud ja tutvustatakse ka vajalikke eelteadmisi õppematerjali paremaks omandamiseks.

Esimeses peatükis tutvustatakse iOS operatsioonisüsteemi, kui palju seda kasutatakse ja arendajatele pakutavaid võimalusi.

Teises peatükis tutvustatakse arenduskeskkonda Xcode ja sellega seotud rakendust iOS Simulator, Xcode’i kasutamiseks vajalikke süsteeminõudeid ja õpetatakse antud keskkonda Mac arvutisse paigaldama ning vajadusel ka uuendama.

Kolmandas peatüki alguses tutvustatakse õppematerjali läbimise käigus loodavat rakendust ja seejärel õpetatakse samm sammult rakenduse Eurokalkulaator loomist iOS operatsioonisüsteemile:

- luues uue projekti;
- tutvustades ja seadistades arenduskeskkonda;
- luues kasutajaliidese ja seda koodiga sidudes;
- luues klasse, muutujaid ja funktsioone, tagades sellega Eurokalkulaatori osalise funktsionaalsuse.

Peatüki lõpus antakse iseseisvad ülesanded, mille tegemisel saab õpitut kinnistada ja loodava rakenduse funktsionaalsust täiendada. Samuti on lisatud aadress, kust saab loodud rakenduse lähtekoodi vajadusel allalaadida ja projektina avada.

Õppematerjali kokkuvõttes on lühidalt kirjeldatud antud õppematerjalis loodut ja antud soovitus edasiõppimise võimaluse kohta.

Töö lõppu on lisatud poolelioleva (Lisa 1) ja valmis rakenduse klasside kood (Lisa 2) paremaks vigade parandamiseks õppimise käigus.

3.3 Õpiteede tutvustus

Õppematerjali on soovitav kasutada arvutis avatuna, sest see võimaldab vajadusel kiiremini teksti seest otsida ja ekraanipiltide jooniseid lähemalt vaadata ning on keskkonnasäästlikum kui õppematerjali paberi peale välja trükkimine.

Õppematerjali sissejuhatavas osas tutvustatakse õppematerjali sisu ja vajalikke eelteadmisi, mille kaudu saab igaüks määrata, kas õppematerjal on talle sobilik ja kas soovitakse antud teemat õppida.

Õppematerjali kaks esimesed peatükki “iOS operatsioonisüsteem” ja “Arenduskeskkond Xcode” on soovitatav mõttega läbi lugeda. Kindlasti paljudele on uueks informatsiooniks arendajatele mõeldud programmid ja võib tekkida huvi arendaja konto loomise vastu. Teise peatüki puhul on kindlasti vaja tutvuda arendaja arvuti süsteeminõuetega, et üldse oleks võimalik õppematerjali kasutada ja juhul kui õppimiseks kasutatavas Mac arvutis pole arenduskeskkonda paigaldatud või vajab see uuendamist, võib see aeglase allalaadimiskiirusega Interneti puhul aega võtta.

Kolmas peatükk, mis tervenisti keskendub rakenduse loomisele, on soovitatav selle esimesest osast alustades põhjalik läbi töötada. Juhul kui tekib probleeme ja vigu koodis, on mõistlik uuesti läbi vaadata läbitöötatud peatükid ja vajadusel olenevalt, kas ollakse peatükist numbriga 3.7.8 eespool või mitte, on võimalik õppematerjali lisadest leida vastavalt poolelioleva ja valmis rakenduse klasside koodid, kust õppija saab vigade parandamiseks seda koodi enda loodud koodiga võrrelda.

3.3.1 Õppimise jätkamise võimalused

iOS operatsioonisüsteemi jaoks rakenduse loomise edasiseks õppimiseks ei õnnestunud leida eestikeelseid õppematerjale, kuid huvi korral soovitab õppematerjali autor läbida tasuta Stanfordi Ülikooli inglisekeelse kursuse nimega **Developing iOS 7 Apps for iPhone and iPad**. Kursuse materjal on kättesaadav õpikeskkonnast **iTunesU** ja aadressil https://itunes.apple.com/us/course/developing-ios-7-apps-for/id733644550.

3.4 Õppematerjali testimine

Õppematerjali kohta sihtrühmalt tagasiside saamiseks ja võimalike vigade avastamiseks viidi läbi õppematerjali testimine, mille käigus paluti testitavatel iseseisvalt läbida õppematerjal. Pärast testimist tuli testijatel vastata küsitluse (Lisa 2), millest saadud tulemused on käesoleva töö lisas (Lisa 3).

Testijate valim moodustati vastavalt õppematerjali sihtrühmale. Õppematerjali testijad olid Tallinna Ülikooli Informaatika Instituudi bakalaureuseõppe 3. kursuse üliõpilased. Testimine viidi läbi 9. jaanuaril 2014 Tallinna Ülikooli Informaatika Instituudis kolme üliõpilasega, kes olid nõus eksamite kõrvalt testimise jaoks aega leidma. Testijad olid läbinud ained “IFI6069 Programmeerimise põhikursus” ja “IFI6083 Algoritmid ja andmestruktuurid” ning neil oli õppematerjali läbimiseks piisav varasem programmeerimiskogemus, kuid ükski testija polnud ennem Mac arvutit ja OS X operatsioonisüsteemi töövahendina kasutanud.

Pärast testimist läbi viidud küsitluse tulemustes analüüsist on võimalik järeldada, et õppematerjal oli testijate jaoks arusaadavaks ja loogiline. Koodinäiteid ja ekraanipilte loeti õppematerjalis piisavaks ning õppematerjali peeti läbitavaks 4 akadeemilise tunniga. Kümne punkti skaalas, kus 10 tähistab, et õppematerjal oli väga huvitav ja 0, et ei olnud üldse huvitav, hinnati õppematerjali 8, 9 ja 8, mis on õppematerjali autori arvates väga hea tulemus. Kõik testimises osalejad soovitaksid õppematerjali ka teistele, kellel on huvi

rakenduste loomise vastu iOS operatsioonisüsteemile ja isegi soovi korral Objective-C ning mugava Mac OS X operatsioonisüsteemile mõeldud arenduskeskkonnaga tutvumiseks.

Testimine läks autori arvates edukalt. Lähtudes tagasiside küsitluse analüüsist ja kommentaaridest, suusõnalisest ja kirjalikust tagasisidest, tehti õppematerjalis järgmised muudatused:

- lisati siltide taustavärvi muutmise ja sildi teksti joondamise osa seletus vormindati paksu kirjaga;
- ekraani lõuendile lisatava nupu sisuveaga vale värvikood muudeti kujult RGB 255, 102, 102 kujule RGB 255, 255, 102 ja lisati värviakna sulgemise seletus;
- nuppude vajutamistele käivitavate funktsioonide järjekorra valikulisus sai selgitatud.
- õppematerjali sissejuhatusse lisati soovitus piltide lähemalt vaatamiseks ja seletati õppematerjalis olevate joonistel võimalikku värvide erinevust arenduskeskkonnas olevatega sõltuvalt kasutatavast kuvarist;
- parandati üksikud kirjavead;
- paigutati ümber koodinäited, et leheküljenumbrid kopeerimisel kaasa ei tuleks;
- lisati seletus muutujanimede juurde kuuluva tärni (*) olemusest;
- lisati seletus kahe sildi kasutamise põhjus kalkulaatori välja loomisel.

Kokkuvõte

Seminaritöö eesmärk oli luua programmeerimiskogemusega huvilistele iseseisvaks õppimiseks eestikeelne õppematerjal rakenduse loomisest iOS operatsioonisüsteemi jaoks.

Seminaritöö eesmärgi saavutamiseks uuris autor iOS operatsioonisüsteemi ja arenduskeskkonda Xcode, andis ülevaate koostatava õppematerjali nõuetest ja ülesehitusest, tõi välja võimalikud õpiteed, viis läbi testimise ja küsitluse õppematerjali sihtrühmaga ning analüüsis saadud tagasidet.

Seminaritöö raames õppematerjali koostamine oli kasulik ka töö autorile, suurendades juba enne olnud huvi õpetamise vastu ja andes suurema julguse õppematerjalide koostamiseks ka tulevikus.

Õppematerjali testimise eesmärgiks oli sihtrühmalt tagasiside saamine ja võimalike vigade avastamine. Testimise käigus lasti testitavatel iseseisvalt läbida õppematerjal ja seejärel vastata tagasiside küsitlusele, mille analüüsist võib üldistades järeldada, et õppematerjal on arusaadav ja huvitav. Testimise käigus välja tulnud mõningad soovitused ja üksikud vead võeti arvesse ning õppematerjali muudeti paremaks.

Käesoleva seminaritöö raames koostatud õppematerjal on allalaetav aadressilt http://romil.ee/seminaritoo/oppematerjal_rakenduse_loomiseks_iOS_operatsioonisusteemiga_seadme_jaoks.pdf.

Loodud õppematerjali edasiarendamiseks on võimalik uurida iOS 7 SDK võimalusi, tuues välja mobiilse seadme tähtsamate funktsionaalsuste kasutamist ja selle kaudu luua õppematerjal keerulisema rakenduse loomiseks, näiteks kasutades mobiilse seadme kaamerat või andmebaasi andmete hoiustamiseks. Lisaks on võimalik uurida keerulisemate rakenduste kasutajaliideseid ja seejärel tuua välja parimad praktikad.

Kasutatud kirjandus

Apple Inc. (2013, juuni 10). *Apple Unveils iOS 7*. Retrieved jaanuar 1, 2014, from Apple: http://www.apple.com/pr/library/2013/06/10Apple-Unveils-iOS-7.html

Apple Inc. (2013). *Choosing an iOS Developer Program*. Retrieved jaanuar 1, 2014, from Apple: https://developer.apple.com/programs/start/ios/

Apple Inc. (2013, oktoober 22). *iOS Human Interface Guidelines.* Retrieved jaanuar 1, 2014, from Apple: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf

Apple Inc. (2013, oktoober 22). *iOS Simulator Guide.* Retrieved jaanuar 1, 2014, from Apple: https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/iOS_Simulator_Guide.pdf

Apple Inc. (2013, september 18). *iOS Technology Overview.* Retrieved detsember 27, 2013, from iOS Developer Library: https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/iOSTechOverview.pdf

Apple Inc. (2013, oktoober 22). *Start Developing iOS Apps Today.* Retrieved jaanuar 1, 2014, from Apple: https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/RoadMapiOS.pdf

Hughes, N. (2013, detsember 31). *iOS 7 now installed on 78% of active Apple handheld devices*. Retrieved jaanuar 1, 2014, from Apple Insider: http://appleinsider.com/articles/13/12/31/ios-7-now-installed-on-78-of-active-apple-handheld-devices

IDC. (2013, november 12). *Android Pushes Past 80% Market Share While Windows Phone Shipments Leap 156.0% Year Over Year in the Third Quarter, According to IDC*. Retrieved jaanuar 1, 2014, from IDC: http://www.idc.com/getdoc.jsp?containerId=prUS24442013

Lisad

Lisa 1 Õppematerjal

Lisa 2 Testimise tagasiside küsitluse ankeet

Lisa 3 Testimise tagasiside küsitluse tulemused

Lisa 1 Õppematerjal

Tallinna Ülikool

Informaatika Instituut

Romil Rõbtšenkov

Õppematerjal rakenduse loomiseks iOS operatsioonisüsteemiga seadme jaoks.

Tallinn 2014

Sisukord

Sissejuhatus

Käesolev õppematerjal on koostatud seminaritöö “Rakenduse loomine iOS operatsioonisüsteemiga seadme jaoks.” raames ja sobib programmeerimiskogemusega huvilistele, kes soovivad iseseisvalt omandada iOS operatsioonisüsteemi jaoks rakenduse loomise esmased oskused. Õppematerjali paremaks omandamiseks on soovitatav eelteadmiste olemasolu klassidest, pinudest ja programmeerimiskeelest C. Lähtudes eelteadmiste ja varasema programmeerimiskogemuse olemasolust sobib õppematerjal väga hästi Tallinna Ülikooli Informaatika Instituudi bakalaureuseõppe kolmanda kursuse üliõpilastele iseseisvaks õppimiseks. Õppematerjali eesmärk on anda lühike ülevaase iOS operatsioonisüsteemist, selle jaoks arendamisest ja Xcode arenduskeskkonnast ning seejärel õpetada sammhaaval rakenduse loomist iOS operatsioonisüsteemi jaoks.

Esimeses peatükis tutvustatakse iOS operatsioonisüsteemi, kui palju seda kasutatakse ja arendajatele pakutavaid võimalusi.

Teises peatükis tutvustatakse arenduskeskkonda Xcode ja sellega seotud rakendust iOS Simulator, Xcode’i kasutamiseks vajalikke süsteeminõudeid ja õpetatakse antud keskkonda Mac arvutisse paigaldama ning vajadusel ka uuendama.

Kolmandas peatüki alguses tutvustatakse õppematerjali läbimise käigus loodavat rakendust ja selle funktsionaalsust. Seejärel õpetatakse samm sammult rakenduse loomist iOS operatsioonisüsteemile. Peatüki lõpus antakse iseseisvad ülesanded, mille tegemisel saab õpitut kinnistada ja loodava rakenduse funktsionaalsust täiendada.

Õppematerjal sisaldab rohkesti ekraanipilte ja koodinäiteid, et oleks lihtsam õppida ja õpitav oleks arusaadavam. Joonised on mahutatud teksti sisse, et materjal oleks mugavalt loetav, aga kuna nad on väga heast kvaliteedist, saab neid vajadusel arvutist lähemalt vaadata. Joonistel olevad värvid võivad natuke erineda arenduskeskkonnas olevatest värvidest sõltuvalt kuvarite erinevusest.

Töö lõppu on lisatud poolelioleva (Lisa 1) ja valmis rakenduse klasside kood (Lisa 2) paremaks vigade parandamiseks õppimise käigus.

Õppematerjali autor soovib edukat õppimist!

1 iOS operatsioonisüsteem

iOS on Apple'i poolt loodud operatsioonisüsteem, mis on kasutusel seadmetel iPhone, iPad, ja iPod Touch. See haldab seadmete riistvara ja varustab tehnoloogiaga, mis on vajalik seadme jaoks loodavatele rakendustele. Operatsioonisüsteemi on eelpaigaldatud hulgaliselt Apple'i poolt loodud rakendusi nagu Telefon, Kalender, E-post, Safari, Kaamera, Muusika jpt, mis võimaldavad kasutajale nutitelefonile omaseid funktsionaalsusi (Apple Inc, 2013).

1.1 iOS kasutajad

Turuuuringutele, analüüsile ja konsulteerimisele keskendunud, Ammerika Ühendriikides asuv ettevõte, International Data Corporation (IDC) avaldas oma pressiväljaandes 2013. aasta kolmandas kvartalis populaarsemate operatsioonisüsteemide jaotuse ülemaailmsel mobiiliseadmete turul. Välja toodud andmete järgi on teiste operatsioonisüsteemidega võrreldes populaarsemad iOS, turuosaga 14.4 % ja Android turuosaga 74.9 % (IDC, 2013).

Eesti turundusfirma smartAD veebipõhise vidina nimega Addresstising raamat[1] andmete järgi on Eestis 65303 - 72559 Android operatsioonisüsteemi kasutavat mobiiliomanikku ning iOS operatsioonisüsteemi kasutavad vastavalt 38809 - 43133 iPad tahvelarvuti ja 37106 – 41229 iPhone'i omanikku. Android operatsioonisüsteemiga tahvelarvutite kasutajate arvu ei ole välja toodud. Nende andmete põhjal saab järeldada, et iPhone'i omanikke on Eestis palju ja suure kasutajaskonnaga iOS operatsioonisüsteemile rakenduste loomise oskus on väga oluline.

1.2 iOS 7

iOS 7 on õppematerjali loomise ajal kõige uuem iOS operatsioonisüsteemi versioon, mida tutvustati avalikkusele 2013. aasta juunis. Apple'i poolt läbi viidud uuringu järgi kasutatakse 2013. aasta detsembri seisuga operatsioonisüsteemi iOS 7 ülekaalukalt 78%, iOS 6 kasutatakse 18% ja kõiki vanemaid versioone kokku kasutatakse 4%. Antud statistika näitab, et üleminek uuele operatsioonisüsteemile on olnud kiire ja kasutajad on uue funktsionaalsuse omaks võtnud (Hughes, 2013).

[1] Veebipõhine vidin **Addresstising raamat** on saadaval aadressil http://smartad.eu/estonia/addresstising_book/

Võrreldes vanemate iOS operatsioonisüsteemidega, on iOS 7 kasutajaliidese uuendusliku kujundusega (Joonis 1) ja pakub kasutajatele uusi omadusi ja funktsionaalsust nagu Juhtimiskeskus (*Control Center*), Teavituskeskus (*Notification Center*) jpt (Apple Inc, 2013).

Joonis 1 iOS 7 (Apple Inc, 2013)

1.3 Arendamine iOS 7 operatsioonisüsteemi jaoks

Apple on loonud iOS 7 operatsioonisüsteemile rakenduste arendajatele eraldi veebikeskkonna, tööriistad ja materjalid, tehes sellega nii arendamise kui ka õppimise võimalikult lihtsaks. Teatud materjalidele ligipääsu saamiseks ja App Store'i kaudu valmisrakenduste levitamiseks, on vaja aga kindlasti liituda ühe arendajatele mõeldud tugiprogrammiga.

1.3.1 iOS 7 SDK

iOS Software Development Kit (SDK) sisaldab endas erinevaid tööriistu, mis on vajalikud *native* tüüpi rakenduste arendamiseks, seadmetele paigaldamiseks, käivitamiseks ja ka testimiseks. *Native* tüüpi rakendused on loodud kasutades iOS operatsioonisüsteemile mõeldud teeke ja Objective-C programmeerimiskeelt ning töötavad otseselt iOS operatsioonisüsteemil. Erinevalt veebipõhistest rakendustest, on *native* tüüpi rakendused paigaldatud füüsiliselt seadmele ja on alati kasutajale kätte saadavad, isegi juhul kui seade on Lennukirežiimis. Kõik rakendused salvestatakse koos võimalike andmetega kasutaja kohta arvutisse, kasutades selleks programmi iTunes (Apple Inc, 2013).

1.3.2 Reeglid rakenduste kasutajaliidese kujundamiseks

Selleks, et iOS operatsioonisüsteemi jaoks tehtud rakendustel oleks hästi toimiv kasutajaliides, on Apple loonud dokumendi nimega OS Human Interface Guidelines[2], kus on arendajate jaoks täpselt kirjeldatud kasutajaliidese disaini eeskirjad. Lisaks sisaldab see piisavalt nõuandeid hea kasutajaliidese loomiseks, seal on kirjeldatud kuidas tuleb integreerida sotsiaalmeediat ja milliseid valmisobjekte on võimalik kasutajaliidese loomisel kasutada, millest tuleb lähtuda ikoonide kujundamisel ning tutvustatakse ka kuidas rakenduste loomisel saab arvestada vaegnägijatega (Apple Inc, 2013).

1.3.3 iOS Dev Center

iOS Dev Center[3] on veebikeskkond, kus arendajatel on võimalik leida Apple'i poolt loodud dokumentatsiooni ja muid materjale, mis arendamisel kasuks tulevad. Samuti on alamlehtedena olemas arendajatele mõeldud foorum ja muud abilehed. Sarnased veebikeskkonnad on loodud Mac OS jaoks rakenduste ja Safari jaoks pistikprogrammide arendajatele.

1.3.4 iOS arendajatele mõeldud tugiprogrammid

Apple on loonud arendajatele mõeldud kolm erinevat programmi, mis võimaldavad arendajatele peale nendega liitumist erinevaid hüvesid.

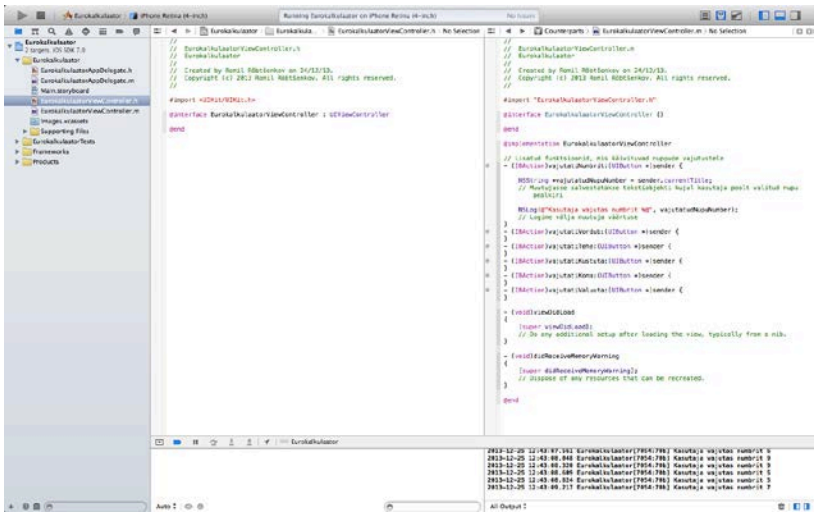
- **iOS Developer Program** on kõige tavalisem programm annab 99 dollari eest aastas võimaluse arendatud rakendusi testida päris seadmetel ja valmisrakendusi levitada App Store'i kaudu.
- **iOS Developer Enterprise Program** võimaldab 299 dollari eest aastas arendada ja levitada rakendusi ettevõte, avalike asutuste, õppeasutuste või muude asutuse siseselt.
- **iOS Developer University Program** on tasuta ja on mõeldud kraadiõpet või kõrgharidust võimaldavatele õppeasutustele, andes nende tudengitele võimaluse rakendusi päris seadmetel testida (Apple Inc, 2013).

[2] iOS Human Interface Guidelines on saadaval aadressil https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf
[3] iOS Dev Center on saadaval aadressil https://developer.apple.com/devcenter/ios/index.action

Viimase kirjeldatud programmiga, iOS Developer University Program, on liitunud Tallinna Ülikooli Informaatika Instituudi Haridustehnoloogiakeskus. Antud õppematerjali kasutamiseks ei ole vajalik ühegi arendajatele mõeldud programmiga liituda.

2 Arenduskeskkond Xcode

Xcode on Apple'i poolt loodud arenduskeskkond, mis võimaldab projektihaldust, koodi redigeerimist, kompileerimist, vigade otsimist, failide versioonihaldust, testimist jpm. Arenduskeskkonna keskse osa moodustab Xcode nimeline rakendus (Joonis 2), mis pakub koodi redigeerimise võimalust ja ligipääsu kõikidele teistele tööriistadele. Tähtsamad rakendused, mis Xcode arenduskeskkonda veel kuuluvad, on jõudluse ja teiste tegurite testimist võimaldav rakendus Instruments ja mobiilset seadet simuleeriv iOS Simulator. Xcode on tasuta allalaetav Mac arvutile App Store-ist (Apple Inc, 2013).



Joonis 2 Arenduskeskkond Xcode

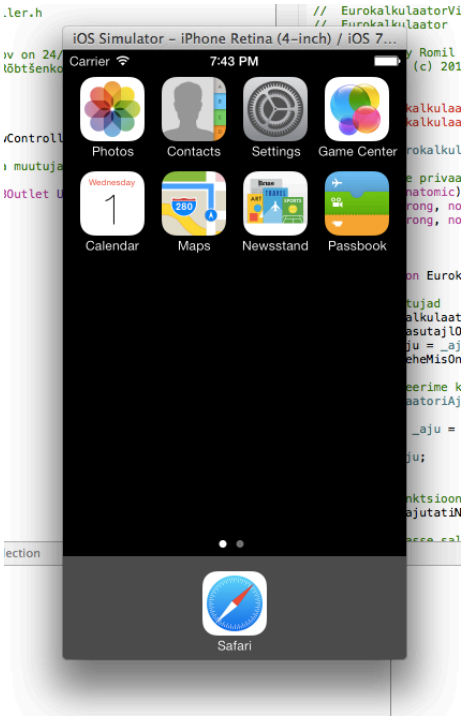
2.1 Süsteeminõuded

Arendaja arvuti ja operatsioonisüsteemi piirang on Apple'i poolt küllaltki range, nõudes arendaja arvutiks kindlasti Apple'i poolt loodud Mac arvutit ja OS X operatsioonisüsteemi. Ametlikult puuduvad nii arendustarkvara kui ka tugi, et arendamisel kasutada teisi arvuteid ja Microsoft Windows operatsioonisüsteemi. Selleks, et arendada rakendust iOS 7 operatsioonisüsteemile on vaja täita järgmised nõuded:

- Apple'i Mac arvutil peab olema operatsioonisüsteem OS X 10.7 (Lion) või uuem;
- arvutisse peab olema paigaldatud arenduskeskkond Xcode 5;
- arenduskeskkonnas peab olema iOS 7 SDK (Apple Inc, 2013).

2.2 iOS Simulator

iOS Simulator võimaldab kiiret prototüüpimist ja loodud rakenduste jooksvaks testimiseks kogu arendusprotsessi käigus (Joonis 3). Antud rakendus on üks Xcode arenduskeskkonna osa koos iOS SDK-ga ja töötab nagu iga teine Mac rakendus simuleerides iPhone või iPad seadet, võimaldades testida arendatavat rakendust ka erinevate iOS versioonidega (Apple Inc, 2013).



Joonis 3 iOS Simulator rakendus

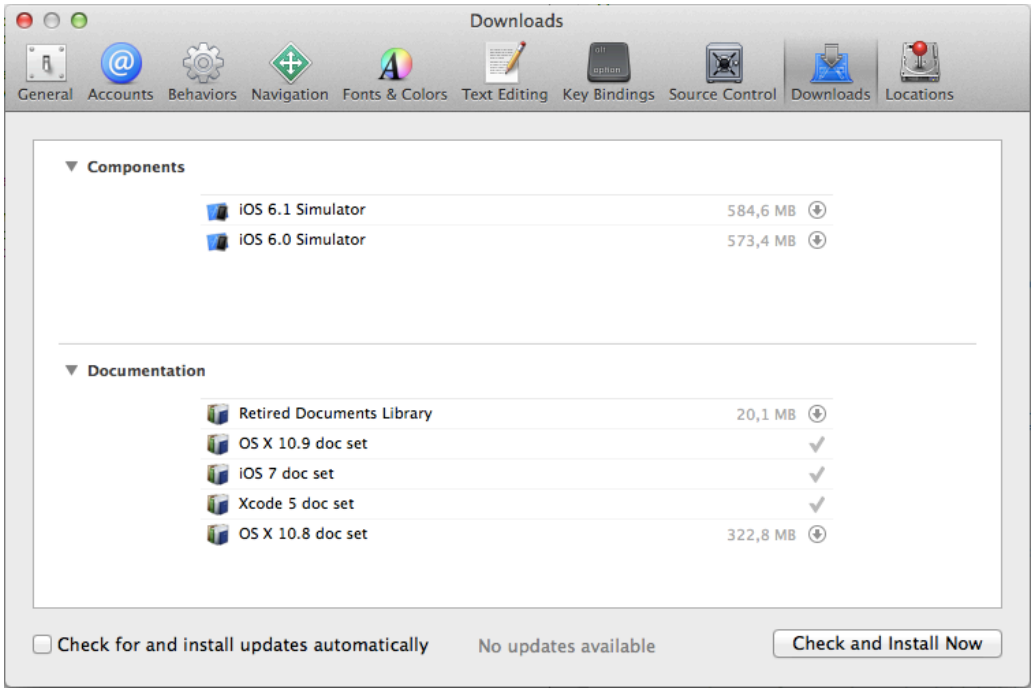
2.3 Arenduskeskkonna Xcode 5 paigaldamine

Kõige uuema Xcode arenduskeskkonna saab allalaadida käivitades Mac arvutiga rakenduse App Store, mis on 10.7 või hilisema Mac OS versiooniga eelpaigaldatud ja sealt üles otsida ning paigaldada tasuta kättesaadav rakendus Xcode (Joonis 4). iOS SDK paigaldatakse koos Xcode arenduskeskkonnaga (Apple Inc, 2013).

Joonis 4 Rakenduse Xcode ikoon

2.4 Arenduskeskkonnas Xcode uuenduste kontrollimine

Arenduskeskkonnas Xcode uuenduste kontrollimeks tuleb see alustuseks käivitada. Seejärel menüüst tuleb valida **Xcode > Preferences…**. Avanenud dialoogiaknas tuleb valida **Downloads** vahekaart ja seejärel uuenduste kontrollimiseks ning paigaldamiseks vajutada **Check and Install Now**. Juhul kui kõik vajalikud uuendused on paigaldatud, kuvatakse dialoogiakna alumises servas hall kiri **No updates available** (Joonis 5), vastasel juhul paigaldatakse uuendused.



Joonis 5 Uuenduste paigaldamise dialoogiaken

3 Rakenduse arendamine

Käesoleva õppematerjali käigus arendatakse rakendus osalise funktsionaalsusega, andes õpitu kinnistamiseks võimaluse rakendus iseseisvalt lõpetada, täites õppematerjalis antud iseseisvad ülesanded.

Enne rakenduse arendama asumist on soovitatav veenduda arenduskeskkonna Xcode ja SDK versiooni vastavust nõuetele (2.1 Süsteeminõuded) ja vajadusel kontrollida uuenduste olemasolu ning need ka paigaldada (2.4 Arenduskeskkonnas Xcode uuenduste kontrollimine)!

3.1 Loodava rakenduse tutvustus

Loodava rakenduse ideeks on luua kalkulaator, millel oleks lisafunktsionaalsus muuta arvutamisel saadud tulemus soovi korral eurodeks või kroonideks. Õppematerjali käigus loodava rakenduse osaline funktsionaalsusena piirdub liitmise ja kustutamise võimaldamisega. Ülejäänud funktsionaalsuse tagamine iseseisvateks ülesanneteks, mis on eraldi välja toodud peatükis 3.8 Iseseisvad ülesanded.

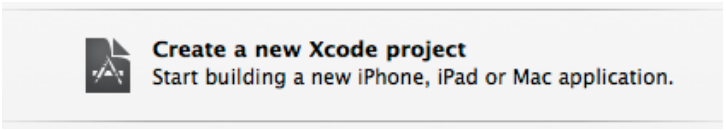
Loodava rakenduse nimeks on valitud Eurokalkulaator (Joonis 6).

Joonis 6 Eurokalkulaatori rakendus valmiskujul

Rakenduse arendamisel luuakse kaks erineva funktsionaalsusega klassi, õpitakse kasutama pinu, visuaalse poole sidumisel koodiga, luuakse kasutaja nupuvajutustele käivituvad funktsioonid ning mitmeid teisi abifunktsioone koos muutujatega.

3.2 Projekti loomine

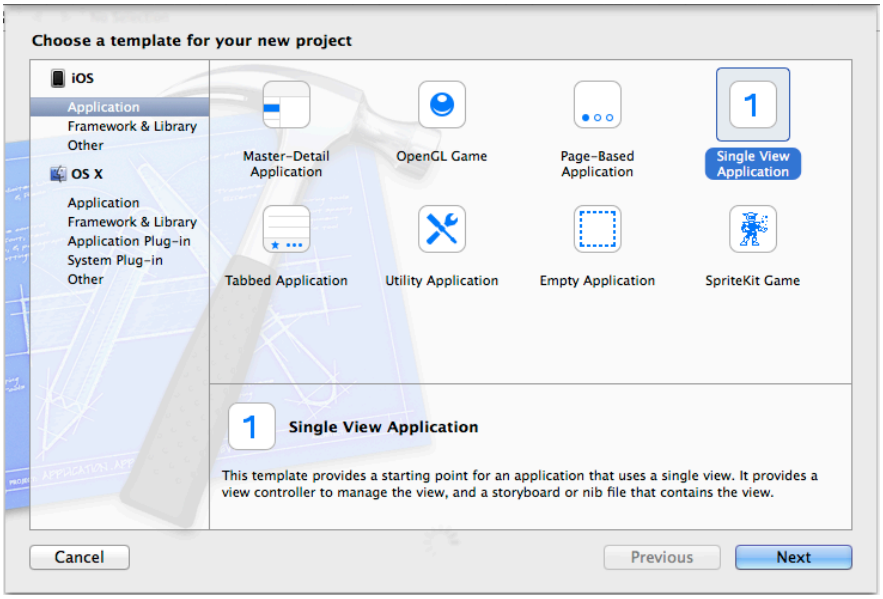
Uue projekti loomiseks tuleb käivitada arendamiskeskkond **Xcode**. Käivitumisel avanenud aknast tuleb valida **Create a new Xcode project** (Joonis 7) või menüüst **File > New Project**.



Joonis 7 Uue projekti alustamine

Kuna antud arenduskeskkond võimaldab ka arendada rakendusi operatsioonisüsteemi OS X jaoks, on vaja täpsustada loodava rakenduse tüüp ja soovi korral ka valida vajaminevaid teeke. Vastavalt valikutele luuakse projekti mall. Eurokalkulaatori rakendus on iOS operatsioonisüsteemi jaoks ja ühe vaatega.

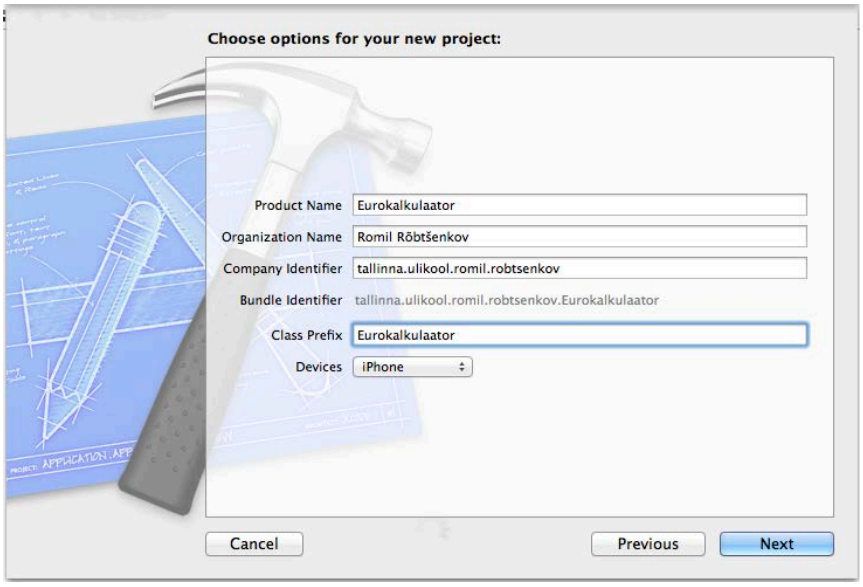
Malli valiku aknast tuleb valida **iOS** alt **Application**, seejärel **Single View Application** ja valiku kinnitamiseks tuleb vajutada **Next** (Joonis 8).



Joonis 8 Malli valimine

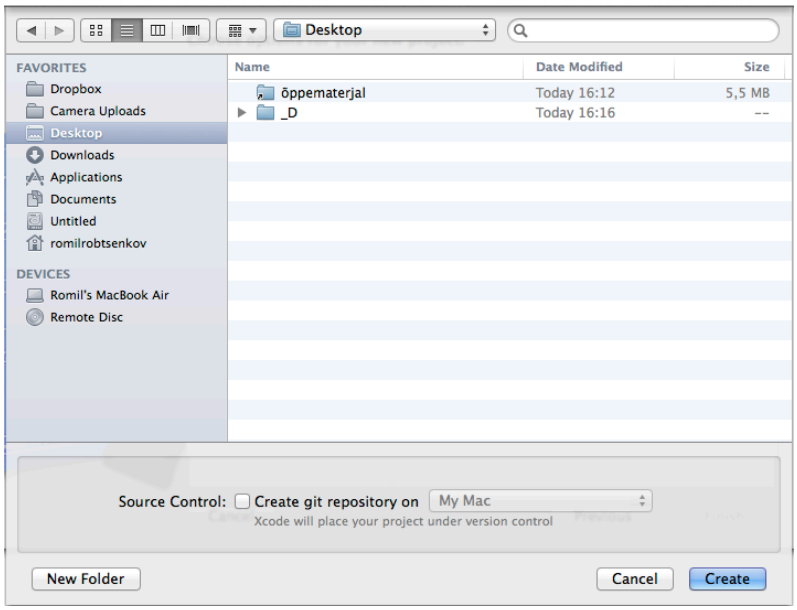
Projekti valikute alt tuleb täita kõik väljad. **Product Name** on loodava rakenduse nimi **Eurokalkulaator**. **Organization Name** on üldjuhul **arendaja nimi**. **Company Identifier** on rakenduste eristamiseks mõeldud rida, mis peab olema piisavalt pikk, et kindlasti erineda kõikidest teistest arendajatest, näiteks **tallinna.ulikool.eesnimi.perenimi**. **Class Prefix** on automaatselt loodavate klasside eesliide, enamasti on see sama rakenduse

nimega, antud näites **Eurokalkulaator**. **Devices** alt tuleb teha valik **iPhone**, sest antud rakendus on mõeldud ainult iPhone nutitelefonidele (Joonis 9). Projekti loomisega edasi liikumiseks tuleb vajutada **Next**.



Joonis 9 Projekti valikud

Järgmise tuleb valida projektifailide jaoks loodava kausta soovitav asukoht arvutis. Valitud asukohta luuakse uus kaust rakenduse nimega. Projekti põhifailiks on alati loodavas kaustas **.xcodeproj** laiendiga rakenduse nimega fail. Antud näites salvestatakse projektifaile sisaldav kaust töölauale. **Create git repository** valik tuleb jätta märkimata, sest antud õppematerjalis versioonihaldust arenduskeskkonnas Xcode ei tutvustata. Valikute kinnitamiseks tuleb vajutada **Create** (Joonis 10).



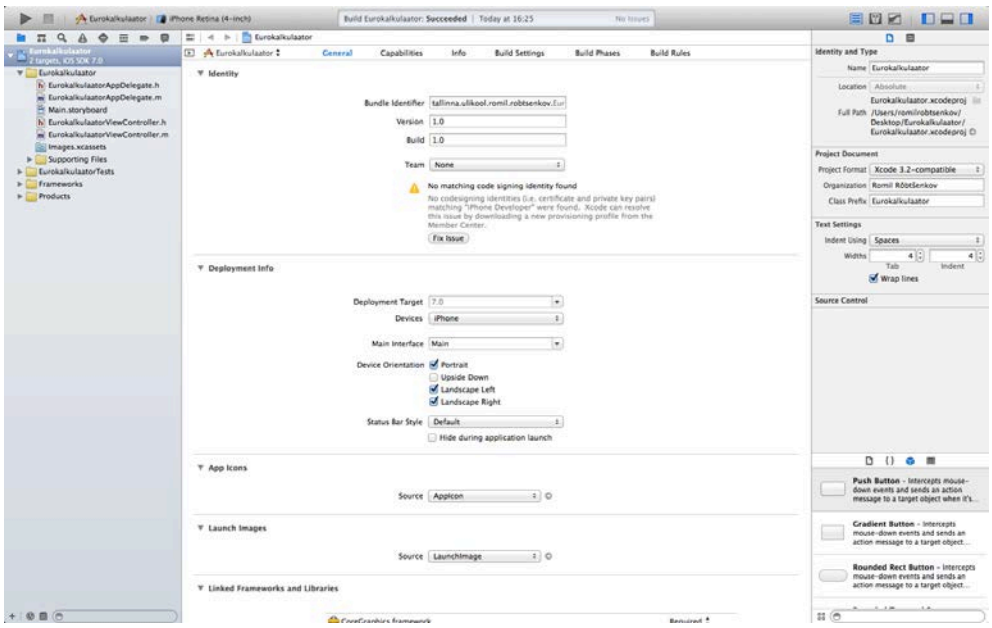
Joonis 10 Projektifailide salvestuskoha valimine

3.3 Arenduskeskonna tutvustus ja seadistamine

Selleks, et antud õppematerjalis loodavat rakendust oleks mugavam ja kiirem arendada, on vaja enne arendama asumist tutvuda arenduse käigus kasutatavate ikoonidega ja kindlasti seadistada arenduskeskkond sobivalt.

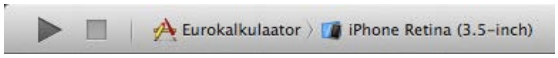
3.3.1 Tutvustus

Projekti loomisel käivitub arenduskeskkond (Joonis 11).



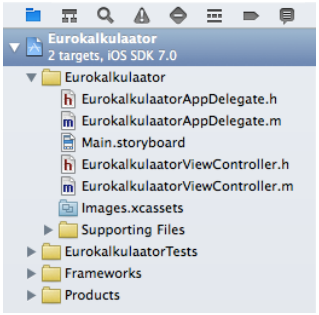
Joonis 11 Arenduskeskkond

Arenduskeskkond on väga funktsioonirohke, kuid antud õppematerjalis tutvustakse ainult hädavajalikku antud rakenduse mugavaks arendamiseks. Rakenduse käivitamine simulaatoris ja simulaatori valik on programmi põhiaknas vasakul üleval (Joonis 12).



Joonis 12 Rakenduse käivitamine ja simulaatori valik

Põhiaknas sellest allapoole jääb vasak küljepaneel, mis näitab hierarhiliselt projektis kasutusel olevaid faile (Joonis 13).

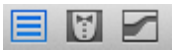


Joonis 13 Failihaldur

Küljepaneelide avamiseks ja peitmiseks on vastavad ikoonid põhiaknas paremal üleval (Joonis 14).

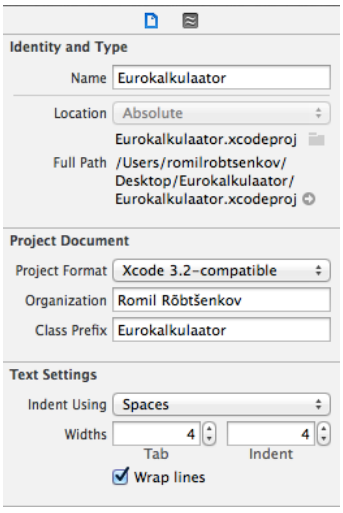
Joonis 14 Küljepaneelide avamise ja peitmise ikoonid

Nendest ikoonidest vasakul on vaadete muutmiseks olevad ikoonid (Joonis 15). Pidulikku riietust visualiseeriv keskmine ikoon **Assistant editor** jagab põhivaate vertikaalselt kaheks. Tavavaatesse naasmiseks tuleb vajutada kolmest kõrvutiolevast ikoonist vasakpoolset **Standard editor**. Kõige parempoolsem ikoon, mida antud õppematerjalis ei kasutata, on versioonihalduse puhul erinevate sama faili versioonide võrdlemiseks.



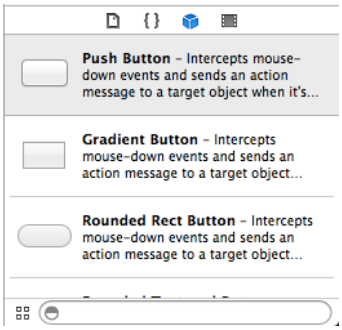
Joonis 15 Vaadete muutmise ikoonid

Paremas küljepaneelis on võimalik vastavalt valitud failile või objektile muuta selle parameetreid (Joonis 16).



Joonis 16 Objekti parameetrid

Parema küljepaneeli alumises osas on valik valmisobjekte, mida saab loodavasse rakendusse lisada (Joonis 17).



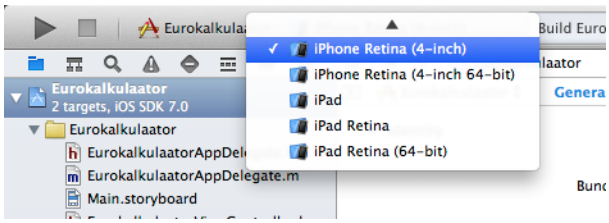
Joonis 17 Valmisobjektide loend

3.3.2 Seadistamine

Soovitatav on teha põhiaken täisekraani suuruseks vajutades paremal üleval nurgas olevale ikoonile (Joonis 18). Täisekraani režiimist saab väljuda vajutades samale nupule.

Joonis 18 Täisekraani režiimi sisselülitamise ikoon

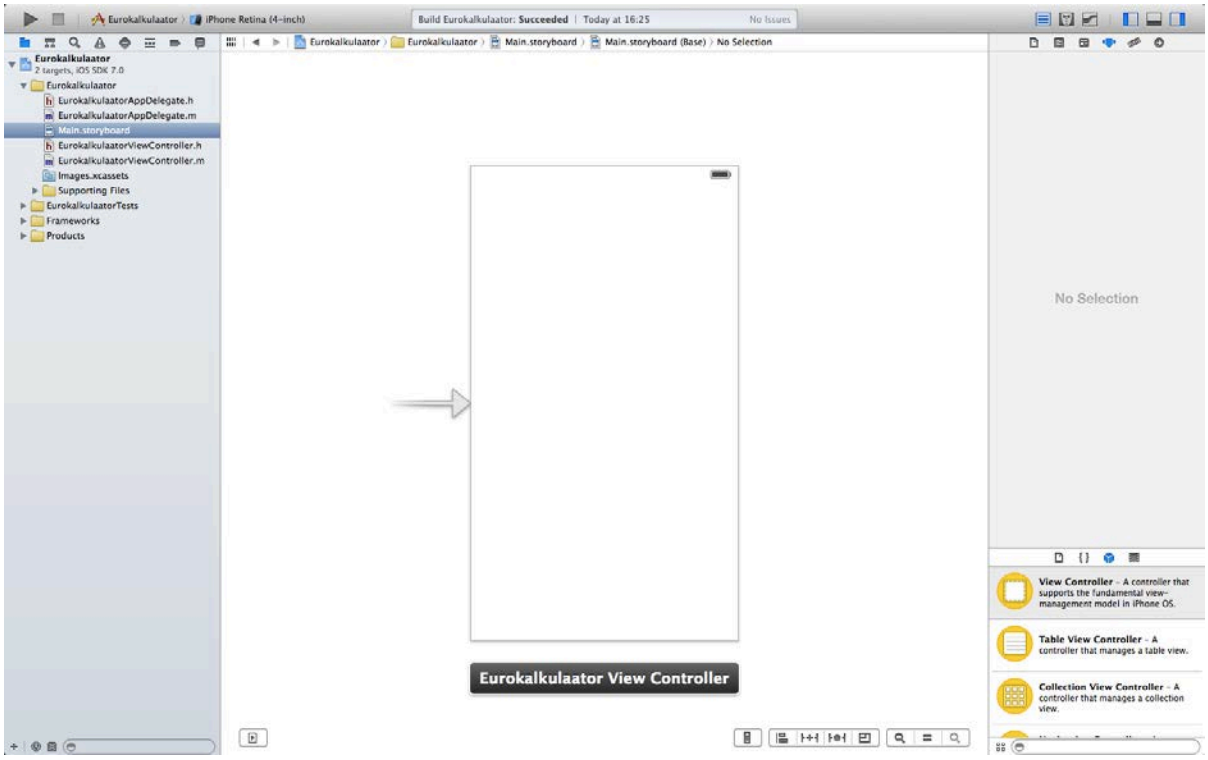
Loodav rakendus on mõeldud ainult uuematele iPhone nutitelefonide, millel on võrreldes vanemate mudelitega suurem ekraan. Simulaatorite valikust tuleb valida **iPhone Retina (4-inch)** (Joonis 19).



Joonis 19 Simulaatori valik

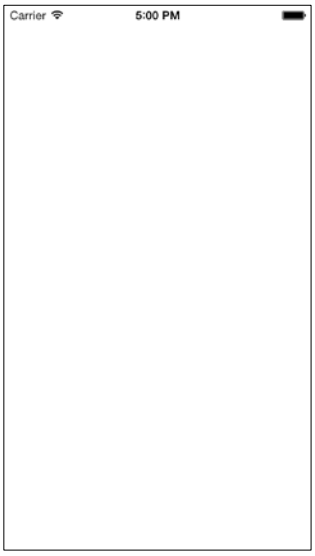
3.4 Eurokalkulaatori kasutajaliidese loomine

Alustuseks tuleb failihalduri küljepaneelist valida fail nimega **Main.storyboard** (Joonis 20).



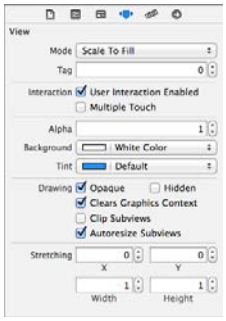
Joonis 20 Põhiakna vaade pärast faili Main.storyboard valimist

Rakenduse käivitamisel, kasutades kolmnurga kujulist ikooni põhiakna vasakul üleval nurgas, käivitatakse simulaator eraldi aknas. Rakendus on võimalik käivitada simulaatoris ka klahvikombinatsiooniga **cmd⌘ + r.** Kuna hetkel on rakendus täiesti tühi, siis selle käivitamisel, kuvab simulaator valget ekraani (Joonis 21).



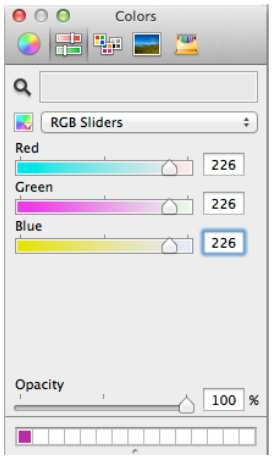
Joonis 21 Rakenduse vaade simulaatoris

Esimese sammuna tuleb valida telefoniekraani kujutav lõuend põhivaates, mille tagajärjel peaks parem küljepaneel kuvama valitud lõuendi parameetreid (Joonis 22).



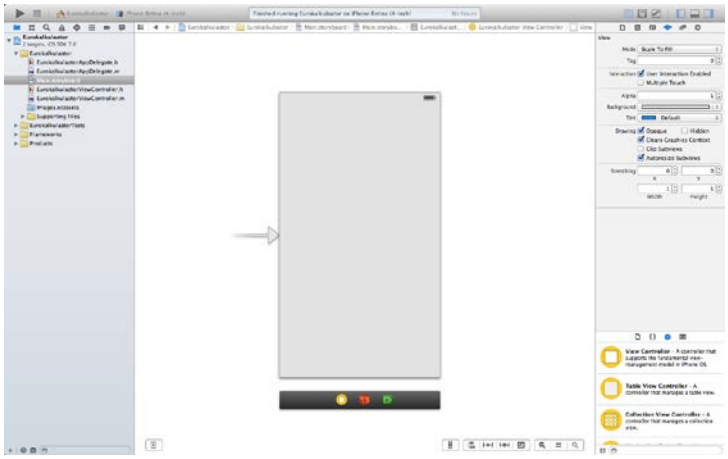
Joonis 22 Telefoniekraani lõuendi parameetrid

Paremas küljepaneelis tuleb **View** paneelis muuta parameetrit **Background**. Parameetri nime kõrval olevast rippmenüüst tuleb valida **Other**, seejärel avanenud dialoogiaknast teise vahekaardi ja muuta värviskeemi valikud kujule RGB 226, 226, 226 (Joonis 23). Pärast värvide valikut võib dialoogiakna sulgeda paremal üleval asuvast punasest ümmargusest ikoonist.



Joonis 23 Värvi valimine

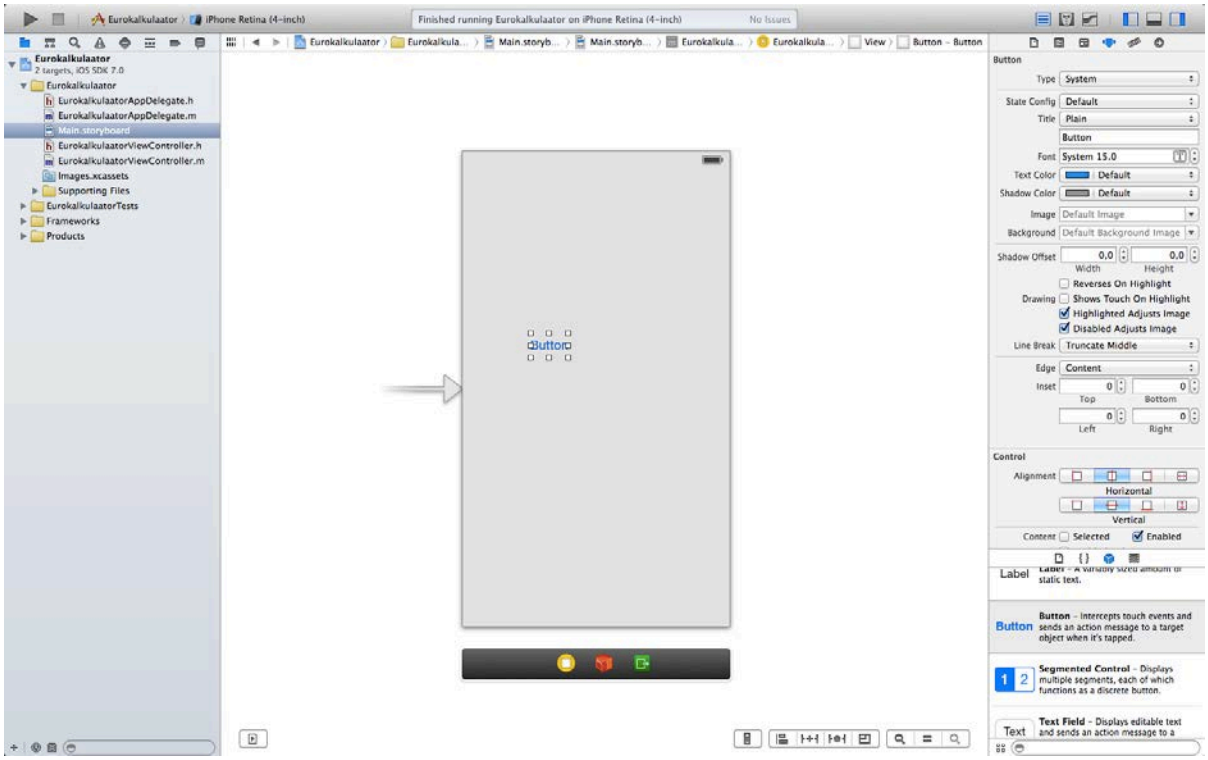
Tulemusena muutub telefoniekraani kujutava lõuendi taustavärv helehalliks (Joonis 24).



Joonis 24 Tausta muutumine

3.4.1 Nuppude loomine

Järgmise sammuna tuleb Eurokalkulaatorile lisada nupud. Töö efektiivseks teostamiseks on mõistlik luua üks korrektne nupp, mida seejärel paljundada kopeerimise ja kleepimisega. Nupu lisamiseks tuleb paremas küljepaneelis asuvast objektide loendist valida **Button** ning lohistada see telefoniekraani kujutavale lõuendile (Joonis 25). Lisatava nupu asukoht ei ole antud hetkel oluline.

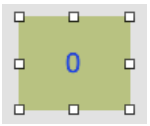


Joonis 25 Nupu lisamine

Märkides nupu aktiivseks, on võimalik muuta selle parameetreid. Parameetrid korraga nähtavale ei mahu. Kerides paremas küljepaneelis võib leida **View** paneelist parameetri **Background**.

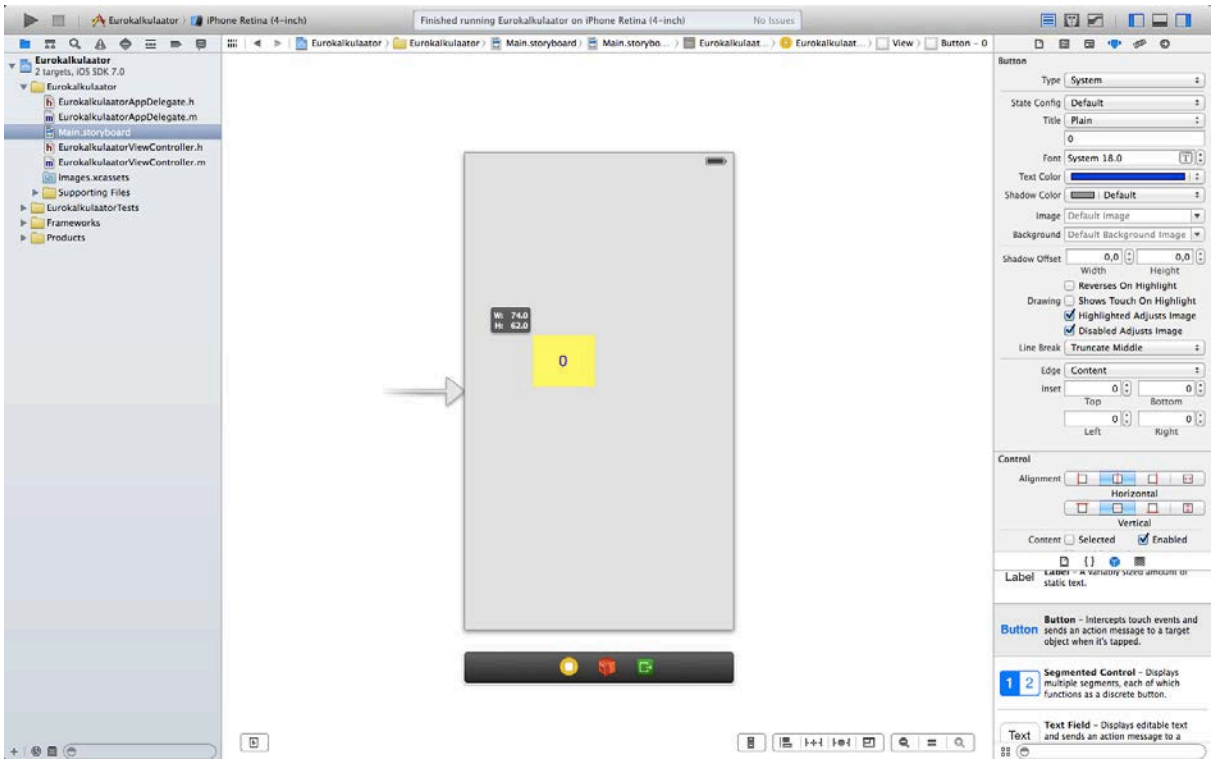
Samalaadselt lõuendi taustavärvi muutmisega tuleb nupu taustavärv muuta helekollaseks, vastavalt RGB 255, 255, 102. Teksti värv tuleb muuta **Button** paneelist parameetri alt nimega **Text Color**. Teksti värv võiks olla tumesinine, vastavalt RGB 0, 0, 255.

Sobiva teksti suuruse **18**, saab määrata **Font** parameetri alt. Nupu pealkirja muutmiseks tuleb sellel teha hiirega topeltklõps, seejärel saab seda muuta. Loodud nupu pealkiri peaks olema arv **0** (Joonis 26).



Joonis 26 Nupu väljanägemine

Sobivaks nupu laiuseks on **74** ja kõrguseks **62**, kuid kindlasti ei tohiks kasutada väiksemaid suurusi kui **44**, sest sellest väiksematele nuppudel võib kasutajal olla raske näpuga pihta saada. Neid parameetreid saab muuta peale nupu valimist (Joonis 26), võttes nupu ühest nurgas kinni ning seda tõmmates. Abistamisteks kuvatakse koheselt nupu kohale selle kõrgus ja laius (Joonis 27).

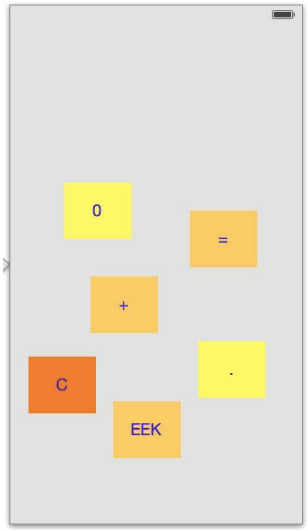


Joonis 27 Nupu suuruse muutmine

Kasutades kopeerimise ja kleepimise meetodit, tuleb luua juurde 5 nuppu. Kopeerimiseks tuleb valida hiireklõpsuga loodud nupu ja kasutades klahvikombinatsiooni, vastavalt kopeerimiseks **cmd⌘ + c** ja kleepimiseks vastavalt **cmd⌘ + v**. Loodud nuppude pealkirjad ja taustavärv peaks vastavalt funktsionaalsusele olema järgmine:

- [.] taustavärv jääb samaks;
- [=], [+], [EEK] taustavärv RGB 255, 204, 102;
- [C] taustavärv RGB 255, 128, 0.

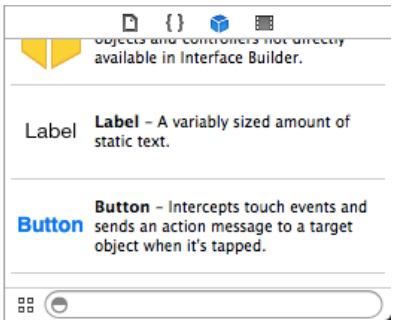
Taustavärvide kasutus on valikuline ja funktsionaalsust ei muuda, kuid oluline on kindlasti muuta nuppude pealkirjad (Joonis 28).



Joonis 28 Erinevat värvi nuppude väljanägemine

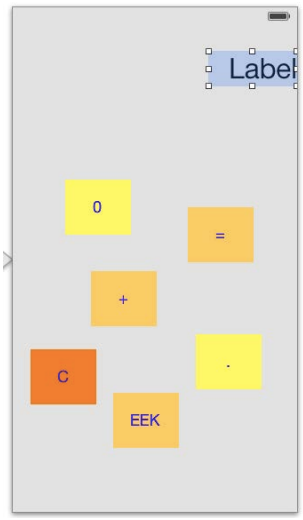
3.4.2 Sildi loomine

Eurokalkulaatorile tuleb luua väli, kuhu hakatakse kuvama sisestatud numbreid ja arvutuste tulemusi. Selleks tuleb lõuendile lisada paremast küljepaneelist, samalaadselt nupu lisamisele, objektide loendist objekt **Label** (Joonis 29).



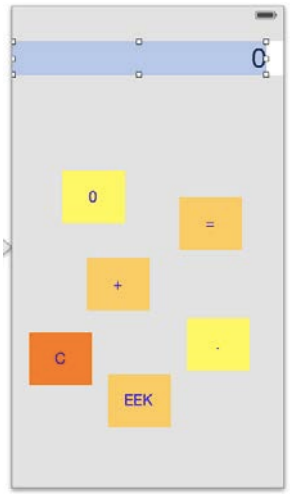
Joonis 29 Silt valmisobjektide loendis

Järgmiseks tuleb muuta loodud sildi omaduste alt **taustavärv valgeks**. Teksti suurus peab olema **32** ja tekst tuleb **joondada paremale**. Sildi laius peab olema **100** ja kõrgus **40**. Loodud silti kasutame abiks kujundamisel (Joonis 30).



Joonis 30 Kitsama sildi lisamine

Tuleb luua ka teine silt, kasutades kopeerimise ja kleepimise meetodit. Tekkinud sildi laiuseks tuleb muuta **300**. Seejärel tuleb kustutada kitsama sildi tekst tühjaks ja paremal sildil muuta kujule **0**. Mõlemad sildid tuleb paigutada ühele kõrgusele selliselt, et kitsam silt oleks parema servaga joondatud vastu paremat äärt ja laiem silt vasaku servaga vastu vasakut äärt, osaliselt kattes kitsamat silti. Sildid võiksid asuda mõõdukal kõrgusel telefoniekraani lõuendi ülemisest äärest (Joonis 31). Kujundamisel kahe sildi sellisel kujul kasutamine ei kuulu parimate praktika alla, kuid sildile veerise tekitamine on palju keerulisem ja võtaks kordades rohkem aega.

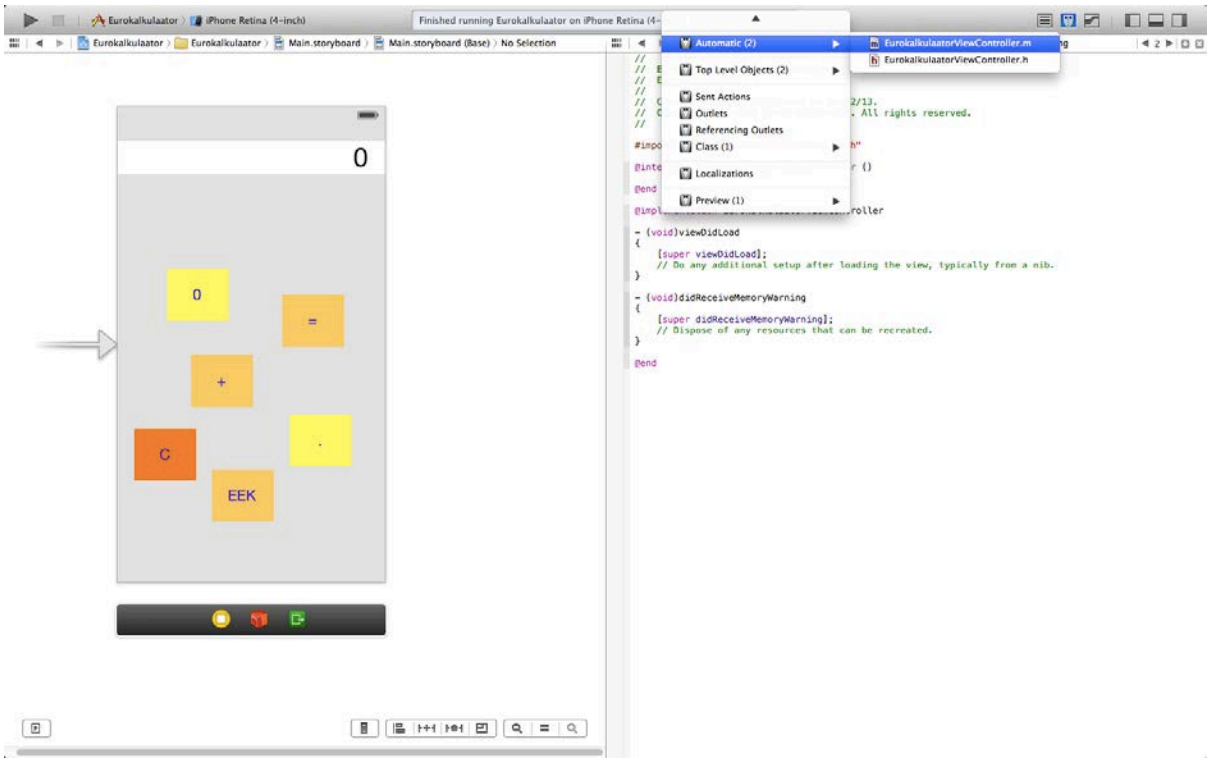


Joonis 31 Mõlemad sildid telefoniekraani lõuendil

3.5 Eurokalkulaatori kasutajaliidese sidumine koodiga

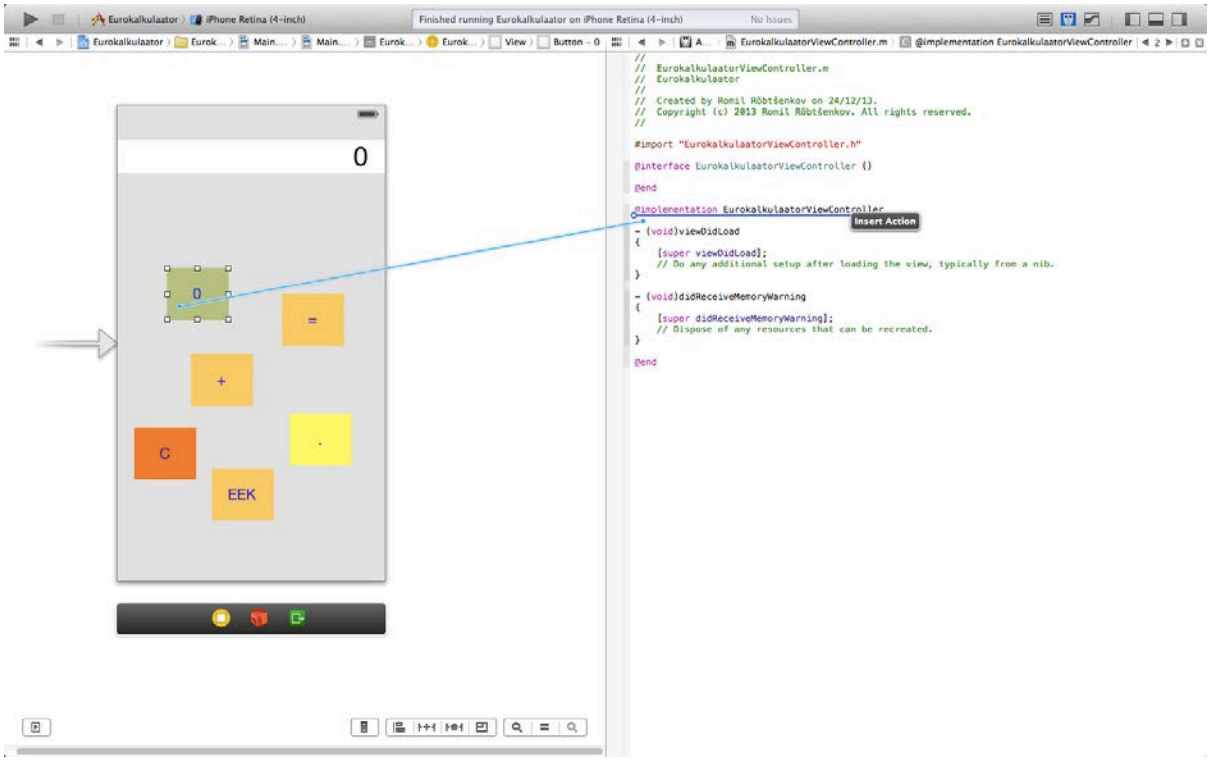
Enne Eurokalkulaatori kasutajaliidese kujundamise lõpule viimist, on vaja siduda nupud koodiga, lisades koodi nuppude vajutamistele vastavad funktsioonid. Seda on vaja teha selleks, et hiljem oleks võimalik paremini aru saada mis nupule kasutaja vajutas.

Parema vaate loomiseks arenduskeskkonnas, on soovitatav peita vasak ja parem küljepaneel (abiks Joonis 14). Järgmiseks tuleb jagada ekraani vertikaalselt kaheks osaks (abiks Joonis 15). Juhul kui paremal pool kuvatavaks failiks ei ole **EurokalkulaatorViewController.m** tuleb paremale ekraanipoolele valida kuvamise valikumenüüst kuvatavaks failiks **Automatic** > **EurokalkulaatorViewController.m** (Joonis 32).



Joonis 32 Paremale ekraanipoolele kuvatava faili valik

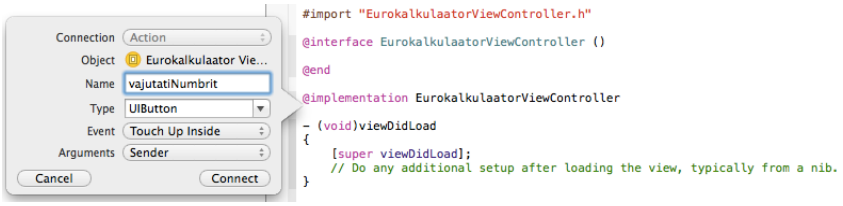
Järgmiseks tuleb siduda nupud koodiga. Alustada tuleb nupust, mis tähistab numbrit **0**. Selleks tuleb **ctrl** klahvi all hoides valida nupp ja hiireklõpsu all hoides lohistada nool koodi, kohe pärast klassi alguse määramist (Joonis 33).



Joonis 33 Nupu sidumine koodiga

Avaneb dialoogiaken, kus tuleb määrata funktsiooni nimeks **vajutatiNumbrit** ja **Type** peab olema **UIButton**. Antud õppematerjal on kõik funktsiooninimed ja muutujad eesti keeles, et oleks võimalik paremini eristada muudetavat sisu Objective-C programmeerimiskeele osadest.

Oma valikute kinnitamiseks tuleb vajutada **Connect** (Joonis 34).



Joonis 34 Lisatava funktsiooni omaduste määramine

Koodiga sidumisel peaks olema koodi lisandunud funktsioon **vajutatiNumbrit** (Koodinäide 1).

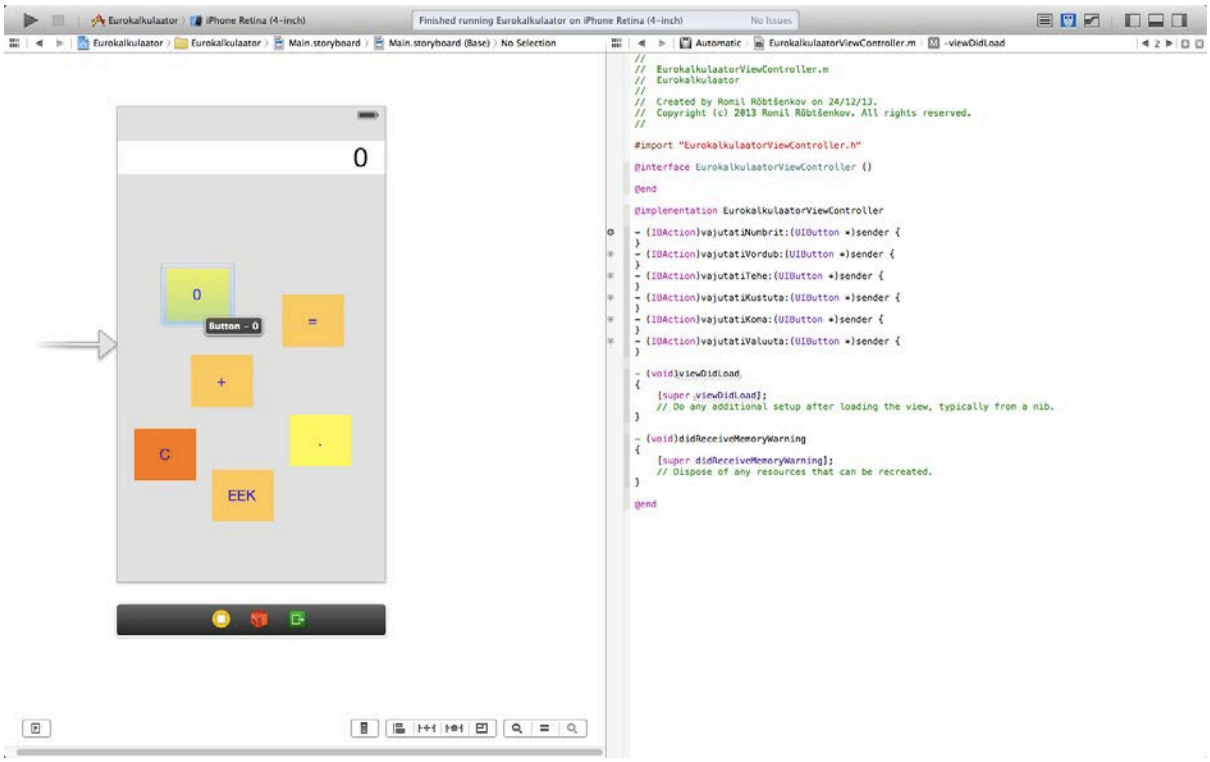
```
- (IBAction)vajutatiNumbrit:(UIButton *)sender {
}
```

Koodinäide 1 Funktsioon vajutatiNumbrit

Funktsioonide lisamist tuleb korrata kõigi teiste nuppudega. Funktsioonidele soovitatavad nimed on nuppude järgi järgmised:

- [=] vajutatiVordub;
- [+] vajutatiTehe;
- [C] vajutatiKustuta;
- [.] vajutatiKoma;
- [EEK] vajutatiValuuta.

Igast lisatud funktsioonist vasakul, on must täpikujuline ikoon, millele hiirega liikudes kuvatakse kõik selle funktsiooniga seotud nupud (Joonis 35).

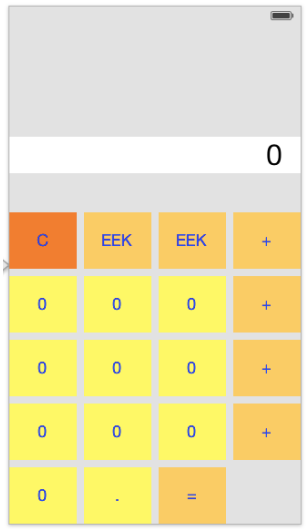


Joonis 35 Funktsiooniga seotud nuppude kuvamine

3.6 Eurokalkulaatori kasutajaliidese lõpetamine

Nüüd on kõik nupud koodiga seotud ja kasutades kopeerimise kleepimise metoodikat, on võimalik Eurokalkulaatori kasutajaliidese kujundamine lõpule viia. Kopeerida on võimalik ka mitu nuppu korraga. Kopeerimisel tuleb kindlasti arvestada, et nuppude funktsionaalsus oleks sama, näiteks kui tegu on tehtemärgiga nupuga, siis kindlasti tuleb kopeerida juba tehtemärgiga nuppu. Seda tuleb teha selleks, et vastavalt nupule käivituks selle vajutamisel õige funktsioon. Enne nuppude ümbernimetamist on soovitatav nupud oma kohtadele

paigutada. Abiks tulevad paigutamise sinised abijooned, mida pakutakse automaatselt, lähtudes teistest nuppudest, servadest, mõistlikust vahemaast nuppude vahel jne. Nuppude paigutamisel võiks lähtuda näidisjoonisest (Joonis 36).

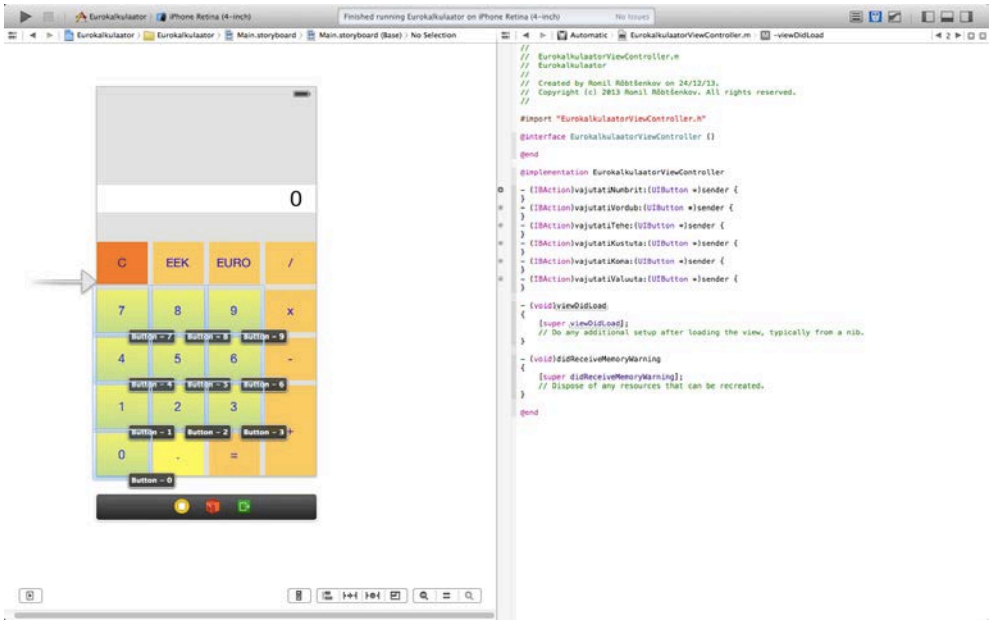


Joonis 36 Eurokalkulaator enne nuppude ümbernimetamist

Pärast nuppude ümbernimetamist on nuppude visuaalne kujundamine lõppenud (Joonis 37).

Joonis 37 Eurokalkulaator pärast nuppude ümbernimetamist

Kui nüüd hiirega liikuda täpikujulisele ikoonile, mis asub enne iga lisatud funktsiooni, siis näiteks funktsiooniga **vajutatiNumbrit** seotud nuppudena näidatakse kõiki nuppe numbritega (Joonis 38). Juhul kui see nii ei ole, siis tuleb loodud uued nupud kustutada ja kopeerimisel ning kleepimisel lähtuda kindlasti nupul olevast pealkirjast.



Joonis 38 Funktsiooniga seotud nuppude kuvamine

Visuaalsele poolele võib veel lisada Eurokalkulaatori logo ja rakenduse looja nime. Seda võib teha kasutades sildi objekte. Selleks on soovitatav lõpetada vaate poolitamine ja sisse lülitada parempoolne küljepaneel. Edasi tuleb objektide loendist lisada sildid ja need vastavalt soovile kujundada. Antud näite puhul on logoks kasutatud **Custom** tekstistiili **Futura Medium** suurusega **37**. Juurde on lisatud vari **Shadow** värviskeemiga RGB 255, 204, 102 ja **Shadow Offset** vertikaalselt **2** ja horisontaalselt **2**. Rakenduse looja nime puhul on kasutatud tekstil helehalli värvi. Rakenduse võib käivitada klahvikombinatsiooniga **cmd⌘ + r** ja vaadata, kuidas näeb kasutajaliides välja simulaatoris (Joonis 39).

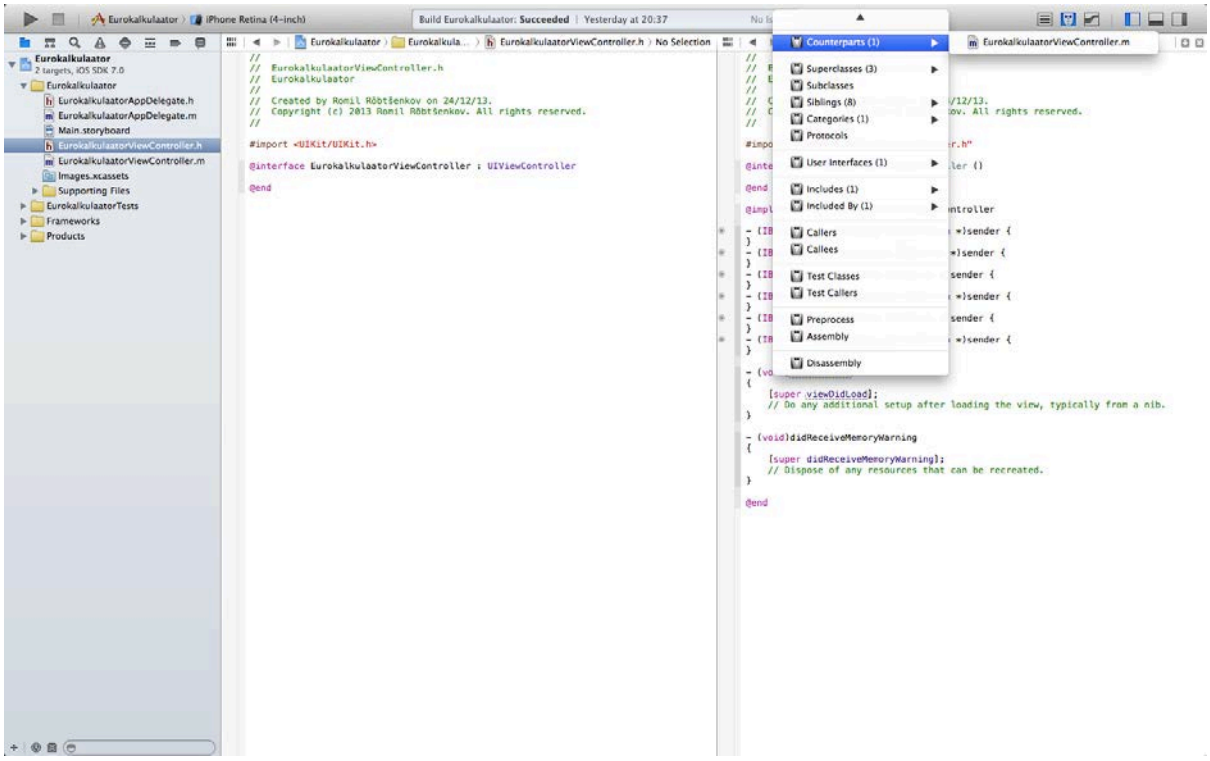
Joonis 39 Eurokalkulaatori kasutajaliidese valmiskuju

3.7 Eurokalkulaatori aju

Enne uue koodi programmeerima asumist on mõistlik tutvuda juba olemasoleva koodiga. Antud õppematerjali eesmärgiks ei ole õpetada Objecitve-C programmeerimiskeelt. Sellest lähtuvalt piirdutakse ainult kasutatava koodi funktsionaalsuse kommenteerimise ja kirjeldamisega ning seda vastavalt tähtsusele antud rakenduse loomisel.

3.7.1 Arenduskeskkonna ettevalmistus programmeerimiseks

Alustuseks tuleb sisse lülitada vasak küljepaneel projektis kasutusel olevate failide nägemiseks. Järgmiseks tuleb failihaldurist ühe hiireklõpsuga avada klassi päisefail **EurokalkulaatorViewController.h**. Vertikaalselt jagatud ekraani paremale poole peaks olema automaatselt kuvatud klassi põhifaili **EurokalkulaatorViewController.m**. Juhul kui nii ei ole, on vaja parema ekraanipoole kuvamise valikumenüüst valida **Counterparts** (Joonis 40).



Joonis 40 Paremale ekraanipoolele kuvatava faili valik

3.7.2 Olemasoleva klassi EurokalkulaatorViewController koodi kirjeldamine

Projekti alustades luuakse koheselt klass **EurokalkulaatorViewController**. Klass koosneb päisefailist **.h** laiendiga ja põhifailist **.m** laiendiga. Päisefail näitab klassist väljapoole kättesaadavat funktsionaalsust, teisisõnu on avalikuks API-ks. Põhifailis on välja toodud klassi kogu funktsionaalsus.

3.7.2.1 Klassi päisefail EurokalkulaatorViewController.h

Klassi päisefailis **EurokalkulaatorViewController.h** on hetkel kolm rida koodi. Viimased kaks rida tähistavad klassi **EurokalkulaatorViewController** algust ja lõppu. Samuti määratakse kooloniga eraldades antud klassi ülemklassi (superklassi) **UIViewController** (Koodinäide 2).

```
@interface EurokalkulaatorViewController : UIViewController

@end
```

Koodinäide 2 Klassi algus ja lõpp

Päise failis olev esimene rida tähistab ülemklassi päisefaili importimist ja sellest tulenevalt ka funktsionaalsuse kaasamist antud klassi (Koodinäide 3).

```
#import <UIKit/UIKit.h>
```

Koodinäide 3 Ülemklassi importimine

3.7.2.2 Klassi põhifail EurokalkulaatorViewController.m

Klassi põhifailis **EurokalkulaatorViewController.m** on koodiridu rohkem. Sarnaselt päisefailile määratakse ka selles failis klassi algus ja lõpp.

Esimest määramist tehakse klassisiseseks kasutamiseks mõeldud privaatsete muutujate joaks (Koodinäide 4).

```
@interface EurokalkulaatorViewController ()

@end
```

Koodinäide 4 Klassi määramine privaatsete muutujate jaoks

Seejärel määratakse klassi algus ja lõpp funktsionaalsuse hoidmiseks (Koodinäide 5).

```
@implementation EurokalkulaatorViewController
```

```
@end
```

Koodinäide 5 Klassi alguse ja lõpu määramine funktsionaalsuse hoidmiseks

Põhifaili imporditakse ka selle sama klassi päisefail (Koodinäide 6).

```
#import "EurokalkulaatorViewController.h"
```

Koodinäide 6 Klassi päisefaili importimine

Antud klassi põhifaili on nüüdseks õppematerjali peatükis Eurokalkulaatori kasutajaliidese sidumine koodiga lisatud funktsioonid (Koodinäide 7). Funktsioonide järjekord ei ole oluline. Lähemalt kirjeldatakse antud funktsioone järgmises peatükis.

```
// Lisatud funktsioonid, mis käivituvad nuppude vajutustele
- (IBAction)vajutatiNumbrit:(UIButton *)sender {
}
- (IBAction)vajutatiVordub:(UIButton *)sender {
}
- (IBAction)vajutatiTehe:(UIButton *)sender {
}
- (IBAction)vajutatiKustuta:(UIButton *)sender {
}
- (IBAction)vajutatiKoma:(UIButton *)sender {
}
- (IBAction)vajutatiValuuta:(UIButton *)sender {
}
```

Koodinäide 7 Nuppude vajutustel käivituvad funktsioonid

Samuti on põhifailis kaks projekti loomisest olemas olnud funktsiooni, millest esimene **viewDidLoad** käivitub koheselt pärast rakenduse vaate laadimist ja teine **didReceiveMemoryWarning** käivitub kui rakendusel tekib mälupuudus (Koodinäide 8).

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a
nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

Koodinäide 8 Funktsioonide viewDidLoad ja didRecieveMemoryWarning kood

3.7.3 Funktsioonide defineerimine

Enne rakenduse programmeerimist on mõistlik tutvuda lihtsama funktsiooniga. Vaadates varasemalt lisatud funktsiooni **vajutatiNumbrit**, on funktsiooni ülesehitus võrdlemisi hästi seletatav.

Koheselt funktsiooni algusesesse kirjutatakse sulgudesse funktsiooni tüüp, mis antud näites on **IBAction**. Samuti võib seal olla funktsiooni poolt tagastatava väärtuse tüüp, näiteks **NSString**, kui funktsioon tagastaks tekstiobjekti. Seejärel kirjutatakse funktsiooni nimi ja pärast semikoolonit valikuliselt saab määrata sisendparameetri, mille tüübiks selles näites on **UIButton** ja seda tähistatakse muutujaga **sender** (Koodinäide 9).

```
- (IBAction)vajutatiNumbrit:(UIButton *)sender {
}
```

Koodinäide 9 Funktsioon vajutatiNumbrit

3.7.4 Muutujad ja logi

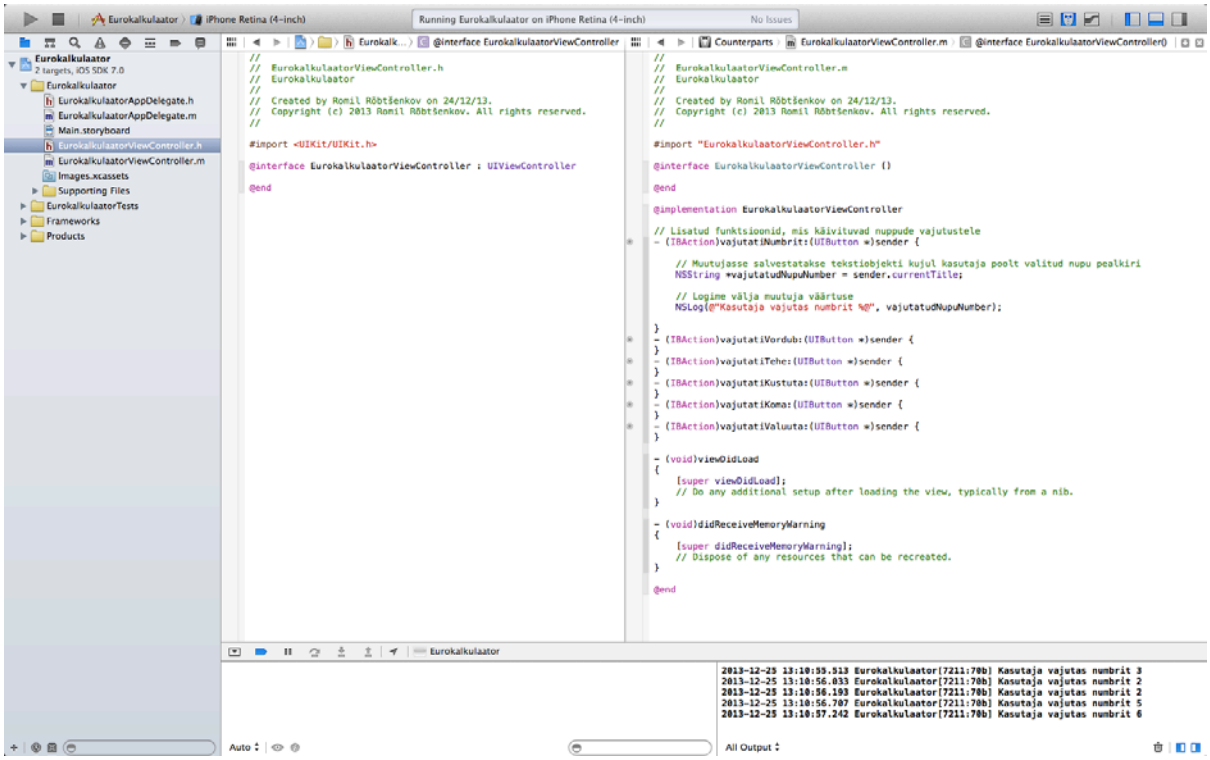
Logi kasutamine on väga oluline ning selleta on programmeerimisel keeruline hakkama saada. Kinnistamaks funktsiooni **vajutatiNumbrit** ülesehitust ja veendumaks, et nupule vajutamisel funktsioon käivitub, on mõistlik logisse välja kirjutada, mis numbriga nupule kasutaja vajutas. Selleks on vaja luua funktsiooni **vajutatiNumbrit** sisse uus muutuja, kuhu tuleb kasutaja poolt vajutatud nupu number lisada teksti kujul ja siis saab selle muutuja väärtuse välja kirjutada logisse. Igale muutujale tuleb määrata tüüp. Muutujate juures kasutatav tärn (*****) tähistab viidet mälupesale. Antud muutuja **vajutatudNupuNumber** tüübiks on **NSString**, mis tähistab, et tegu on tekstiobjketiga. Tekstiobjketi tähistatakse **@** sümboliga, näiteks tühi tekstiobjekt oleks kujul **@””**. Kasutades välja logimiseks funktsiooni **NSLog** asendatakse tekstiobjekti sees olev **%@** muutuja **vajutatudNupuNumber** väärtusega (Koodinäide 10) .

```
- (IBAction)vajutatiNumbrit:(UIButton *)sender {

    NSString *vajutatudNupuNumber = sender.currentTitle;
    // Muutujasse salvestatakse tekstiobjekti kujul kasutaja poolt
valitud nupu pealkiri
```

```
    NSLog(@"Kasutaja vajutas numbrit %@", vajutatudNupuNumber);
    // Logime välja muutuja väärtuse
}
```

Koodinäide 10 Funktsioon vajutatiNumbrit

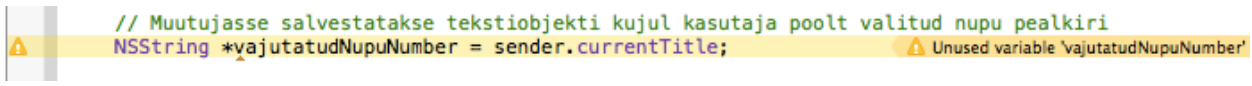
Pärast rakenduse käivitamist simulaatoris, peavad numbritega nuppude vajutamisega tekkima logisse vastavad sissekanded. Logi avaneb automaatselt põhiakna alumisse paneeli (Joonis 41). Alumist paneeli saab avada ja sulgeda sarnaselt küljepaneelidega.



Joonis 41 Logi paneeli kuvamine

3.7.5 Vigade nägemine ja kõrvaldamine

Vigasid koodis näidatakse jooksvalt koodist vasakul kollase või punase hoiatusikooniga., millele vajutades pakutakse üldjuhul välja ka lahendust vea kõrvaldamiseks. Programmeerimise käigus hoiatatakse isegi kasutamata muutujate eest (Joonis 42).

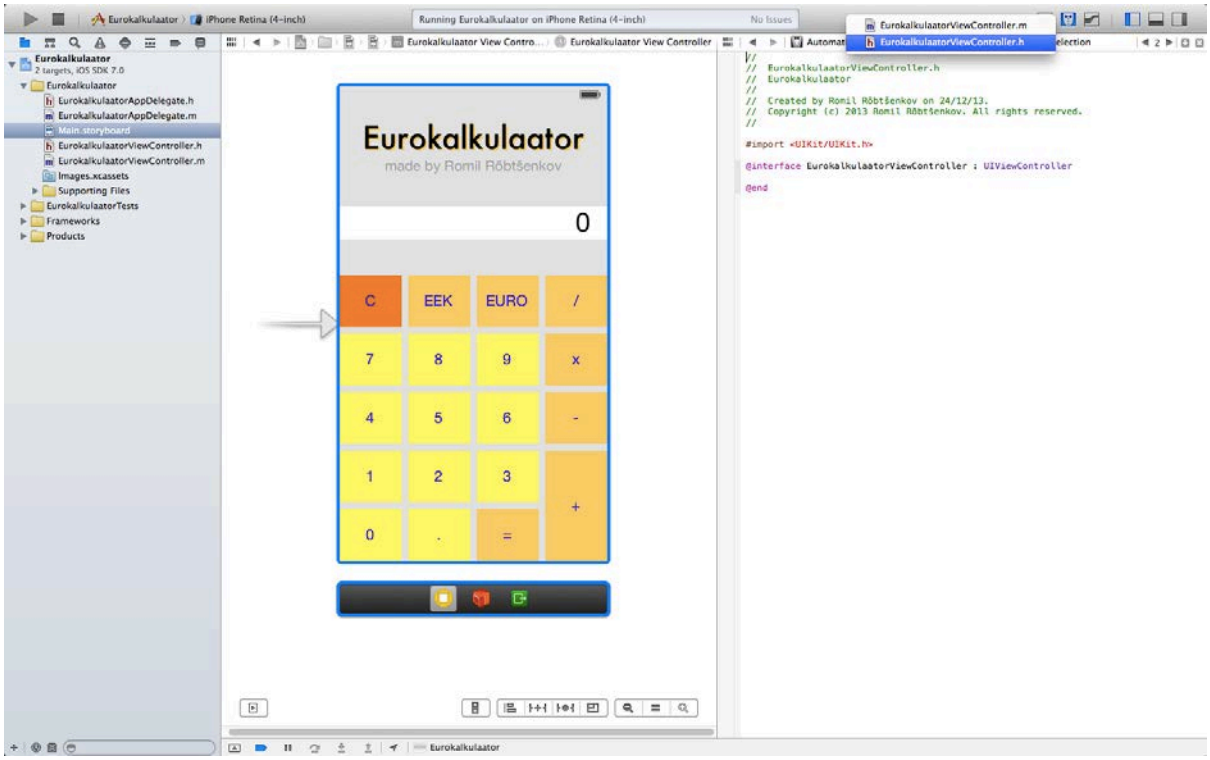


Joonis 42 Hoiatus kasutamata muutuja eest

3.7.6 Numbrite lisamine väljale

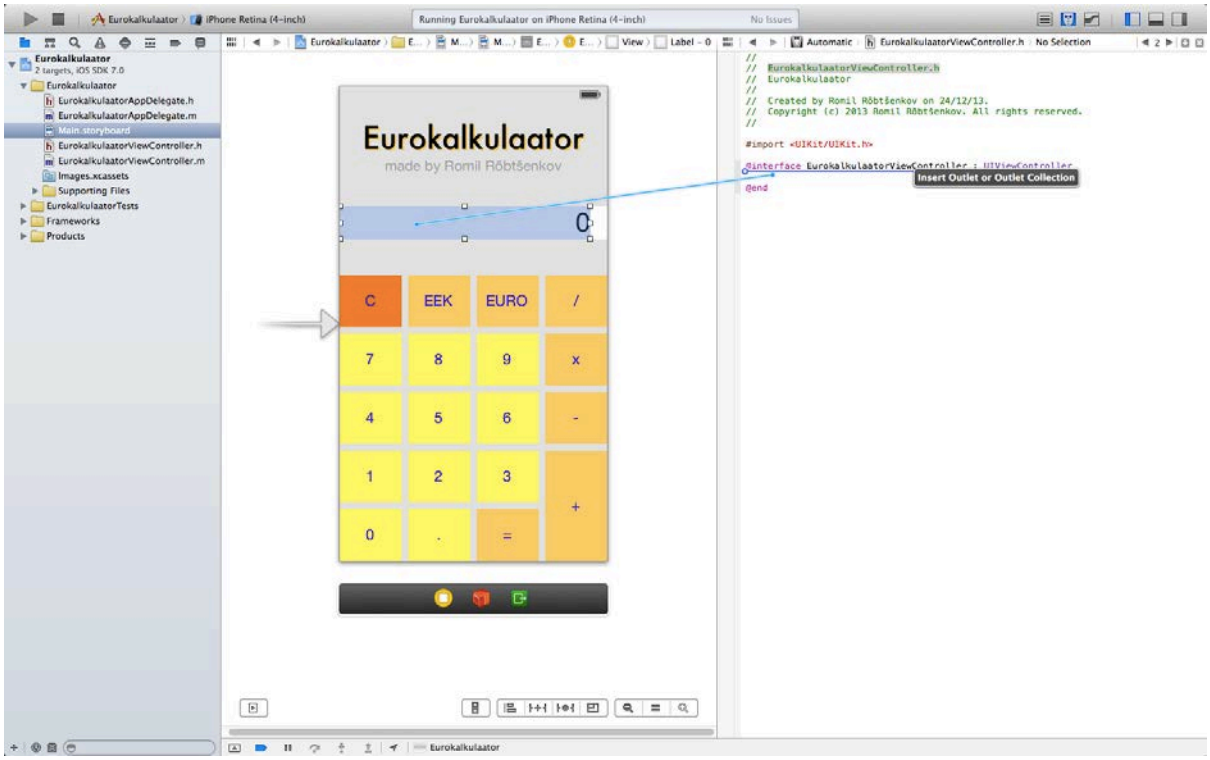
Selleks, et kasutaja poolt vajutatavate nuppude numbrid jõuaksid kalkulaatori väljale, on vaja väli siduda koodiga. Selleks on vaja valida vasakust küljepaneelist fail

Main.storyboard ja paremale poole jaotatud põhiaknasse avada klassi päisefail **EurokalkulaatorViewController.h** (Joonis 43).



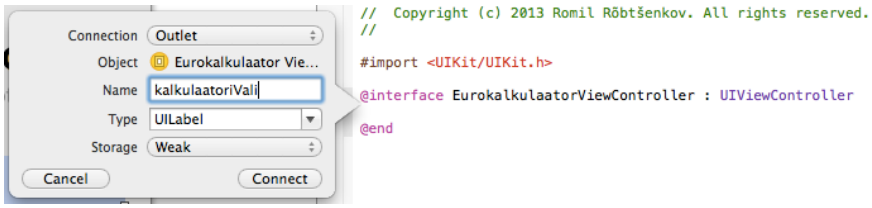
Joonis 43 Paremale ekraanipoolele kuvatava faili valik

Järgmiseks tuleb sarnaselt nuppude sidumisel koodiga, hoides all **ctrl** klahvi, valida laiem silt ja hiireklõpsu all hoides, lohistada nool koodi kohe pärast klassi alguse määramist (Joonis 44).



Joonis 44 Kalkulaatori välja sidumine koodiga

Avaneb dialoogiaken, kus tuleb määrata muutuja nimi millega viidatakse kalkulaatori väljale, olgu selleks antud näites **kalkulaatoriVali** (Joonis 45).



Joonis 45 Muutuja nime valik dialoogiaknas

Selle lisamisel peaks koodi olema tekkinud muutuja definitsioon, mis viitab kalkulaatori väljale (Koodinäide 11). Muutuja defineerimisel määratakse sulgudesse esimese parameetrina kaua muutujat mälus hoitakse. Vastavalt kirjutades **weak** või **strong** määratakse, kas muutujat hoitakse võimalikult kaua mälus või ainult seni, kuni muutujat kasutatakse. Teise parameetrina määratakse sulgudes, kas muutuja poole pöördumisel arvestatakse järjekorda või mitte. Antud rakenduses, kuna ei ole muutuja samaaegset kasutust mitme erineva programmiharu poolt, defineeritakse kõik muutujad **nonatomic** kujul, sellega määrates, et muutuja poole pöördutakse arvestades järjekorda.

```
@property (weak, nonatomic) IBOutlet UILabel *kalkulaatoriVali;
```

Koodinäide 11 Kalkulaatori välja muutuja defineerimine

Selleks, et muutujat saaks kasutada, tuleb see lisada ka klassi põhifaili. Alustuseks on soovitatav failihaldurist avada klassi päisefaili, mille tagajärjel peaks automaatselt jaotatud põhiakna paremale poole olema kuvatud klassi põhifaili. Nüüd saab põhifaili lisada kohe pärast klassi algust ja enne esimest funktsiooni **vajutatiNumbrit,** samanimeline muutuja nagu päisefailis (Koodinäide 12). Kõik edaspidised pöördumised muutujale teostatakse muutuja nime kaudu, lisades ette **self.**.

```
@implementation EurokalkulaatorViewController

// Lisame muutuja
@synthesize kalkulaatoriVali = _kalkulaatoriVali;

// Lisatud funktsioonid, mis käivituvad nuppude vajutustele
- (IBAction)vajutatiNumbrit:(UIButton *)sender {
```

Koodinäide 12 Kalkulaatori välja muutuja lisamine

3.7.7 Muutuja lisamine

Enne seda kui saab hakata kuvama kasutaja poolt sisestatud numbreid väljale, on mõistlik teha uus muutuja, mis hoiaks meeles, kas kasutajal on numbri sisestamine pooleli või mitte. Teisisõnu, kui kasutaja sisestab mitmekohalist numbrit, siis oleks mõistlik teada, et kasutaja alles veel sisestab pikemat numbrit. Samuti kui vajutatakse tehtemärgiga nuppu, siis oleks mõistlik teada, et kasutaja enam numbrit ei sisesta ja hakkab peale tehtemärgi valimist, numbrit sisestama otsast peale. Kuna antud muutujat on vaja kasutada ainult selle klassi piires, siis tuleb defineerida privaatne muutuja, mis oleks **BOOL** tüüpi ehk hoiaks meeles ainult väärtuseid **TRUE** või **FALSE**, kuid olenemata muutuja tüübist, on alati muutuja loomisel selle väärtuseks **nil** (Koodinäide 13).

```
@interface EurokalkulaatorViewController ()

@property (nonatomic) BOOL kasutajlOnNumbriSisestaminePooleli;

@end
```

Koodinäide 13 Muutuja kasutajalOnNumbriSisestaminePooleli defineerimine

Muutuja tuleb lisada ka klassi kohe pärast muutujat kalkulaatoriVali (Koodinäide 14).

```
@synthesize kasutajlOnNumbriSisestaminePooleli =
_kasutajlOnNumbriSisestaminePooleli;
```

Koodinäide 14 Muutuja kasutajalOnNumbriSisestaminePooleli lisamine

Järgmiseks tuleb lisada **vajutatiNumbrit** funktsiooni sisse loogikaavaldis kontrollimaks, kas kasutajal on numbri sisestamine pooleli või mitte. Kui kasutajal on numbri sisestamine pooleli siis tuleb väljale juurde lõppu lisada uus number lisaks funktsiooniga **stringByAppendingString**. Kõik funktsioonid lisatakse nurksulgude vahele. Kui numbri sisestamine pooleli ei ole siis saab kirjutada väljal oleva numbri üle ja määrata, et antud hetkest on kasutajal numbri sisestamine uuesti pooleli (Koodinäide 15).

```
// Kontrollime kas kasutajal on numbri sisestamine pooleli
    if (self.kasutajlOnNumbriSisestaminePooleli) {

        // Lisame väljale lõppu numbri juurde
        self.kalkulaatoriVali.text = [self.kalkulaatoriVali.text
stringByAppendingString:vajutatudNupuNumber];
    } else {

        // Kirjutame välja teksti üle uue numbriga
        self.kalkulaatoriVali.text = vajutatudNupuNumber;

        // Määrame, et nüüdsest on kasutajal numbri sisestamine pooleli
        self.kasutajlOnNumbriSisestaminePooleli = TRUE;
    }
```

Koodinäide 15 Funktsiooni vajutatiNumbrit uue funktsionaalsuse juurde lisamine

Samuti tuleb esialgu funktsioonides **vajutatiVordub** ja **vajutatiTehe** muuta muutuja **kasutajlOnNumbriSisestaminePooleli** väärtuseks **FALSE**, sest vajutades neid nuppe, kasutaja enam numbrit edasi ei sisesta, vaid alustab numbri sisestamist otsast peale (Koodinäide 16).

```
self.kasutajlOnNumbriSisestaminePooleli = FALSE;
```

Koodinäide 16 Muutuja kasutajalOnNumbriSisestaminePooleli väärtuse muutmine

Rakenduse käivitamisel simulaatoris on võimalik proovida sisestada väljale mitmekohalisi numbreid ja vajutades tehtemärgiga nuppu või võrdusmärgiga nuppu, saab numbri sisestamist alustada otsast peale (Joonis 46).

Joonis 46 Eurokalkulaator pärast numbritega nuppude vajutusi

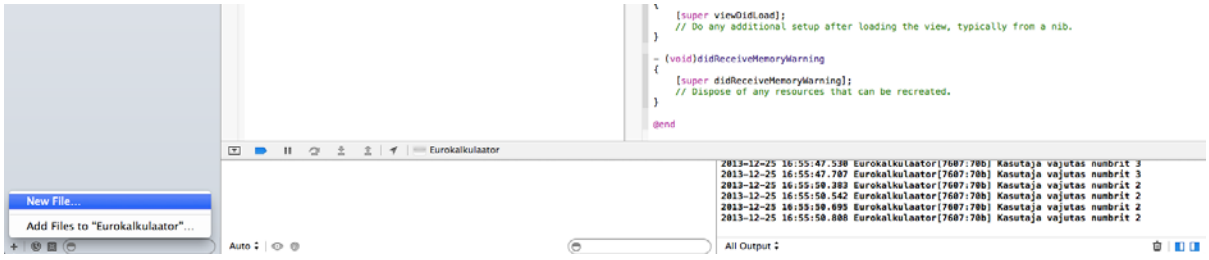
Antud õppematerjali osani valminud rakenduse **EurokalkulaatorViewController** klassi failide lähtekood on õppematerjali lisas **(**Lisa 1**)**. Sealt on võimalik järgi vaadata, kas kõik vajalik siiani on olemas ja õigesti programmeeritud.

3.7.8 Klassid

Klassid aitavad eristada rakenduses erineva funktsionaalsusega osasid. Hetkel on kasutusel klass **EurokalkulaatorViewController**. Rakenduse paremaks ülesehituseks oleks vaja juurde luua uus klass, mis tegeleks arvutustega ja arvude hoiustamisega.

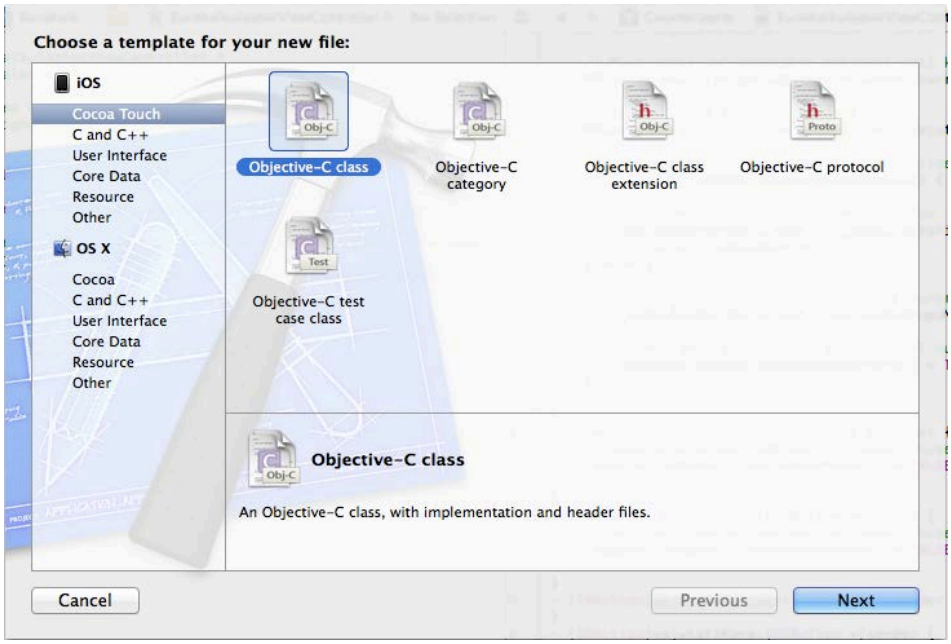
3.7.8.1 Uue klassi lisamine

Uue klassi lisamiseks tuleb lisada uus fail, minnes vasaku küljepaneeli alumisse nurka ja klõpsates pluss märgiga ikoonile ja valides **New File…** (Joonis 47).



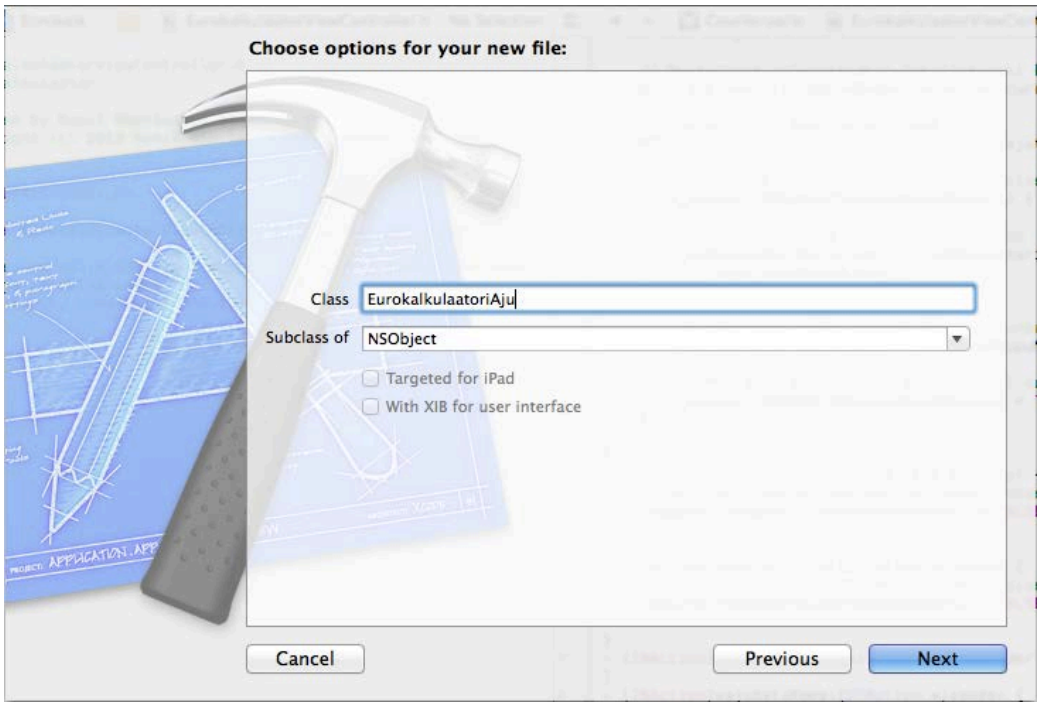
Joonis 47 Uue faili loomine

Järgmiseks avanenud dialoogiaknas tuleb valida **Objective-C class** ja kinnitada valikut vajutades **Next** (Joonis 48).



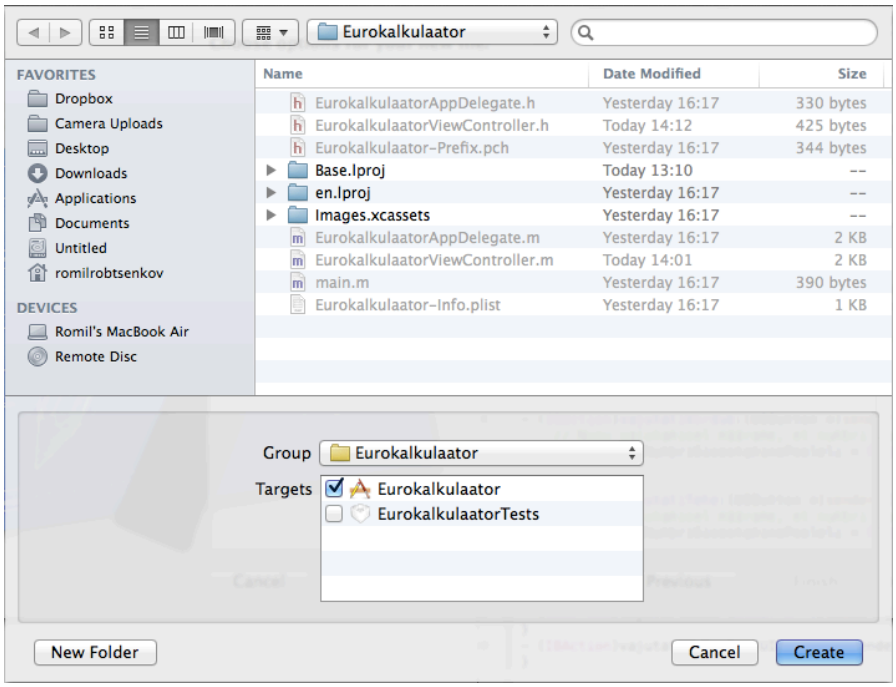
Joonis 48 Faili tüübi valik

Klassile tuleb anda nimi **EurokalkulaatoriAju** ja kinnitada valikut vajutades **Next** (Joonis 49).



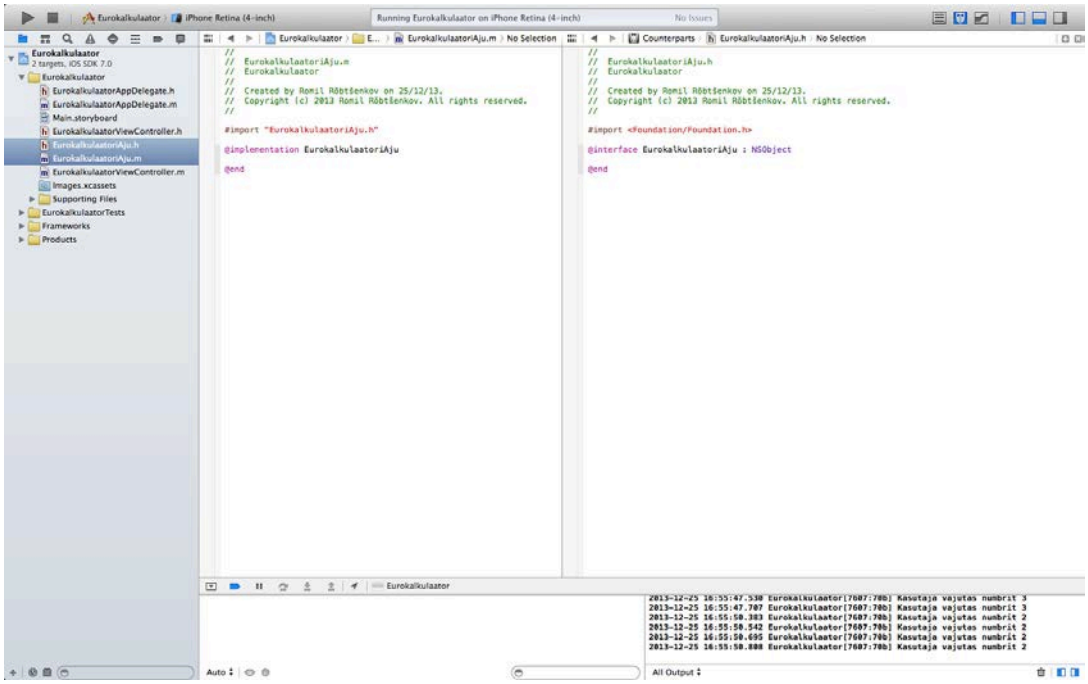
Joonis 49 Klassi nime valik

Juhul kui ei ole soovi faile kuskil eraldi hoiustada, saab jätta valiku samaks ning uus klass luuakse ülejäänud projektiga samasse kausta. Valik tuleb kinnitada vajutades **Create** (Joonis 50).



Joonis 50 Klassi faili salvestamine

Uus klass on loodud (Joonis 51).



Joonis 51 Uus klass arenduskeskkonnas

Koheselt tuleb klassi põhifailis **EurokalkulaatoriAju.m** määrata ainult klassipiires kasutatava funktsionaalsuse hoiustamise koht (Koodinäide 17).

```
#import "EurokalkulaatoriAju.h"

// Klassis kasutavate privaatsete muutujate hoiustamise koht
@interface EurokalkulaatoriAju()

@end

@implementation EurokalkulaatoriAju

@end
```

Koodinäide 17 EurokalkulaatoriAju klassipiires kasutatava funktsionaalsuse hoiustamise koha määramine

3.7.8.2 Klasside sidumine

Selleks, et saaks kasutada loodud klassi funktsionaalsust juba ennem olemas olnud klassis EurokalkulaatorViewController, on vaja klassid omavahel siduda. Selleks tuleb faili **EurokalkulaatorViewController.m** kohe pärast klassi päise importimist importida uue loodud klassi **EurokalkulaatoriAju** päis **EurokalkulaatoriAju.h** (Koodinäide 18).

```
#import "EurokalkulaatorViewController.h"
#import "EurokalkulaatoriAju.h"
```

Koodinäide 18 EurokalkulaatoriAju klassi päisefaili importimine

Järgmiseks tuleb samas failis **EurokalkulaatorViewController.m** defineerida privaatne muutuja **aju**, mille tüübiks on **EurokalkulaatoriAju** (Koodinäide 19).

```
// Defineerime privaatse muutuja
@property (nonatomic) BOOL kasutajlOnNumbriSisestaminePooleli;
@property (strong, nonatomic) EurokalkulaatoriAju *aju;
```

Koodinäide 19 Privaatse muutuja defineerimine

Seejärel tuleb teiste muutujate lisamise järele lisada muutuja **aju** ja funktsioon, mis initsialiseerib klassi (Koodinäide 20).

```
@synthesize aju = _aju;

// Initsialiseerime klassi
- (EurokalkulaatoriAju *)aju
{
    if(!_aju){
         _aju = [[EurokalkulaatoriAju alloc] init];
    }
    return _aju;
}
```

Koodinäide 20 Muutuja lisamine ja EurokalkulaatoriAju klassi initsialiseerimine

3.7.9 Pinu

Järgmiseks on vaja koostada pinu, kuhu peab saama lisada numbreid ja samuti neid sealt võtta . Muutuja **Pinu** tüübiks **NSMutableArray** ning see muutuja suudab hallata mistahes objekte. Pinu tuleb defineerida privaatse muutujana failis **EurokalkulaatoriAju.m** (Koodinäide 21).

```
@property (nonatomic, strong) NSMutableArray *pinu;
```

Koodinäide 21 Pinu muutuja defineerimine

Järgmiseks tuleb samasse faili lisada muutuja ja initsialiseerime pinu (Koodinäide 22).

```
@synthesize pinu = _pinu;

// Initsialiseerime pinu
- (NSMutableArray *)pinu
{
    if (_pinu == nil) {
        _pinu = [[NSMutableArray alloc] init];
    }

    return _pinu;
}
```

Koodinäide 22 Pinu muutuja lisamine ja käivitamine

3.7.9.1 Pinusse lisamine

Selleks, et pinusse saaks lisada objekte, tuleb luua uus funktsioon faili **EurokalkulaatoriAju.m**, mis sinna parameetrina saadetud objekte vastu võtab ja seejärel lisab pinusse (Koodinäide 23).

```
- (void)lisaPinusse:(double)arv
{
    [self.pinu addObject:[NSNumber numberWithDouble:arv]];
}
```

Koodinäide 23 Funktsioon lisaPinusse

Selleks, et pinusse saaks lisada objekte ka väljaspool antud klassi, tuleb funktsioon lisada ka antud klassi päisefaili **EurokalkulaatoriAju.h** (Koodinäide 24).

```
@interface KalkulaatoriAju : NSObject

- (void)lisaPinusse:(double)arv;

@end
```

Koodinäide 24 Funktsiooni avalikuks tegemine

3.7.9.2 Pinust võtmine

Selleks, et pinust saaks võtta elemente välja, tuleb luua eraldi funktsioon faili **EurokalkulaatoriAju.m**, mis tagastab viimase objekti ja kui see ei ole tühi objekt, siis kustutab selle objekti pinust (Koodinäide 25).

```
-(double)votaPinust
{
    NSNumber *arvPinust = [self.pinu lastObject];

    if (arvPinust) {
        [self.pinu removeLastObject];
    }

    return [arvPinust doubleValue];
}
```

Koodinäide 25 Funktsioon votaPinust

3.7.10 Funktsionaalsus

Kalkulaatori funktsionaalsuse tekitamiseks tuleb alguses hakata lisama kasutaja poolt sisestatud numbreid pinusse, kuid seda on vaja teha alles siis, kui kasutajal pole enam numbri sisestamine pooleli. Toimides nii saab sisestada mitmekohalise numbri pinusse korraga, mitte eraldi ühe numbri haaval. Selleks tuleb lisada pinusse numbri kalkulaatori

väljalt ainult juhul, kui kasutaja vajutab võrdub või ühte tehtemärgiga nuppudest. Enne seda tuleb kindlasti kontrollida, et kasutajal oli numbri sisestamine pooleli. Seda on vaja selleks, et tühjasid objekte pinusse lisama ei peaks. Enne numbri lisamist tuleb muuta number teksti kujult **double** tüübiks, et hiljem saaks pinust võetud arvuga koheselt arvutusi teha. Antud funktsionaalsus tuleb lisada funktsioonidele **vajutatiVordub** ja **vajutatiTehe** (Koodinäide 26).

```
- (IBAction)vajutatiVordub:(UIButton *)sender {

    if (self.kasutajlOnNumbriSisestaminePooleli){

        // Lisame pinusse numbri kalkulaatori väljalt
        [self.aju lisaPinusse:[ self.kalkulaatoriVali.text
doubleValue]];

        self.kasutajlOnNumbriSisestaminePooleli = FALSE;
    }

}

- (IBAction)vajutatiTehe:(UIButton *)sender {

    if (self.kasutajlOnNumbriSisestaminePooleli){
        [self.aju lisaPinusse:[ self.kalkulaatoriVali.text
doubleValue]];
        self.kasutajlOnNumbriSisestaminePooleli = FALSE;
    }
}
```

Koodinäide 26 Funktsiooides vajutatiVordub ja vajutatiTehe uue funktsionaalsuse lisamine

3.7.10.1 Tehtemärgiga ja võrdusmärgiga nupu vajutus

Selleks, et kasutaja poolt sisestatud arve saaks kalkulaatoril liita, on vaja luua uus funktsioon **EurokalkulaatoriAju** klassi. Funktsiooni **teeArvutus** eesmärgiks on võtta vastu parameetrina tehtemärki ja seejärel võttes pinust kaks viimast arvu, teostada nende arvudega tehe vastavalt tehtemärgile ning tagastada tulemus. Samuti on oluline saadud tulemus lisada pinusse, et sellega saaks uusi arvutusi teha (Koodinäide 27).

```
// arvutuste tegemise funktsioon, mis tagastab arvutuse tulemuse
- (double)teeArvutus:(NSString *)tehtemark
{
    double tulemus = 0;

    // Kontrollib, mis tehtemärgiga on tegu
    if ([tehtemark isEqualToString:@"+"]) {

        tulemus = [self votaPinust] + [self votaPinust];
    }

    // Lisab tulemuse pinusse
    [self lisaPinusse:tulemus];

    // Tagastab Tulemuse
    return tulemus;
}
```

Koodinäide 27 Funktsioon teeArvutus

Selleks, et antud funktsiooni oleks võimalik kasutada väljaspool antud klassi, on vaja see lisada ka klassi päise faili **EurokalkulaatoriAju.h** sarnaselt funktsiooniga **lisaPinusse** (Koodinäide 28).

```
- (void)lisaPinusse:(double)arv;
- (double)teeArvutus:(NSString *)tehtemark;
```

Koodinäide 28 Funktsiooni teeArvutus avalikuks muutmine

Järgmiseks on vaja käivitada loodud funktsioon failis **EurokalkulaatorViewController.m**, kui kasutaja vastavale nupule vajutab. Tähtis on meeles pidada tehtemärki, millega peab hakkama teostama arvutust. Selleks on vaja defineerida failis **EurokalkulaatorViewController.m** uus muutuja **teheMisOnMeeles**, mis hoiab tekstiobjektina meeles tehtemärki, mis nuppu kasutaja vajutab enne teise numbri sisestamist (Koodinäide 29).

```
@property (strong, nonatomic) NSString* teheMisOnMeeles;
```

Koodinäide 29 Uue muutuja teheMisOnMeeles defineerimine

Pärast klassi defineerivat osa, tuleb muutuja **teheMisOnMeeles** luua (Koodinäide 30).

```
@synthesize teheMisOnMeeles = _teheMisOnMeeles;
```

Koodinäide 30 Muutuja teheMisOnMeeles loomine

Funktsiooni **vajutatiTehe** tuleb juurde lisada funktsionaalsus, mis kontrollib, kas tehtemärk oli enne nupu vajutust juba meelest. Juhul kui oli meeles, siis teostatakse arvutus, enne seda luues ajutise muutuja **tulemus**. Saadud tulemus muudetakse teksti

kujule ja kuvatakse kalkulaatori väljal. Samuti jäetakse meelde uus tehtemärk. Juhul kui tehtemärki meeles ei olnud, jäetakse nüüd nupuvajutusest saadud tehtemärk meelde ning mingisuguseid arvutusi ei teostata (Koodinäide 31).

```
// Kontrollime, et meeles olev tehe ei ole tühi või nil
if ([@"" isEqualToString: self.teheMisOnMeeles] || self.teheMisOnMeeles
== nil) {

    // Tehe jäetakse meelde
    self.teheMisOnMeeles = sender.currentTitle;

    NSLog(@"tehe '%@' jäeti meelde", self.teheMisOnMeeles);

} else {

    // Tuleb teha arvutus eelnevalt meelde jäetud tehtega ja jätta uus
tehe meelde
    double tulemus = [self.aju teeArvutus: self.teheMisOnMeeles];

    NSLog(@"Tehti arvutus tehtega %@", self.teheMisOnMeeles);

    // Teisendab tulemuse teksti vormingusse ja kuvame kalkulaatori
väljal
    self.kalkulaatoriVali.text = [NSString stringWithFormat:@"%g",
tulemus];

    // Uus tehe jäetakse meelde
    self.teheMisOnMeeles = sender.currentTitle;

}
```

Koodinäide 31 Funktsiooni vajutatiTehe uue funktsionaalsuse lisamine

Funktsiooni **vajutatiVordub** tuleb lisada sarnane funktsionaalsus, kuid seekord uut tehet meelde ei jäeta, kuna kasutaja vajutas võrdusmärgiga nuppu, mitte nuppu, kus oli peal tehtemärk (Koodinäide 32).

```
// Kontrollime, kas tehe mis on meeles ei ole tühi tekstiobjekt
if (![@"" isEqualToString: self.teheMisOnMeeles]) {

    double tulemus = [self.aju teeArvutus: self.teheMisOnMeeles];

    self.kalkulaatoriVali.text = [NSString stringWithFormat:@"%g",
tulemus];

    NSLog(@"Tehti arvutus tehtega %@", self.teheMisOnMeeles);

    // Muudame meeleshoitava tehte tühjaks tekstiobjketiks
    self.teheMisOnMeeles = @"";
}
```

Koodinäide 32 Funktsiooni vajutatiVordub uue funktsionaalsuse lisamine

Rakenduse käivitamisel simulaatoris saab teostada arvutusi kasutades liitmistehet ja võrdusmärgiga nupp täidab oma funktsionaalsust.

3.7.10.2 Kustutamine kalkulaatori väljalt

Järgmisena tuleb võimaldada kasutajal kustutada pinu tühjaks ja alustada kalkulaatori kasutamisega otsast peale ilma, et kasutaja peaks rakenduse sulgema. Faili **EurokalkulaatoriAju.m** tuleb lisada funktsioon **tuhjendaPinu**, mis antud ülesannet teostab (Koodinäide 33).

```
- (void)tuhjendaPinu
{
    [self.pinu removeAllObjects];
}
```

Koodinäide 33 Funktsioon tuhjendaPinu

Samuti tuleb võimaldada funktsioonile **tuhjendaPinu** ligipääs antud klassiga seotud klassist, lisades funktsiooni klassi **EurokalkulaatoriAju** päisefaili (Koodinäide 34).

```
- (void)tuhjendaPinu;
```

Koodinäide 34 Funktsiooni tuhjendaPinu avalikuks muutmine

Kui käivatakse klassi **EurokalkulaatorViewController** funktsioon **vajutatiKustuta**, siis tuleb kustutada pinu tühjaks, muuta kalkulaatori välja tekst kujule 0**,** kustutada võimalik meeles hoitav tehtemärk ja samuti määrata, et kasutajal pole enam numbri sisestamine pooleli (Koodinäide 35).

```
 - (IBAction)vajutatiKustuta:(UIButton *)sender {

    // Tühjendame pinu
    [self.aju tuhjendaPinu];

    self.kasutajlOnNumbriSisestaminePooleli = FALSE;
    self.teheMisOnMeeles = @"";
    self.kalkulaatoriVali.text = @"0";
}
```

Koodinäide 35 Funktsiooni vajutatiKustuta uue funktsionaalsuse lisamine

Rakenduse käivitamisel simulaatoris on võimalik veenduda kustutamise nupu töötamises.

Eurokalkulaatori osaline funktsionaalsus on antud õppematerjaliga tagatud (Joonis 52).

Joonis 52 Eurokalkulaator valmiskujul

Võimalike koodivigade või funktsionaalsuse probleemide tekkimisel on võimalik nende kõrvaldamiseks vaadata valminud rakenduse klasside lähtekoodi õppematerjali lisast **(**Lisa 2**)**.

3.8 Iseseisvad ülesanded

Antud õppematerjaliga õpitu kinnistamiseks on soovitatav lahendada järgmised ülesanded:

- lisada lahutamise, korrutamise ja jagamise funktsionaalsus;
- EURO või EEK pealkirjaga nupule vajutades vastavalt jagatakse või korrutatakse kalkulaatoril oleva väljal asuv tulemus Eesti Panga euro kursiga 15,6466;
- lisada komakohaga arvude sisestamise võimalus;
- piirata mitme kohalist arvu on võimalik sisestada.

Ülesannete lahendamisel on abiks otsingumootorid ja loominguline lähenemine.

3.9 Valminud rakenduse lähtekood

Valmis rakenduse klasside lähtekood on õppematerjali lisas **(**Lisa 2**)**.

Valmis rakenduse projektifail koos kõigi teiste failidega on allalaetav aadressilt http://romil.ee/seminaritoo/Eurokalkulaator.zip (60KB). Rakenduse avamiseks on vaja allalaetud faili lahtipakkimise teel saadud kaustast käivitada fail **Eurokalkulaator.xcodeproj**.

Kokkuvõte

Käesolev õppematerjalis sai antud ülevaade iOS operatsioonisüsteemist ja Xcode arenduskeskkonnast ning seejärel õpetatud sammhaaval rakenduse loomist iOS operatsioonisüsteemi jaoks.

Eurokalkulaatori nimelise arendatud rakenduse osaline funktsionaalsus sai õppematerjali läbimise käigus tagatud. Iseseisvate ülesannete täitmisega oli võimalik rakendus funktsionaalsus lõpuni arendada.

iOS operatsioonisüsteemi jaoks rakenduse loomise õppimise jätkamiseks soovitab õppematerjali autor läbida tasuta Stanford Ülikooli inglisekeelse kursuse nimega **Developing iOS 7 Apps for iPhone and iPad**. Kursuse materjal on kättesaadav õpikeskkonnast **iTunesU** ja aadressil https://itunes.apple.com/us/course/developing-ios-7-apps-for/id733644550.

Õppematerjali autor loodab, et õppematerjal oli arusaadav ja selle abil õppimine huvitav!

Kasutatud kirjandus

Apple Inc. (2013, juuni 10). *Apple Unveils iOS 7*. Retrieved jaanuar 1, 2014, from Apple: http://www.apple.com/pr/library/2013/06/10Apple-Unveils-iOS-7.html

Apple Inc. (2013). *Choosing an iOS Developer Program*. Retrieved jaanuar 1, 2014, from Apple: https://developer.apple.com/programs/start/ios/

Apple Inc. (2013, oktoober 22). *iOS Human Interface Guidelines*. Retrieved jaanuar 1, 2014, from Apple: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf

Apple Inc. (2013, oktoober 22). *iOS Simulator Guide*. Retrieved jaanuar 1, 2014, from Apple: https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/iOS_Simulator_Guide.pdf

Apple Inc. (2013, september 18). *iOS Technology Overview*. Retrieved detsember 27, 2013, from iOS Developer Library: https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/iOSTechOverview.pdf

Apple Inc. (2013, oktoober 22). *Start Developing iOS Apps Today*. Retrieved jaanuar 1, 2014, from Apple: https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/RoadMapiOS.pdf

Hughes, N. (2013, detsember 31). *iOS 7 now installed on 78% of active Apple handheld devices*. Retrieved jaanuar 1, 2014, from Apple Insider: http://appleinsider.com/articles/13/12/31/ios-7-now-installed-on-78-of-active-apple-handheld-devices

IDC. (2013, november 12). *Android Pushes Past 80% Market Share While Windows Phone Shipments Leap 156.0% Year Over Year in the Third Quarter, According to IDC*. Retrieved jaanuar 1, 2014, from IDC: http://www.idc.com/getdoc.jsp?containerId=prUS24442013

Lisad

Lisa 1 Poolelioleva rakenduse klasside kood

Lisa 2 Valmis rakenduse klasside kood

Lisa 1 Poolelioleva rakenduse klasside kood

EurokalkulaatorViewController.h

```
#import <UIKit/UIKit.h>

@interface EurokalkulaatorViewController : UIViewController

// Defineerime välja tähistava muutuja. Kindlasti peab olema seos
muutuja ja välja vahel
@property (weak, nonatomic) IBOutlet UILabel *kalkulaatoriVali;

@end
```

EurokalkulaatorViewController.m

```
#import "EurokalkulaatorViewController.h"

@interface EurokalkulaatorViewController ()

// Defineerime privaatse muutuja
@property (nonatomic) BOOL kasutajlOnNumbriSisestaminePooleli;

@end

@implementation EurokalkulaatorViewController

// Lisame muutujad
@synthesize kalkulaatoriVali = _kalkulaatoriVali;
@synthesize kasutajlOnNumbriSisestaminePooleli =
_kasutajlOnNumbriSisestaminePooleli;

// Lisatud funktsioonid, mis käivituvad nuppude vajutustele
- (IBAction)vajutatiNumbrit:(UIButton *)sender {

    // Muutujasse salvestatakse tekstiobjekti kujul kasutaja poolt
valitud nupu pealkiri
    NSString *vajutatudNupuNumber = sender.currentTitle;

    // Logime välja muutuja väärtuse
    NSLog(@"Kasutaja vajutas numbrit %@", vajutatudNupuNumber);

    // Kontrollime kas kasutajal on numbri sisestamine pooleli
    if (self.kasutajlOnNumbriSisestaminePooleli) {

        // Lisame väljale lõppu numbri juurde
        self.kalkulaatoriVali.text = [self.kalkulaatoriVali.text
stringByAppendingString:vajutatudNupuNumber];
    } else {

        // Kirjutame välja teksti üle uue numbriga
        self.kalkulaatoriVali.text = vajutatudNupuNumber;

        // Määrame, et nüüdsest on kasutajal numbri sisestamine pooleli
        self.kasutajlOnNumbriSisestaminePooleli = TRUE;
    }
```

```
}
- (IBAction)vajutatiVordub:(UIButton *)sender {
    // Nupu vajutamisel määrame, et numbri sisestamine pole enam pooleli
    self.kasutajlOnNumbriSisestaminePooleli = FALSE;

}
- (IBAction)vajutatiTehe:(UIButton *)sender {
    // Nupu vajutamisel määrame, et numbri sisestamine pole enam pooleli
    self.kasutajlOnNumbriSisestaminePooleli = FALSE;

}
- (IBAction)vajutatiKustuta:(UIButton *)sender {
}
- (IBAction)vajutatiKoma:(UIButton *)sender {
}
- (IBAction)vajutatiValuuta:(UIButton *)sender {
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a
nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

Lisa 2 Valmis rakenduse klasside kood

EurokalkulaatorViewController.h

```
#import <UIKit/UIKit.h>

@interface EurokalkulaatorViewController : UIViewController

// Defineerime välja tähistava muutuja. Kindlasti peab olema seos
muutuja ja välja vahel
@property (weak, nonatomic) IBOutlet UILabel *kalkulaatoriVali;

@end
```

EurokalkulaatorViewController.m

```
#import "EurokalkulaatorViewController.h"
#import "EurokalkulaatoriAju.h"

@interface EurokalkulaatorViewController ()

// Defineerime privaatse muutuja
@property (nonatomic) BOOL kasutajlOnNumbriSisestaminePooleli;
@property (strong, nonatomic) EurokalkulaatoriAju *aju;
```

```
@property (strong, nonatomic) NSString* teheMisOnMeeles;

@end

@implementation EurokalkulaatorViewController

// Lisame muutujad
@synthesize kalkulaatoriVali = _kalkulaatoriVali;
@synthesize kasutajlOnNumbriSisestaminePooleli =
_kasutajlOnNumbriSisestaminePooleli;
@synthesize aju = _aju;
@synthesize teheMisOnMeeles = _teheMisOnMeeles;

// Käivitame klassi
- (EurokalkulaatoriAju *)aju
{
    if(!_aju) _aju = [[EurokalkulaatoriAju alloc] init];

    return _aju;
}

// Lisatud funktsioonid, mis käivituvad nuppude vajutustele
- (IBAction)vajutatiNumbrit:(UIButton *)sender {

    // Muutujasse salvestatakse tekstiobjekti kujul kasutaja poolt
valitud nupu pealkiri
    NSString *vajutatudNupuNumber = sender.currentTitle;

    // Logime välja muutuja väärtuse
    NSLog(@"Kasutaja vajutas numbrit %@", vajutatudNupuNumber);

    // Kontrollime kas kasutajal on numbri sisestamine pooleli
    if (self.kasutajlOnNumbriSisestaminePooleli) {

        // Lisame väljale lõppu numbri juurde
        self.kalkulaatoriVali.text = [self.kalkulaatoriVali.text
stringByAppendingString:vajutatudNupuNumber];
    } else {

        // Kirjutame välja teksti üle uue numbriga
        self.kalkulaatoriVali.text = vajutatudNupuNumber;

        // Määrame, et nüüdsest on kasutajal numbri sisestamine pooleli
        self.kasutajlOnNumbriSisestaminePooleli = TRUE;
    }

}
- (IBAction)vajutatiVordub:(UIButton *)sender {

    if (self.kasutajlOnNumbriSisestaminePooleli){

        // Lisame pinusse numbri kalkulaatori väljalt
        [self.aju lisaPinusse:[self.kalkulaatoriVali.text doubleValue]];

        self.kasutajlOnNumbriSisestaminePooleli = FALSE;
    }

    // Kontrollime, kas tehe mis on meeles ei ole tühi tekstiobjekt
    if (![@"" isEqualToString:self.teheMisOnMeeles]) {
```

```
        double tulemus = [self.aju teeArvutus:self.teheMisOnMeeles];

        self.kalkulaatoriVali.text = [NSString stringWithFormat:@"%g",
tulemus];

        NSLog(@"Tehti arvutus tehtega %@", self.teheMisOnMeeles);

        // Muudame meeleshoitava tehte tühjaks tekstiobjketiks
        self.teheMisOnMeeles = @"";
    }
}

- (IBAction)vajutatiTehe:(UIButton *)sender {

    if (self.kasutajlOnNumbriSisestaminePooleli){

        // Lisame pinusse numbri kalkulaatori väljalt
        [self.aju lisaPinusse:[self.kalkulaatoriVali.text doubleValue]];

        self.kasutajlOnNumbriSisestaminePooleli = FALSE;
    }

    // Kontrollime, et meeles olev tehe ei ole tühi või nil
    if ([@"" isEqualToString:self.teheMisOnMeeles] ||
self.teheMisOnMeeles == nil) {

        // Tehe jäetakse meelde
        self.teheMisOnMeeles = sender.currentTitle;
        NSLog(@"Tehe '%@' jäeti meelde", self.teheMisOnMeeles);

    } else {

        // tuleb teha arvutus eelnevalt meelde jäetud tehtega ja jätta
uus tehe meelde
        double tulemus = [self.aju teeArvutus:self.teheMisOnMeeles];

        NSLog(@"Tehti arvutus tehtega %@", self.teheMisOnMeeles);

        // Teisendab tulemuse teksti vormingusse ja kuvame kalkulaatori
väljal
        self.kalkulaatoriVali.text = [NSString stringWithFormat:@"%g",
tulemus];

        // Uus tehe jäetakse meelde
        self.teheMisOnMeeles = sender.currentTitle;

    }
}

- (IBAction)vajutatiKustuta:(UIButton *)sender {

    // Tühjendame pinu
    [self.aju tuhjendaPinu];

    self.kasutajlOnNumbriSisestaminePooleli = FALSE;
    self.teheMisOnMeeles = @"";
    self.kalkulaatoriVali.text = @"0";
}
```

```
- (IBAction)vajutatiKoma:(UIButton *)sender {
}
- (IBAction)vajutatiValuuta:(UIButton *)sender {
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a
nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

EurokalkulaatoriAju.h

```
#import <Foundation/Foundation.h>

@interface EurokalkulaatoriAju : NSObject

- (void)lisaPinusse:(double)arv;
- (double)teeArvutus:(NSString *)tehtemark;
- (void)tuhjendaPinu;

@end
```

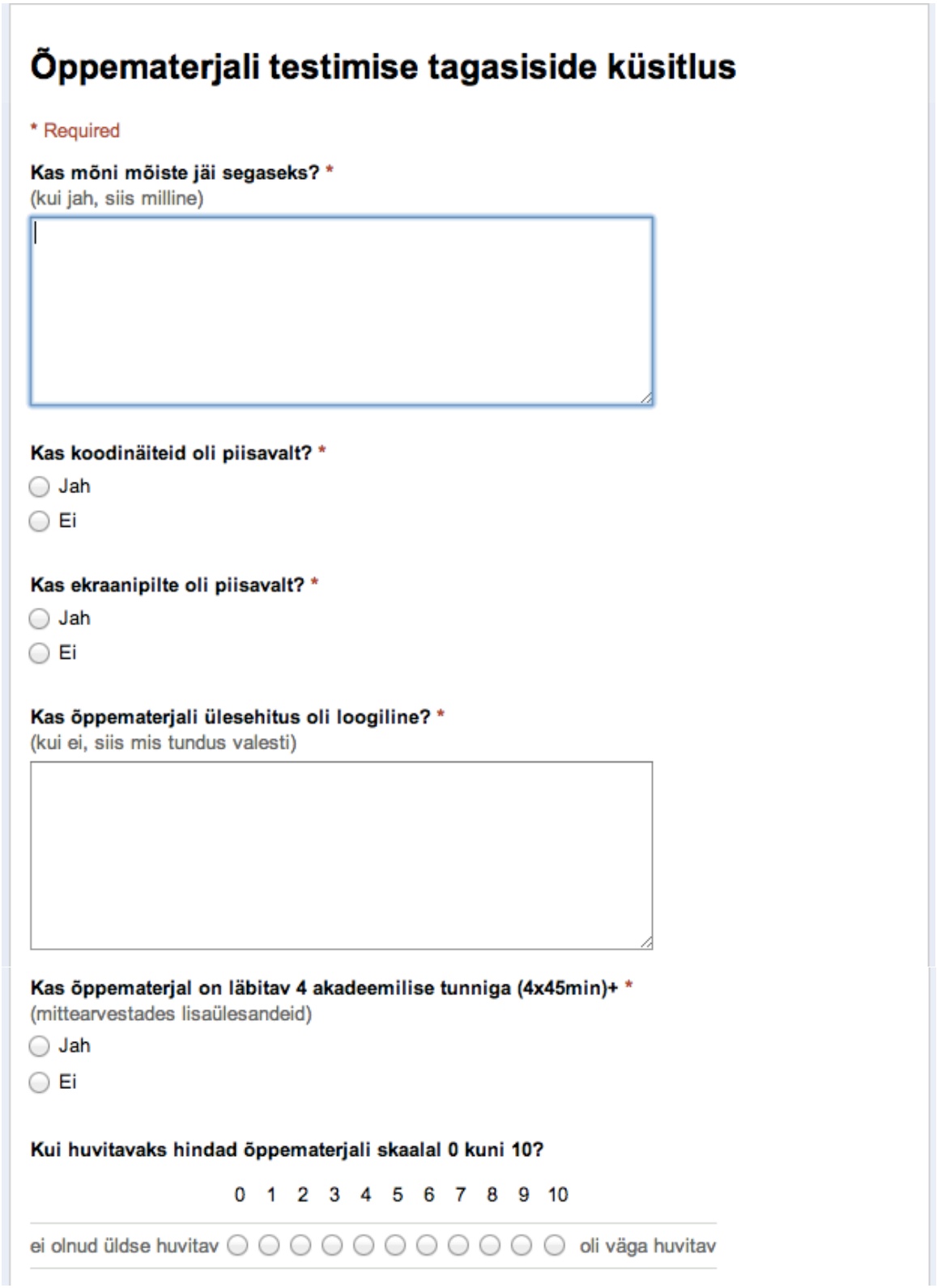
EurokalkulaatoriAju.m

```
#import "EurokalkulaatoriAju.h"

// Klassis kasutavate privaatsete muutujate hoiustamise koht
@interface EurokalkulaatoriAju()

// Defineerime pinu
@property (nonatomic, strong) NSMutableArray *pinu;

@end

@implementation EurokalkulaatoriAju

// Lisame muutuja
@synthesize pinu = _pinu;

// Käivitame pinu
- (NSMutableArray *)pinu
{
    if (_pinu == nil) {
        _pinu = [[NSMutableArray alloc] init];
    }

    return _pinu;
```

```
}

// Pinusse lisamise funktsioon
- (void)lisaPinusse:(double)arv
{
    [self.pinu addObject:[NSNumber numberWithDouble:arv]];
}

// Pinust võtmise funktsioon
-(double)votaPinust
{
    NSNumber *arvPinust = [self.pinu lastObject];
    if (arvPinust) {
        [self.pinu removeLastObject];
    }

    return [arvPinust doubleValue];
}

// arvutuste tegemise funktsioon, mis tagastab arvutuse tulemuse
- (double)teeArvutus:(NSString *)tehtemark
{
    double tulemus = 0;


    // Kontrollib, mis tehtemärgiga on tegu
    if ([tehtemark isEqualToString:@"+"]) {

        tulemus = [self votaPinust] + [self votaPinust];
    }

    // Lisab tulemuse pinusse
    [self lisaPinusse:tulemus];

    // Tagastab Tulemuse
    return tulemus;
}

- (void)tuhjendaPinu
{
    [self.pinu removeAllObjects];
}

@end
```

Lisa 2 Testimise tagasiside küsitluse ankeet

Õppematerjali testimiselt tagasiside saamiseks koostati küsitlus, mis asub aadressil: https://docs.google.com/forms/d/13-2D7vfOucpFWSrXrjdO0j3vgCSWAxmLfX6-ZjLIkFw/viewform.



Õppematerjali testimise tagasiside küsitlus

* Required

Kas mõni mõiste jäi segaseks? *
(kui jah, siis milline)

Kas koodinäiteid oli piisavalt? *
- Jah
- Ei

Kas ekraanipilte oli piisavalt? *
- Jah
- Ei

Kas õppematerjali ülesehitus oli loogiline? *
(kui ei, siis mis tundus valesti)

Kas õppematerjal on läbitav 4 akadeemilise tunniga (4x45min)+ *
(mittearvestades lisaülesandeid)
- Jah
- Ei

Kui huvitavaks hindad õppematerjali skaalal 0 kuni 10?

	0	1	2	3	4	5	6	7	8	9	10	
ei olnud üldse huvitav	○	○	○	○	○	○	○	○	○	○	○	oli väga huvitav

Kas soovitaksid õppematerjali teistele?
(kui ei, siis miks)

Kommentaarid (valikuline)
(mis meeldis, mis ei meeldinud, mis oleks võinud olla teisiti jne)

Submit

Never submit passwords through Google Forms.

Lisa 3 Testimise tagasiside küsitluse tulemused

Testija	Kas õppematerjal on läbitav 4 akadeemilise tunniga (4x45min)+	Kas mõni mõiste jäi segaseks?	Kas koodinäiteid oli piisavalt?	Kas ekraanipilte oli piisavalt?	Kas õppematerjali ülesehitus oli loogiline?	Kui huvitavaks hindad õppematerjali skaalal 0 kuni 10?	Kas soovitaksid õppematerjali teistele?	Kommentaarid (valikuline)
Testija 1	Jah	Nii palju kui ma lugesin/sirvisin, siis ei.	Jah	Jah	Oli küll loogiline. Ehitus progresseerus vastavalt ülesande kulgemisele. Ajasid näpuga järge ja valmis oligi.	9	Jah kindlasti. Kuna õppematerjalis on kokkupuude objektorienteeritud C'ga ning lisaks tutvustatakse väga mugavat ning kasutaja sõbralikku programmeerimis keskkonda. Seega ülevaade/õpetus mitmest valdkonnast.	Kollasel nupul vale värvi kood taustad valgeks siltidel osa on puudu. pikema labeli teksti joondamine puudu (et kuhu täpsemalt) Miks muutuja nimed eesti keelsed? pole esimene funktsioon (numbrid) - oleneb, mis pidi drag & drop teha. Kas kõik funktsiooni alla või järjestikuliselt. LK 36, 43, 44 (koodinäide31 allkiri) - vajutatiTehet. Mujal vajutatiTehe LK 37 Klassile antava nimi tekstis vale. Peaks olema "EurokalkulaatoriAju" LK 40 Eurokalkulaatori aju osa, on veidi segane. Et kus mis täpselt läheb.
Testija 2	Jah	Ei	Jah	Jah	Jah	8	Jah, õppematerjal andis algajale iOS arendajale hea ülevaate arenduskeskkonnast ja lihtsamate funktsioonide loomisest.	Ilmnesid erinevad pisivead, mis aga ei seganud õppematerjali läbimist ja sellest arusaamist. Natukene häiris see, et mõned ekraanipildid olid liiga väikesed ja nendest aru saamiseks pidi pingutama. Pisivigadest teavitasin õppematerjali loojat.
Testija 3	Jah	Oli kasutatud mõistet nagu funktsiooni juurde, aga sisse oleks olnud paremini arusaadav millegipärast	Jah	Jah	Jah	8	Kui kellelgil on vaja õppida IOS-ile arendamist, siis on väga hea materjal alustamiseks	255, 102, 102 on lõheroosa, mitte kollane, vist pidi olema 255, 255, 102 Kirjavead: tekstiobjketi pärat kasutajlOnNumbriSisestaminePooleli KalkulaatoriAju -> EuroKalkulaatoriAju 3.7.8.2 Klasside sidumine peatükis läks veidike aega, enne kui sain aru, mis faili tuleb mis kopeerida Koodi kopeerimisel tulid leheküljenumbrid kaasa