Portering av Android applikation till iOS

– En analys av verktyget J2ObjC

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för datavetenskap

Examensarbete:
Alex Nhan
Adam Nilsson

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2014

Sammanfattning

Mobila applikationer har blivit allt viktigare i dagens teknologiska samhälle. För att kunna möta kundernas behov måste dessa applikationer genomgå en ständig förändring och förbättring. Många företag måste därför formar om deras applikationer så att de kan stödja andra plattformar på ett effektivt sätt. Syftet och målet med detta examensarbete var att hitta ett effektivt sätt att portera en Android applikation till en iOS applikation. Detta examensarbete fokuserade på ett verktyg utvecklat av Google med namnet J2ObjC som kan översätta Java kod till Objective-C kod. J2ObjC kan enbart översätta kod ej relaterad till Android API. Verktyget användes praktisk för att undersöka om den var den mest lämpliga metoden för att portera applikationer med.

Slutsatsen av detta examensarbete var att J2ObjC inte är ett verktyg som kan översätta en hel applikation direkt. Verktyget är tänkt att vara ett stöd till den traditionella plattformsspecifika utvecklingen, dvs. ett verktyg som samarbetar med utvecklarna i översättningsprocessen. Resultatet från undersökningen visade en tidsförbättring när man översatte en kod med J2ObjC i jämförelse med när man översatte koden för hand. Resultatet visade att införandet av J2ObjC i utvecklingsprocessen kommer att förbättra den tid det tar för att översätta koder. Dock krävs ytterligare undersökningar angående prestandan av den översatta koden.

Keywords: Android, iOS, J2ObjC, Java, Objective-C, Android API

Abstract

Mobile applications have become really important in today's technological society. To be able to meet the customers' demands, these applications must undergo a constant alteration and improvement. Many companies must therefore make their mobile applications support different platforms effectively. The purpose and goal of this thesis is to find a way to create a native iOS application from an Android application effectively. This thesis focuses on a tool named J2ObjC which can translate Java code to Objective-C code. J2ObjC is a tool developed by Google which can translate code not related to the Android API. This tool was put into practical use to investigate if it is the most suitable method for application porting.

The conclusion of this thesis is that J2ObjC is not a tool which can translate a whole application directly. J2ObjC is an aid to traditional platform specific development, or in other words a tool which collaborates with the developers in the translation process. The result from the investigation shows a time improvement while translating a code with J2ObjC in comparison with translating the code by hand. These results prove that adopting J2ObjC in the development will improve the time it takes to translate codes. However, further researches regarding the performance of the translated code are necessary.

Keywords: Android, iOS, J2ObjC, Java, Objective-C, Android API

Förord

Detta är ett examensarbete som gjordes under vårterminen 2014 på Saab AB, Training and Simulation i Helsingborg. Denna examensrapport är resultatet av det arbete som gjordes på deras kontor.

Arbetet har varit mycket lärorik samt givit oss en mycket bra insyn i hur det kommande arbetslivet kommer se ut.

Först och främst vill vi tacka alla på Saab i Helsingborg för ett trevligt bemötande och goda råd under arbetets gång. Vi vill även tacka för att vi fick äran att göra vårt examensarbete på Saab.
Ett speciellt tack till vår handledare Andreas Carlsson som har givit oss ett gott stöd under hela arbetets gång.

Sist vill vi tacka vår examinator Christin Lindholm som har givit oss goda råd angående rapportskrivning samt alla frågor som vi har haft under examensarbetets gång.

Adam Nilsson och Alex Nhan

Innehållsförteckning

Kapitel 1

Inledning

Mobila applikationer är väldigt viktiga i dagens samhälle eftersom mycket av dagens teknik bygger på sådana applikationer. Mobiltelefoner, surfplattor, skrivare och datorer är några exempel på tekniker som innehåller mobila applikationer. För att kunna möta kundernas behov måste dessa applikationer genomgår en ständig förändring och förbättring.

Med tanke på kundernas behov har många mobila applikationer blivit porterade till andra plattformar. Ett exempel är iOS applikationen Instagram som används till fotodelning. Denna applikation finns nu även tillgängligt till Android.

Men enligt HyperlinkInfoSystems [11] definition av portering måste en applikation förändras för att kunna stödja en annan plattform. En anledning till varför denna förändring är nödvändigt är pga. att plattformarna använder sig av olika programmeringsspråk. Ett exempel är Objective-C som används för att utveckla applikationer till Apples mobila plattformar, medan Java används för att utveckla applikationer till Android [19]. Eftersom programmeringsspråken skiljer sig så måste koden för varje funktion i en applikation skrivas om för att den skall kunna fungera på en annan plattform. Då funktionerna kommer att ha samma funktionalitet när applikationen körs i en annan plattform, är det väldig ineffektivt om funktionerna måste skrivas om hela tiden till ett annat programmeringsspråk. En omskrivning kan undvikas om det finns ett verktyg som direkt kan göra en översättning av koden för en funktionalitet.

1.1 Bakgrund

Examensarbetet gjordes på Saab AB, Training and Simulation i Helsingborg våren 2014. Saab förser den globala marknaden, såsom regeringen, myndigheten och företag, med produkter, tjänster och lösningar som sträcker sig från militärt försvar till civil säkerhet. Saab AB, Training and Simulation har utvecklat en observationsapplikation för analys och beslutsfattande i realtid. Applikationen stödjer förnärvarande bara två plattformar, Android och Windows. Saab vill med hjälp av detta examensarbete hittar det effektivaste sättet att portera Android versionen av applikationen så att den även fungerar i en iOS miljö. Målet med examensarbetet gick därför ut på att undersöka hur man portera Android applikationer till iOS mer effektivt.

Som nämndes tidigare så har många mobila applikationer blivit porterade till andra plattformar. Med detta i åtanken har Google utvecklat ett verktyg med namnet J2ObjC som översätter Java kod till Objective-C kod. Objective-C är programmeringsspråket som används i iOS plattformar. Fokus på examensarbetet ligger därför på J2ObjC, eftersom konceptet bakom det liknar det önskade målet med examensarbetet.

1.2 Teknisk bakgrund för Saabs system

För att förstå hur Saabs observationsapplikation fungerar behöver man ha kännedom av Saabs system som applikationen använder (se Fig. 1.1.). Applikationen används till att skicka och ta emot information från Saabs system. Saabs system används till att sammanställa den stora mängden data som kommer från applikationerna till förståelig information. Informationen kan sedan skickas ut till alla som är berättigad att se den. De som är berättigad att se informationen är administratören över systemet och medlemmarna i samma grupp som personen som skickade datan.

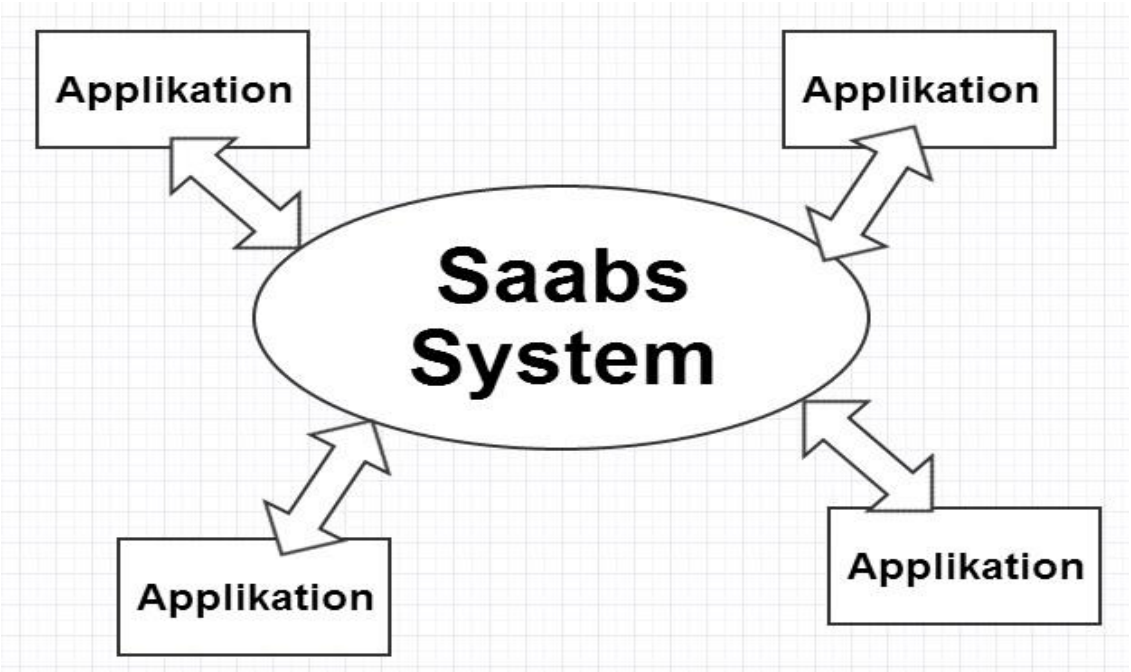


Fig. 1.1. *Saabs System*

Systemet kan användas till både militärt- och civilt bruk beroende på hur det är implementerat. Exempel på användningsområden kan vara vid naturkatastrofer för att rapportera hur stor skadan är, eller till Röda korset för hur olika resurser ska fördelas i en humanitär katastrof. Vid en jordbävning kan exempelvis data angående hur drabbat ett visst område är skickas in med hjälp av applikationen. Dessa data sammanställs i systemet och skickas sedan ut till alla som är berättigad att se informationen. På så sätt kan man ta beslut om hur man ska fördela resurserna så att så många som möjligt får den hjälp de behöver [21].

1.3 Problemformulering

Målet med detta examensarbete gick ut på att analysera vilken som är den effektivaste metoden att portera en Android baserad applikation till iOS. En prototyp togs sedan fram med hjälp av analysen för att bevisa att metoden uppfyller målet med examensarbetet. Kravet på prototypen var att den skall vara skriven i naturligt Objective-C kod.
Android versionen av observationsapplikationen användes till porteringen. Applikationen, som är utvecklat för militärt- och civilt bruk, används till analys och beslutsfattande i realtid.

Med målet av examensarbetet i åtanken besvarades följande frågor i denna examensrapport:

- Vilken mjukvara krävs för att använda metoden?
- Vilken hårdvara krävs för att använda metoden?
- Vilka programmeringsspråk krävs för att kunna göra en portering med metoden?
- Kommer metoden att göra porteringen mer effektivt och mindre tidskrävande?
- Kommer metoden vara det bästa sättet att portera från Android till iOS?

1.4 Avgränsningar

Examensarbetet behandlade endast portering mellan Android applikationer, som är skrivna i Java, och iOS applikationer som är skrivna i Objective-C. Fokus på examensarbetet låg på analysen av hur man skall kunna portera en existerande Android applikation till iOS på ett så effektivt sätt som möjligt.
Konceptet bakom multi-plattform utveckling går ut på att samma kod återanvänds till flera olika mobila plattformar. En kort introduktion till vad multi-plattform utveckling är och varför det inte var lämpligt till detta examensarbete förklaras i kapitel 3.

Kapitel 2

Metod

Metoderna som användes för informationsanskaffningen var av kvalitativ karaktär. Dessa metoder bestod av gruppdiskussioner [17] och intervjuer [14] som var semi-strukturerad. Under implementationen av prototypen användes scrumban [9] för att få en struktur i utvecklingsprocessen. Andra metoder som också användes för att styra arbetet mot målet var tidsplan, förstudier, validering av krav [15] och verifiering av verktyg [15]. Metodvalen som användes i examensarbetet kommer att presenteras och motiveras i detta kapitel.

2.1 Tidsplan

För att få ett strukturerat arbetssätt gjordes en preliminär tidsplan i början av examensarbetet, där alla nödvändiga arbetsmoment som skulle styra examensarbetet mot målet finns med. Tidsplanen uppdaterades i samband med informationsinhämtningen. Den behövde också uppdateras under implementationen av prototypen. För orsaken till denna uppdatering, se avsnitt 2.4.4.
Eftersom slutrapporten inte blev färdigt kunde inte examensarbetet avslutas i tid. Detta medförde att ännu en justering i tidsplanen behövdes. Orsaken till varför slutrapporten inte blev färdigt i tid var pga. att implementationen av prototypen tog längre tid än beräknat. För den slutgiltiga tidsplanen se appendix B.1.

2.2 Intressenter

Det fanns ett antal utvecklare på Saab som har varit med och utvecklat Android applikationen som examensarbetarna porterade. Handledaren för detta examensarbete var en av utvecklarna som har varit med och utvecklat applikationen. Alla intervjuer, diskussioner och valideringar gjordes med handledaren, eftersom handledaren hade hand om detta examensarbete och alltid var tillgängligt.

2.3 Intervju

Intervju med handledaren gjordes i början för att verifiera vad målet med examensarbetet var. Problemformuleringen och tidsplanen som gjordes i början av examensarbetet var baserad på denna intervju. Fler intervjuer gjordes även i samband med informationsinhämtningen till examensarbetet. Den tekniska bakgrunden för Android applikationen, kraven på prototypen och orsaken till varför Saab vill göra denna portering är exempel på information som framkom ur dessa intervjuer. Intervjuerna var semi-strukturerad eftersom tanken bakom intervjuerna var att det skulle leda till öppna gruppdiskussioner runt frågorna. Svaren till frågorna antecknades ner samtidigt som intervjun pågick. Eftersom examensarbetet bestod av två personer kunde den ena anteckna medan den andra intervjuade. Risken för att man missade något under intervjun kunde därför minimeras.

Informationen som dokumenterades ner från intervjun bearbetades därefter på följande sätt för att få ut information som är relevant till frågeställningen:

1. Båda examensarbetarna läste igenom intervjun noggrant.
2. All information som svarade på frågeställningen plockades ut.
3. Informationen som plockades ut skrevs ner i ett nytt dokument på google drive.

Kraven på prototypen framkom ur en intervju som gjordes med handledaren från Saab den 26 februari 2014. Den var speciellt viktigt eftersom resultatet av examensarbetet var baserad på den informationen. Kravet på prototypen var att den skall använda sig av naturlig Objective-C kod när den körs i en iOS miljö. Utan den informationen var området väldigt brett. Det fanns nämligen väldigt många olika verktyg för att portera en applikation till en annan plattform. Gemensamt för dessa verktyg är att de behåller samma kod vid porteringen och går under namnet multi-plattform utveckling. För mer information angående multi-plattform utveckling, se avsnitt 3.1.
Med kravet på prototypen kunde området för informationsökningen begränsas.

För mer information angående teorin bakom intervjuer, se Kvalitativ metod [14] i kapitel 7.

2.4 Gruppdiskussion

Som nämndes i avsnitt 2.2 så var intervjuerna semi-strukturerad. Dessa intervjuer gick oftast över till gruppdiskussioner där man diskutera kring en frågeställning. Ett exempel på frågeställning som utvecklades till en gruppdiskussion är:

- *Vad Android applikationen används till?*

Ur denna frågeställning kunde följande information erhållas:

- Vad applikationen användes till
- Hur applikationen fungerade i verkligheten
- Anledningen till varför de vill portera Android applikationen
- Vilka plattformar som applikationen stödjer förnärvarande

Gruppdiskussioner användes även till att undersöka källkoden för Android applikationen som skulle användas till porteringen. Ur dessa diskussioner kunde följande information erhållas:

- Vilken funktion som handledaren rekommenderade att implementera först
- Hur denna funktion fungerade i detalj
- Hur man kan förbättra koden

För teorin bakom gruppdiskussioner, se Skop [17] i kapitel 7.

2.5 Förstudier

Det krävdes förstudier inom ämnet att portera en Android applikation till iOS, eftersom ingen av examensarbetarna och Saab hade kunskaper inom detta ämne sedan tidigare. Det var också osäkert om det ens var möjligt att hitta en effektiv metod att portera en Android applikation till iOS med.

2.5.1 Bestämning av sökord

För att kunna veta vilken information som skulle inhämtas bestämdes en fråga utifrån målet med examensarbetet:

- *Hur skall man kunna portera en existerande Android applikation så att den stödjer iOS plattformen på ett effektivt sätt.*

Sökorden, se Tabell 2.1, som användes tillsammans med sökmotorerna beskrivna i avsnitt 2.4.2. för att göra informationsökningen är baserad på denna fråga.

Tabell 2.1: Sökord

port android to ios
port apk files to ipa
translate an android application to ios application
is porting an android application to ios possible?
easiest way to port android app to ios
how to port android to ios

2.5.2 Sökmotorer

Informationsinhämtningen gjordes med litteraturstudier och med hjälp av sökmotorerna LUP, LUBsearch och google. Ingen information angående portering mellan Android och iOS kunde hittas genom att studera litteraturer. När sökmotorerna LUP och LUBsearch användes tillsammans med sökorden i Tabell 2.1 gav det heller ingen användbar information. När sökmotorn google användes tillsammans med sökorden i Tabell 2.1 kunde mycket användbar information hittas. Informationsökningen har sedan enbart gjorts med hjälp av sökmotorn google. Examensrapporten är baserad på den information som hämtades från källorna som finns presenterade i kapitel 7.

2.5.3 Bearbetning av information

Bearbetningen av den inhämtade informationen gjordes med hjälp av en egen metod (se Fig. 2.1.). Modellen är baserad på Lauesens [15] verifikations metod och på informationen som hämtades från källorna presenterade i kapitel 7. Eftersom informationen som hämtades bestod av olika sorters kod konverteringsverktyg, behövdes en metod där man kan testa dessa verktyg så att det stämde överens med kravet på prototypen.

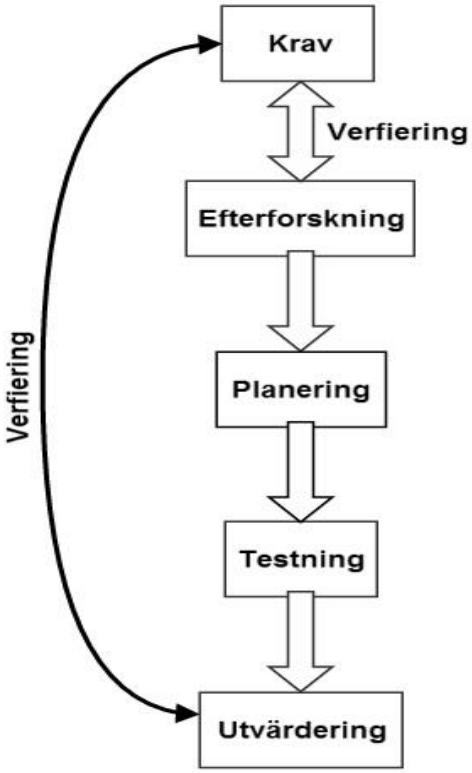


Fig. 2.1. *bearbetning av information*

Alla verktyg som var relevant till examensarbetet bearbetades med hjälp av modellen i Fig. 2.1. på följande sätt:

Modellens arbetsflöde:

1. Efterforskning
    a. En efterforskning görs för varje verktyg som hittas.
    b. Man går endast över från efterforskning till planering om målet med verktyget stämmer överens med kravet för prototypen.
    c. En ny informationsökning för att hitta ett mer passande verktyg görs om målet med verktyget inte stämmer överens med kravet.
2. Planering
    a. En planering görs ifall målet för verktyget stämmer överens med kravet på prototypen.
    b. En planering där man ta fram en metod över hur verktyget skall testas görs
    c. När planeringen är färdig går man över till testning
3. Testning
    a. Under testningen används metoden som togs fram under planeringen
    b. När testningen är färdig görs en utvärdering av resultatet.
4. Utvärdering
    a. En utvärdering över hur testningen gick görs.
    b. En utvärdering av resultatet, användbarheten, eventuella problem och om det verkligen stämmer överens med kravet för prototypen görs.

Fördelen med modellen i Fig. 2.1. är att man hela tiden kan göra en verifiering av den inhämtade informationen mot kravet som ställdes på prototypen av Saab. Enligt Lauesen [15] så är verifikation en teknik där man hela tiden verifierar så att produkten man utveckla stämmer överens med kravet på produkten.
Nackdelen med modellen är att om det hade varit flera metoder som stämde överens med kravet som Saab ställde, så hade man fått gå igenom hela modellen för varje verktyg. Detta skulle ha tagit oerhört lång tid då man måste planera hur man skall testa varje verktyg.

Hela modellen behövde bara genomföras en gång eftersom det enda verktyget som hittades och som stämde överens med målet var J2ObjC. Det är ett verktyg utvecklat av Google som översätter Java kod till naturligt Objective-C kod.
Flera multi-plattform utvecklingsverktyg hittades under informationsinhämtningen. Codename one och PhoneGap är exempel på sådana verktyg. Vad som karaktärisera dessa verktyg är att samma kod kan användas till flera plattformar. Dessa verktyg kom till 1-C i efterforskningsfasen och förkastades. Anledningen till varför dessa verktyg förkastades var för att om man skulle ha använt dessa verktyg så kommer den resulterade iOS prototypen inte använda sig av naturlig Objective-C kod, vilket strider mot kravet för prototypen.

Fokus på analysen, se kapitel 3, i denna examensrapport kommer ligga på J2ObjC. I analys kapitlet kommer även en lite mer utförlig beskrivning av multi-plattform utveckling och varför det inte är lämpligt för detta examensarbete att göras.

För mer information angående de olika stegen i informationsinhämtningsprocessen, se Holmqvist [10] i kapitel 7.

2.6 Prototyp utvecklingsprocess

Implementationen av en fungerande prototyp gjordes med hjälp av resultatet av analysen i kapitel 3. Tanken från början var att låta analysen bestämma vad som var möjligt att portera i den applikation som var given av Saab. Men eftersom examensarbetet gjordes tillsammans med Saab så var målet med hur mycket som skulle porteras redan bestämd från början.

2.6.1 Inlärning av Objective-C

Kunskapen inom Objective-C är viktigt eftersom J2ObjC inte kan översätta kod relaterad till Android API dvs. kod som är relaterad till användargränssnittet [5]. Eftersom ingen av examensarbetarna hade kunskaper inom Objective-C sedan tidigare behövdes en inlärningsprocess. Examensarbetarna fick studera Objective-C språket med hjälp av instruktionerna som finns på Apples hemsida för utvecklare [2].
Inlärningen av Objective-C språket planerades aldrig in i tidsplaneringen, eftersom verktyget J2ObjC som skulle användas till porteringen var okänt från början. Då det var klart hur J2ObjC fungerade var det redan brist på tid. Inlärningen av Objective-C fick därför ske parallellt med kravhanteringen och implementationen av prototypen. Men tack vare att Saab hade utvecklare som hade erfarenhet i Objective-C kunde examensarbetarna får mycket hjälp.

2.6.2 Kravhantering
Eftersom Saab inte hade någon detaljerande kravspecifikation över Android applikationen som skulle porteras, behövdes en mer detaljerad kravspecifikation skapas. Kravspecifikationen som skapades innehöll enbart målet över hur mycket som skulle porteras, dvs. kravspecifikationen täcker inte hela Android applikationen. Alla kraven i kravspecifikationen är skrivna som ”user-stories”.
Av konfidentiella skäl var det inte tillåtet att ta med kravspecifikationen över applikationen i examensrapporten. Alla ”user-story” som användes i avsnitt 3.7 är därför påhittade.

2.6.3 Validering
Gruppdiskussioner med handledaren hölls kontinuerligt under hela examensarbetets gång för att se till så att man ha förstått funktionerna korrekt och att projektet löper på i rätt riktning. Med hjälp av handledaren kunde kraven på funktionerna skrivas ner i kravspecifikationen samtidigt som en validering av kraven mot målet görs.

För mer information om validering, se Lauesen [15] i kapitel 7.

2.6.4 Implementering av prototyp
I början av implementeringen av prototypen användes metoden scrum för att strukturera upp arbetet i utvecklingsprocessen. Dock gick första sprinten i utvecklingsprocessen inte som förväntat. En justering av scrum metoden behövde därför göras. Den nya metoden fick namnet scrumban. Mer om vad scrumban innebär och varför justeringen av scrum metoden var nödvändigt förklaras längre ner i detta avsnitt.
När scrum användes gjordes en del undantag från scrum modellen. Scrum modellen som användes under utvecklingsprocessen ser ut på följande sätt:
Komponenterna som användes från scrum modellen

- Sprint planning
- 1 veckas sprint
- Sprint review
- Daily scrum
- Scrumboard

Undantag som gjordes från scrum modellen

- Product backlog
- Roller

Eftersom examensarbetet gjordes av två personer definierades inga roller. Istället bestämde examensarbetarna tillsammans vad som skall göras och hur man skulle lösa de olika arbetsuppgifterna.
Ingen product backlog skapades eftersom kravspecifikationen användes som en product backlog istället. Anledningen till detta beslut var för att kravspecifikationen liknar en product backlog. Det är väldigt tidskrävande om man behöver skapa ytterligare ett dokument som kommer att innehålla nästan identisk information som kravspecifikationen. Därför gjordes all prioritering i kravspecifikationen istället.

Scrumban
Som nämndes ovan gick första sprinten i utvecklingsprocessen inte som förväntat. Anledningen till att första sprinten misslyckades var för att studerandet av källkoden för applikationen som skulle porteras inte var tillräckligt. Det fanns nämligen funktioner som var dolda i applikationen och som måste göras först. För att undvika att fler sådana problem uppstår behövdes en justering av scrum modellen. Regeln som förbjuder en att lägga till nya uppgifter i en scrumboard under en pågående sprint måste tas bort.
Med detta i åtanken hittades en annan agil metod som heter kanban. Kanban använder sig inte av sprintar och metoden tillåter att man prioritera om uppgifterna som skall implementeras, så att en annan uppgift kan göras först. Denna egenskap från kanban lades till i scrum modellen samtidigt som sprintar togs bort. Modellen som användes under utvecklingsprocessen fick namnet scrumban och använder sig av följande komponenter.

Komponenterna som användes i scrumban:

- **WIP-limits -** begränsa antalet pågående uppgifter
- **Planning -** ett möte hålls om det uppstår problem under utvecklingen
- **Review -** en validering med handledaren görs för att se så att funktionerna är implementerat korrekt
- **Daily scrum -** varje vecka hålls ett möte där man berätta;
    1. vad man ha gjort
    2. om det har uppstått några problem
    3. Vad man ska göra härnäst.
- **Scrumboard -** en tavla där man hänger upp alla uppgifterna som skall implementeras.

Denna justering av modellen medförde att en ändring i tidsplanen behövde göras. Den slutliga tidsplanen kan man hitta i appendix B.1.

För mer information angående teorin bakom scrum och scrumban, se Gambill [9] i kapitel 7.

2.6.5 Rapport

Förutom slutrapporten så skrevs det även en analysrapport till Saab. Analysrapporten skrevs först eftersom rapporten var grunden till den metod som skulle användas till porteringen av Android applikationen. Analysen i kapitel 3 är också baserad på analysrapporten.

2.6 Källkritik

Majoriteten av information som användes i denna examensrapport är tagen från Internet. För några år sedan fanns det inga telefoner i mobiltelefonbranschen som kallades för ”Smartphone”, dvs. en mobil som kan fungera som en dator [22]. Efter införandet av ”Smartphones” så har det skett en stor förändring i dagens mobiltelefonbransch. Från att inte har någon kännedom av ”Smartphones” till att de introducerades, så har antal användare av sådana telefoner ökat enormt. Under år 2013 ägde 92 % av befolkningen i Kina en ”Smartphone” [23]. Mobiltelefonbranschen är därför en bransch som förändras snabbt, vilket medför att det ta inte lång tid förrän en artikel, som kan handla om den senaste tekniken som används i telefoner, blir föråldrad. De flesta artiklarna som användes är därför bara ett eller två år gamla. Mycket av informationen är tagna från artiklar skrivna av olika applikationsutvecklare för att försöka få så pålitligt fakta som möjligt. Mindre pålitliga källor så som forum har undvikits, eftersom det kan finnas okunniga personer i ett forum som bidrar med icke pålitligt information.

J2ObjC är ett verktyg utvecklat av Google som fortfarande är under utveckling. När J2ObjC undersöktes så användes projektets egen hemsida [13] eller hemsidor direkt länkade från projektets hemsida. Anledningen till varför projekthemsidan användes var för att få den senaste informationen som finns för tillfället och för att få den pålitligaste informationen som möjligt. Upphovsmannen till materialet på projektets hemsida är Tom Ball som är ägaren till projektet [13]. Dessa källor anses pålitliga eftersom verktyget är utvecklat av Google som ligger bakom sökmotorn google som har miljontals användare [18]. Det gör ingen nytta för projektet att sprida falsk information, då syftet med informationen på hemsidan är för att kunna sprida J2ObjC verktyget till så många utvecklare som möjligt. Nya fel upptäcks när olika utvecklare använder sig av verktyget till sina projekt. Detta är anledningen till varför J2ObjC är ett open-source verktyg [5]. Balls kontaktinformation finns på hemsidan och informationen på hemsidan uppdaterades januari 2014 [13]. Därför anses informationen på hemsidan som pålitligt och aktuellt.

Kapitel 3

Analys

I detta kapitel kommer först en kort introduktion till vad multi-plattform utveckling är och varför det inte var lämpligt till detta examensarbete att göras. För att kunna förstå resten av examensrapporten kommer J2ObjC, ett verktyg som är utvecklad av Google, sedan att förklaras.

3.1 Multi-Plattform utveckling

Flera multi-plattform utvecklingsverktyg hittades under sökningen efter en effektiv metod att portera Saabs applikation till iOS. PhoneGap [16] och Codename one [1] är exempel på verktyg som hittades. Vad dessa verktyg har gemensamt är att de kan användas till att utveckla applikationer som kan köras på flera olika plattformar med samma kod. “Write once, run everywhere!” är ett uttryck skapad av Sun Microsystem för just multi-plattform utveckling [8].

Phonegap är ett exempel på ett open-source ramverk som gör det möjligt för en utvecklare att skapa mobila applikationer med hjälp av HTML, CSS och JavaScript. Det kan även användas för att omvandla en existerande hemsida till en applikation. Dessa applikationer kan sedan köras på alla plattformar som stödjer denna applikation utan att en omskrivning av kod behövs. PhoneGap omvandlar applikationen till en ny applikation genom att kapsla in koden och gör den nya applikationen redo att bli installerad, enligt Rinaldi [16].

Codename one [1] är ett annat verktyg som kan användas för att utveckla till flera plattformar samtidigt utan att en omskrivning av kod behövs. iPhone, Android, Blackberry, Windows Phone 7 är exempel på plattformar som den kan utveckla till. När man använder sig av verktyget till applikationstutveckling så skrivs koden i Java. Målet med Codename one är att man ska kunna utveckla snabbt till flera plattformar samtidigt som man behåller prestandan [1].

När man vill utveckla en applikation som ska stödja både Android och iOS kan man även använda sig av programmeringsspråket C++. Enligt Begemann [7] har både Android och iOS stöd för C++. Med denna metod kan man återanvända kod som inte är relaterad till användargränssittet. Kod för logiken bakom en funktionalitet som är skriven i C++ är exempel på kod som kan återanvändas.

Men multi-plattform utveckling var ingen möjligt lösning till detta examensarbete efter en diskussion med handledaren på Saab. Enligt handledaren så är det viktigt att Android applikationen som skall porteras, använder sig av naturlig Objective-C språk när den körs i en iOS miljö.

3.2 J2ObjC

Med målet av examensarbetet och kravet på prototypen i åtanken, se avsnitt 1.3, hittades ett verktyg med namnet J2ObjC. Det är ett verktyg utvecklat av Google som kan översätta Java kod till Objective-C kod. Även om verktyget fortfarande är i alfa- och beta stadiet så kan den översätta det mesta av Java språket [13]. En lista över vad J2ObjC kan konvertera finns på deras projekt hemsida [4].

Fördelar
Som nämndes i inledningen till kapitel 1 så måste koden i en applikation skrivas om för att kunna stödja en annan plattform. Anledningen till att denna förändring är nödvändigt är pga. olikheten i programmeringsspråken för de olika plattformarna. Det nämndes också i inledningen att en funktion i en applikation kommer att ha samma funktionalitet när applikationen körs i en annan plattform. Om koden för funktionen måste skrivas om hela tiden för varje ny plattform kommer det vara väldig ineffektivt. "Don't Repeat Yourself" principen som Ball skrev på J2ObjCs projekt hemsida [13] är därför väldig viktigt. Enligt Ball [5] så är detta målet och fördelen med J2ObjC. J2ObjC kan därför översätta koden bakom en funktionalitet utan att utvecklaren behöver skriva om koden till ett annat programmeringsspråk.

Nackdelar
Nackdelen med J2ObjC är förnärvarande att verktyget inte kan garantera översätta allt relaterad till Java. Ett exempel på en sådan nackdel är att J2ObjC inte kan översätta Java kod relaterad till Android API. Utvecklarna av J2ObjC nämnde på deras projekt hemsida [13] att J2ObjC inte kommer vara försett med användargränssnitt översättning. De har heller inga planer på att implementera den funktionen. J2ObjC kan därför enbart översätta kod skriven i ren Java. Detta medför att användargränssnittet måste skrivas om till naturlig iOS språk som är baserad på iOS SDK. Anledningen till varför de inte förser J2ObjC med användargränssnitt översättningen är pga. användargränssnitt standarden som iOS har [5]. Ett exempel är iPhone, till skillnad från Android telefoner, inte har någon bakåt funktion på deras apparater som möjliggör navigering till föregående skärm. iPhone applikationsutvecklarna får själva implementera bakåt funktionen till applikationerna enligt iOS standarden (se Fig. 3.1.).

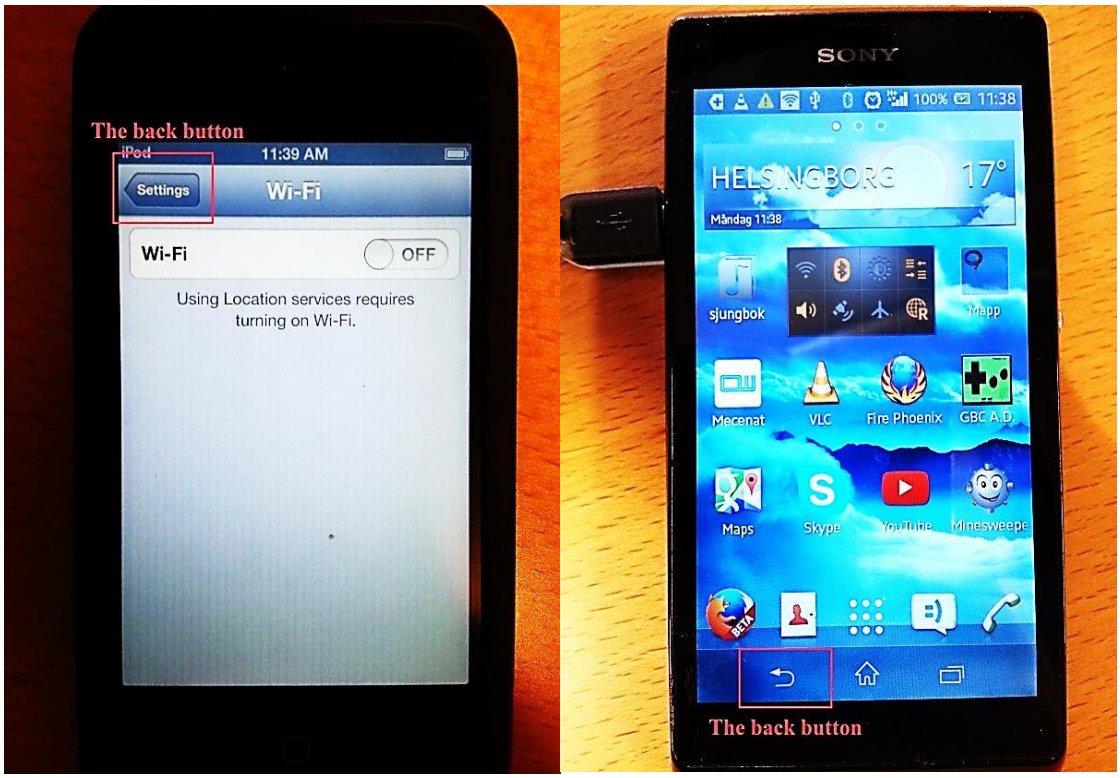


Fig. 3.1. *Bakåt funktion för iPhone (vänster bild) och Android telefoner (höger bild).*

För mer riktlinjer angående iOS standarden för användargränssnittet kan man hitta på Apples hemsida för utvecklare [3].

3.3 Mjukvarukrav för J2ObjC

Som förklarades i "Getting Started" av Ball [6] så kan J2ObjC användas på tre operativsystem för att översätta Java kod:

- Mac OS X 10.7 eller högre (Apples operativsystem)
- Windows
- Linux

Den översatta koden kan dock inte bli kompilerad på Windows eller Linux. Apples datorer, så som iMac, behövs fortfarande för att kunna kompilera koden [6].
J2ObjC [13] är kompatibel med de flesta utvecklingsmiljöer såsom:

- Eclipse med JDK 1.6 eller högre
- XCode version 4 eller högre

XCode är utvecklingsmiljön som används till att skapa iOS applikationer för Apples mobila plattformar. XCode, till skillnad från andra utvecklingsmiljöer såsom Eclipse, finns enbart till Apples egna operativsystem.

3.4 Hårdvarukrav

Kraven på hårdvaran kommer egentligen från kraven på mjukvaran. XCode 5 stödjer inte utveckling för iOS versioner lägre än 4.3 [12]. De äldsta iPhone och iPod modellerna som man kan utveckla applikationer till är därför:

- iPhone 3GS
- iPod touch 3G

Alla iPad och iPad mini modellerna är kompatibel med XCode 5 [12].

3.5 Programmeringskunskaper som krävs för J2ObjC

Android applikationen som är given av Saab är skriven i språket Java. Eftersom denna applikation skall porteras till iOS så är det nödvändigt att ha kunskaper i Java och Objective-C. Objective-C är programmeringsspråket som används till iOS plattformar. En utvecklare behöver dock inte har kunskaper relaterad till Android API för att kunna använda J2ObjC. Som nämndes i avsnitt 3.2; J2ObjC kan inte översätta kod relaterad till Android API, eller med andra ord kod som är relaterad till användargränssnittet. Detta är anledningen till varför denna kunskap inte är nödvändigt.
Kunskapen i Objective-C är nödvändigt för att kunna förstå den översatta koden från J2ObjC. Detta är pga. att utvecklaren måste förstår hur man ska modifiera den översatta koden (se Appendix A.1-A.4). Den översatta koden kan inte kompileras direkt i utvecklingsmiljön XCode eftersom en del av koden måste skrivas om manuellt (se Fig. 3.2.). I Fig. 3.2. har man översatt metoder från Java klassen Math:

- Math.max - returnera det största talet mellan två tal
- Math.min - returnera det minsta talet mellan två tal
- Math.pow - returnera resultatet av tal 1 upphöjt till tal 2
- Math.sqrt - returnera roten av ett tal

De översatta metoderna måste ersättas med iOS motsvarighet (se Appendix A.1-A.4).
För mer information angående hur översättningsprocessen ser ut, se avsnitt 3.7.

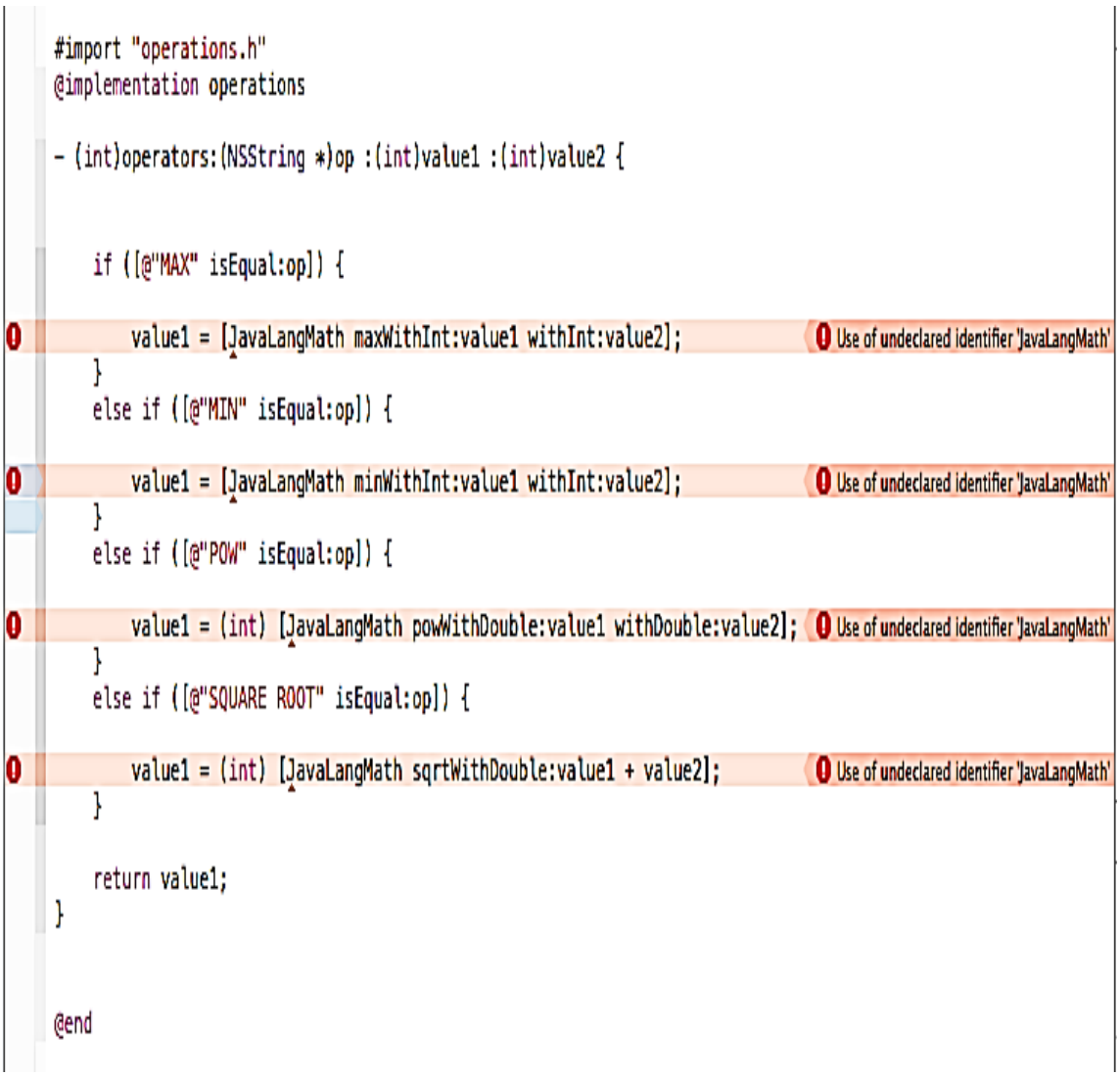


Fig. 3.2. *Problem när den översatta koden användes direkt i XCode* (2014)

3.6 Metod för utvärdering av J2ObjC

J2ObjC användes till att portera Android applikationen given av Saab. Om J2ObjC uppfyller målet med examensarbetet bestämdes av hur effektivt verktyget är på att översätta Java kod till Objective-C kod.
I utvärderingen användes Eclipse som översättningsverktyget, eftersom J2ObjC installerades i Eclipse. XCode användes till att exekvera den översatta koden, då Eclipse inte kunde det.

Verktyg

- J2ObjC
- Eclipse
- XCode

Metod för att testa J2ObjC

Metoden för att testa effektiviteten och tiden det tog att göra en översättning:

- Implementera en ”user-story” med hjälp av J2ObjC
- Implementera en ”user-story” utan J2ObjC

Testet gick ut på att först implementerar en ”user-story” från applikationens kravspecifikation med J2ObjC. Tiden det tog att implementera ”user-storyn” från början till slut skrevs ner i Tabell 3.1. Samma ”user-story” översattes sedan utan användandet av J2ObjC på samma sätt. Ordningen ändrades för nästa ”user-story”, där en översättning utan J2ObjC gjordes först. Det här gjordes för 4 ”user-stories” för att minimera felmarginalen under jämförelseprocessen. Tiden det tog för att implementera varje ”user-story”, bestämde om J2ObjC är den effektivaste metoden att använda till portering av applikationer

Av konfidentiella skäl så är alla ”user-story” som presenteras i examensrapporten påhittade.

3.7 Översättningsprocessen för J2ObjC

Testet gjordes av en utav examensarbetarna. Resultatet är baserat på att examensarbetaren redan hade full förståelse för hur Java koden behövde skrivas om till Objective-C för att uppnå samma resultat från början. Examensarbetaren kunde omedelbart påbörja översättningen utan att göra några ytterligare informationsinhämtningar.

Som det framgår i Fig. 3.3 så består översättningsprocessen för J2ObjC av två faser:

1. Översättningen av Java koden med J2ObjC till Objective-C kod
2. Modifiering av den översatta koden.

Anledningen till varför koden måste modifieras är pga. att allt inte har översatt helt korrekt (se Fig. 3.2.), vilket medför att koden inte kan kompileras i XCode. Den översatta koden måste därför modifieras manuellt.

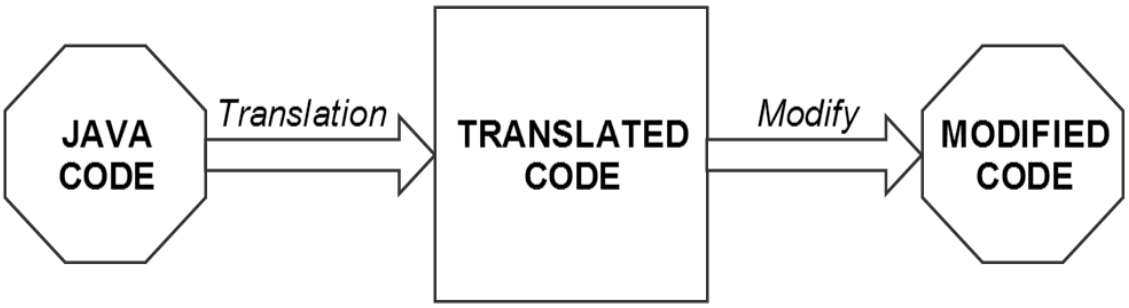


Fig. 3.3. *Översättningsprocessen* (2014)

Utvärderingen av J2ObjC verktyget gjordes för följande 4 ”user-stories”. Dessa ”user-stories” är skrivna på engelska pga. att kravspecifikationen är skriven på engelska.
För kod till:

- ***User-Story 1.2.4***, se Appendix A.1
- ***User-Story 1.2.5***, se Fig. 3.4, Fig. 3.5, Fig. 3.6, Appendix A.2
- ***User-Story 1.2.6***, se Appendix A.3
- ***User-Story 1.2.7***, se Appendix A.4

User-Story 1.2.4:
As a user, I want the application to be able to do division, multiplication, addition and subtraction.
Acceptance criteria:

a) There shall be a function for division
b) There shall be a function for multiplication
c) There shall be a function for addition
d) There shall be a function for subtraction

User-Story 1.2.5:
As a user, I want the application to be able to find the highest value as well as the lowest value between two values. I also want the application to be able to perform value1^{Value2} and take the sum of two values and perform square root with it.
Acceptance criteria:

a) There shall be a function for finding the highest value between two values
b) There shall be a function for finding the lowest value between two values
c) There shall be a function for taking value1 and raised it to the power of value2
d) There shall be a function for performing square root on the sum of two values

User-Story 1.2.6:
As a user, I want the application to be able to add new members
Acceptance criteria:

a) There shall be a function for adding the nickname of a new member

User-Story 1.2.7:
As a user, I want the application to be able to encrypt messages so that the message remain hidden to outsiders.
Acceptance criteria:

a) There shall be a function to encrypt messages

I bilderna nedan kan man se ett exempel på hur översättningen och modifieringen av koden för **User-Story 1.2.5** se ut.

- I Fig. 3.4 kan man se hur Java koden för **User-Story 1.2.5** ser ut innan översättningen.
- I Fig. 3.5 kan man se hur den översatta Java koden ser ut.
- I Fig. 3.6 kan man se hur modifieringen av den översatta koden ser ut.

```
public int operators(String op, int value1, int value2){
  if(op==”MAX”){
        value1=Math.max(value1, value2);
  }else if(op==”MIN”){
        value1=Math.min(value1, value2);
  }else if(op==”POW”){
        value1=(int)Math.pow(value1, value2);
  }else if(op==”SQUARE ROOT”){
        value1=(int)Math.sqrt(value1+value2);
  }
  return value1;
}
```

Fig. 3.4. *Java koden som skall översättas*

```
#line 9
- (int)operatorsWithNSString(NSString *)op
                              withInt:(int)value1
                              withInt:(int)value2{
#line 10
   if ([@”MAX” isEqual:op]){
#line 11
          value1=[JavaLangMath maxWithInt:value1 withInt:value2];
   }
   else if ([@”MIN” isEqual:op]){
#line 13
          value1=[JavaLangMath minWithInt:value1 withInt:value2];
   }
   else if ([@”POW” isEqual:op]){
#line 15
          value1=(int)[JavaLangMath maxWithDouble:value1 withDouble:value2];
   }
   else if ([@”SQUARE ROOT” isEqual:op]){
#line 17
          value1= (int)[JavaLangMath sqrtWithDouble:value1 + value2];
   }
#line 19
   return value1
}
```

Fig. 3.5. *Den översatta Java koden*

```
- (int)operators: (NSString *)op:(int)value1:(int)value2{

          if ([@”MAX” isEqual:op]){
                value1=MAX(value1, value2);
          }
          else if ([@”MIN” isEqual:op]){
                value1=MIN(value1, value2);
          }
          else if ([@”POW” isEqual:op]){
                value1=(int) pow(value1, value2);
          }
          else if ([@”SQUARE ROOT” isEqual:op]){
                value1= (int) sqrt(value1, value2);
          }
          return value1
}
```

Fig. 3.6. *Efter modifieringen av den översatta koden*

3.8 Resultat från utvärderingen av J2ObjC

Tabell 3.1: Resultatet av user-story implementationen med och utan J2ObjC

	Tiden det tog att utveckla en ”user-story” med J2ObjC (minuter)	Tiden det tog att utveckla en” user-story” utan J2ObjC (minuter)
(Implementerat med J2ObjC först) **User-Story 1.2.4**	3 min	6 min
(Implementerat utan J2ObjC först) **User-Story 1.2.5**	3 min	5 min
(Implementerat med J2ObjC först) **User-Story 1.2.6**	3 min	5 min
(Implementerat utan J2ObjC först) **User-Story 1.2.7**	4 min	7 min

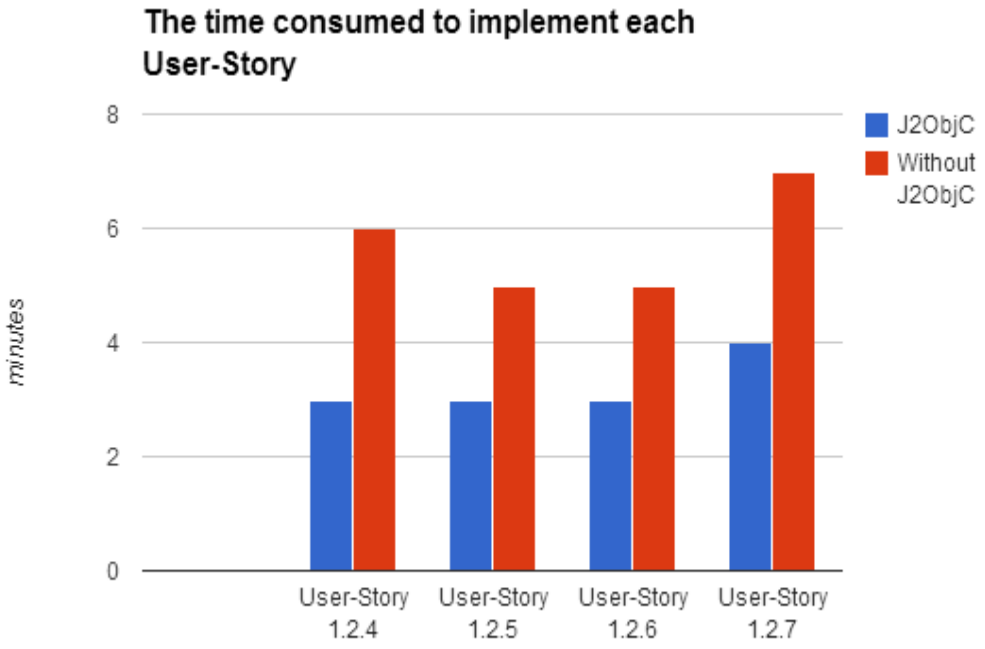


Fig. 3.7. *Resultatet av user-story implementationen med och utan J2ObjC*

I Tabell 3.1 och Fig. 3.7 kan man se resultaten för implementationen av de 4 ”user-stories” i avsnitt 3.7. De 4 ”user-stories” är implementerade både med och utan J2ObjC. Som nämndes i avsnitt 3.6 skiftades ordningen för användandet av J2ObjC. ”User-story” 1.2.4. i Tabell 3.1 implementerades först med J2ObjC och därefter utan. För nästa ”user-story” i tabellen bytes ordningen. Detta gjordes för att minska på felmarginalen vid jämförelseprocessen.
Av resultaten i Tabell 3.1 kan man se att användandet av J2ObjC minskade på tiden det tog att implementera en ”user-story”. Detta framgår speciellt tydligt i Fig. 3.7. Den blåa stapeln i Fig. 3.7 var när man implementerade en ”user-story” med hjälp av J2ObjC. Den röda stapeln var när man implementera utan J2ObjC. Resultaten visar att tiden det tog för att implementera en user-story minskades när J2ObjC användes.

3.9 Användbarheten för J2ObjC

Som nämndes i avsnitt 3.3, så kan J2ObjC installeras i antingen XCode eller Eclipse. I utvärderingen användes J2ObjC med Eclipse. J2ObjC installerades som ett plugin i Eclipse och inga problem uppstod under installationen.

J2ObjC var väldigt enkelt att använda eftersom det behövdes bara tre knapptryckningar för att starta en kod översättning i Eclipse (se Fig. 3.8.). En header fil och en implementations fil, som genererades efter översättningen, innehåller alla variabler och kod.

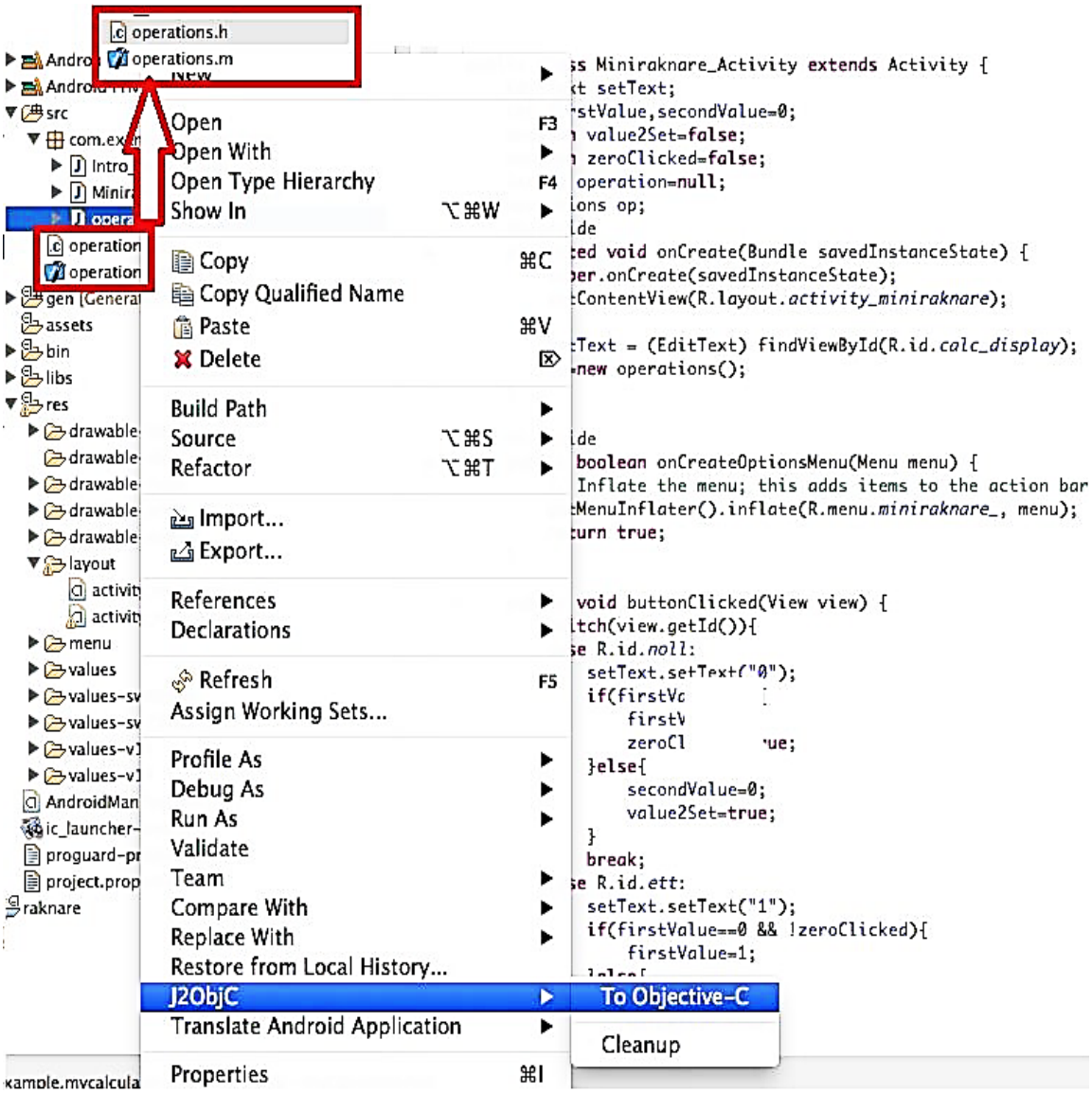


Fig. 3.8. *Start av J2ObjC översättning samt de genererade header- och implementations filerna efter översättningen* (2014)

När man använder Eclipse för översättning så måste man kopiera över koden till XCode projektet efter översättningsprocessen. En modifiering av koden behövs sedan för att kunna kompilera koden med XCode som nämndes i avsnitt 3.5.

3.10 Vad J2ObjC inte kunde översätta

Java klasserna Array, StringBuilder och Math användes i ”user-stories” koden för utvärderingen av J2ObjC. Översättningen fungerade som förväntat eftersom den översatta koden innehöll enbart ren Java kod (se Appendix A.1-A.4).

Vad som inte fungerade:

- att använda den översatta koden direkt i ett XCode projekt
- att översätta en klass som innehåller fel i koden
- att översätta en klass som innehåller kod relaterad till Android API

Att använda den översatta koden direkt, som nämndes i avsnitt 3.5, är inte alltid möjligt. I de flesta fall kommer utvecklaren att stöta på fel såsom framgår i Fig. 3.2. För att lösa dessa fel måste utvecklaren ersätta de översatta metoderna till motsvarigheten i Objective-C (se Appendix A.1-A.4).

En fördel med J2ObjC hittades under utvärderingsprocessen. Såsom framgår i Fig. 3.9. förbjuder J2ObjC att översätta en klass som innehåller fel. Det är en fördel eftersom detta medför att den översatta koden blir mer tillförlitligt

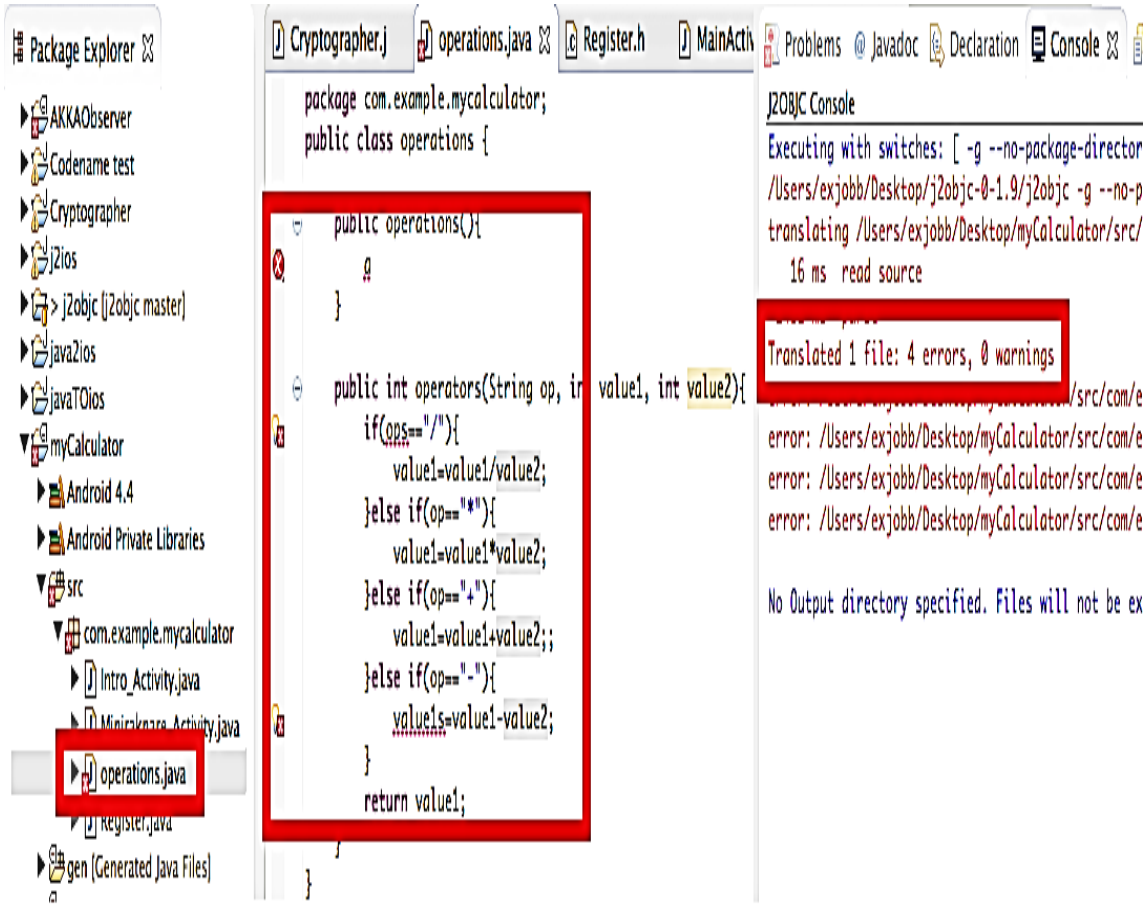


Fig. 3.9. *J2ObjC förbjuder översättning av klasser som innehåller fel* (2014)

Som framgår i Fig. 3.10. kan J2ObjC inte översätta kod relaterad till Android API. Det var förväntat eftersom utvecklarna till J2ObjC nämnde på deras projekt hemsida [13] att J2ObjC inte kommer att stödja användargränssnitt översättning. Om J2ObjC inte kan översätta användargränssnittet så kan den inte översätta en hel mobilapplikation direkt. Detta bevisa även Balls [5] påstående angående att j2ObjC inte är ett verktyg som kan portera en hel applikation direkt.

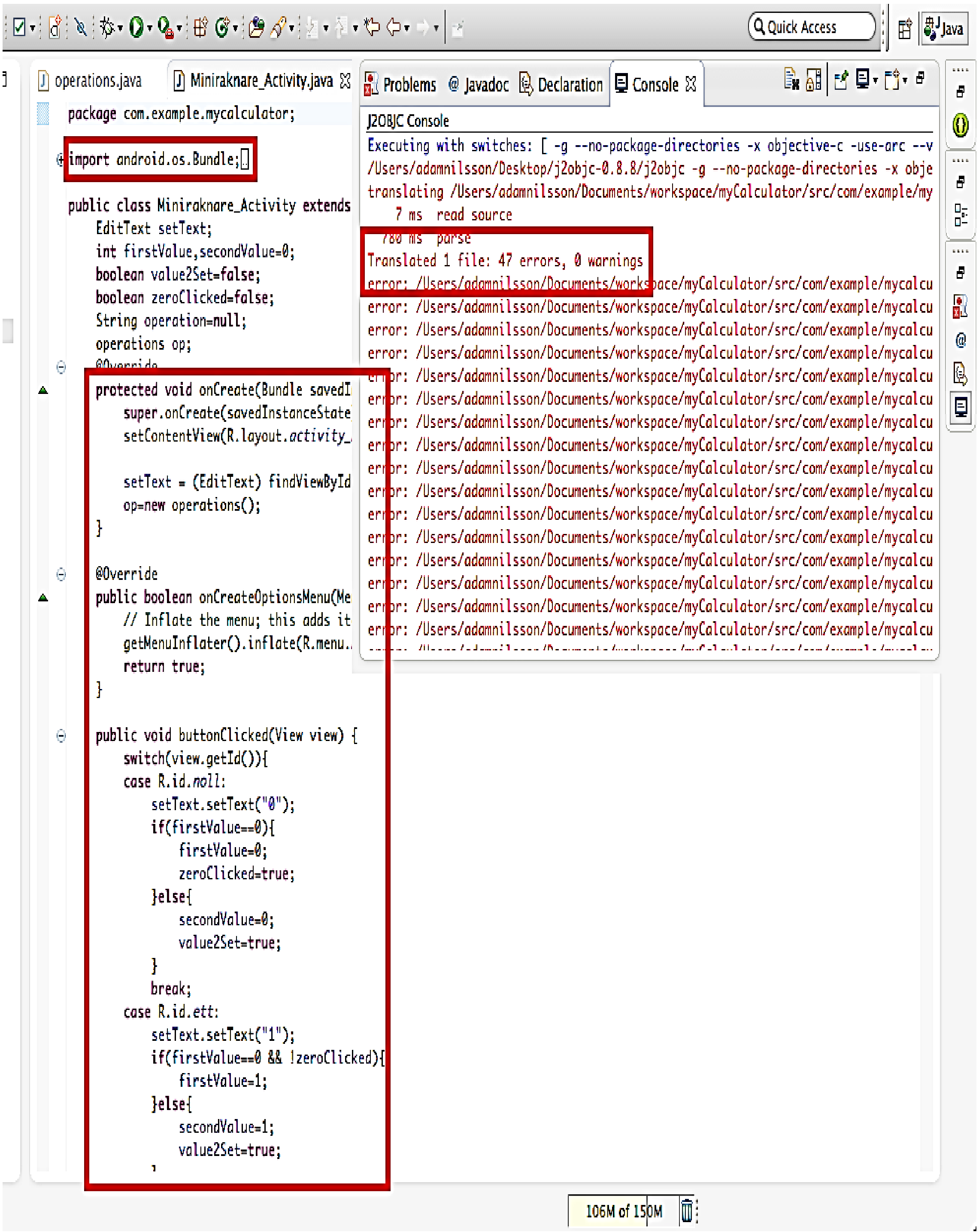


Fig. 3.10. *J2ObjC kan inte översätta kod relaterad till Android API* (2014)

Kapitel 4

Prototyp

4.1 Flödesdiagram

I Fig 4.1 visas flödesdiagrammet över prototypen som utvecklades. Bilderna är tagna från Android applikationen given av Saab för att kunna göra en jämförelse med bilderna på prototypen i avsnitt 4.2. Pilarna visar till vilka sidor man kan komma till om man är på en given sida. Mer specifikt hur man kommer till nästa sida beskrivs i avsnitt 4.2 för varje skärm.

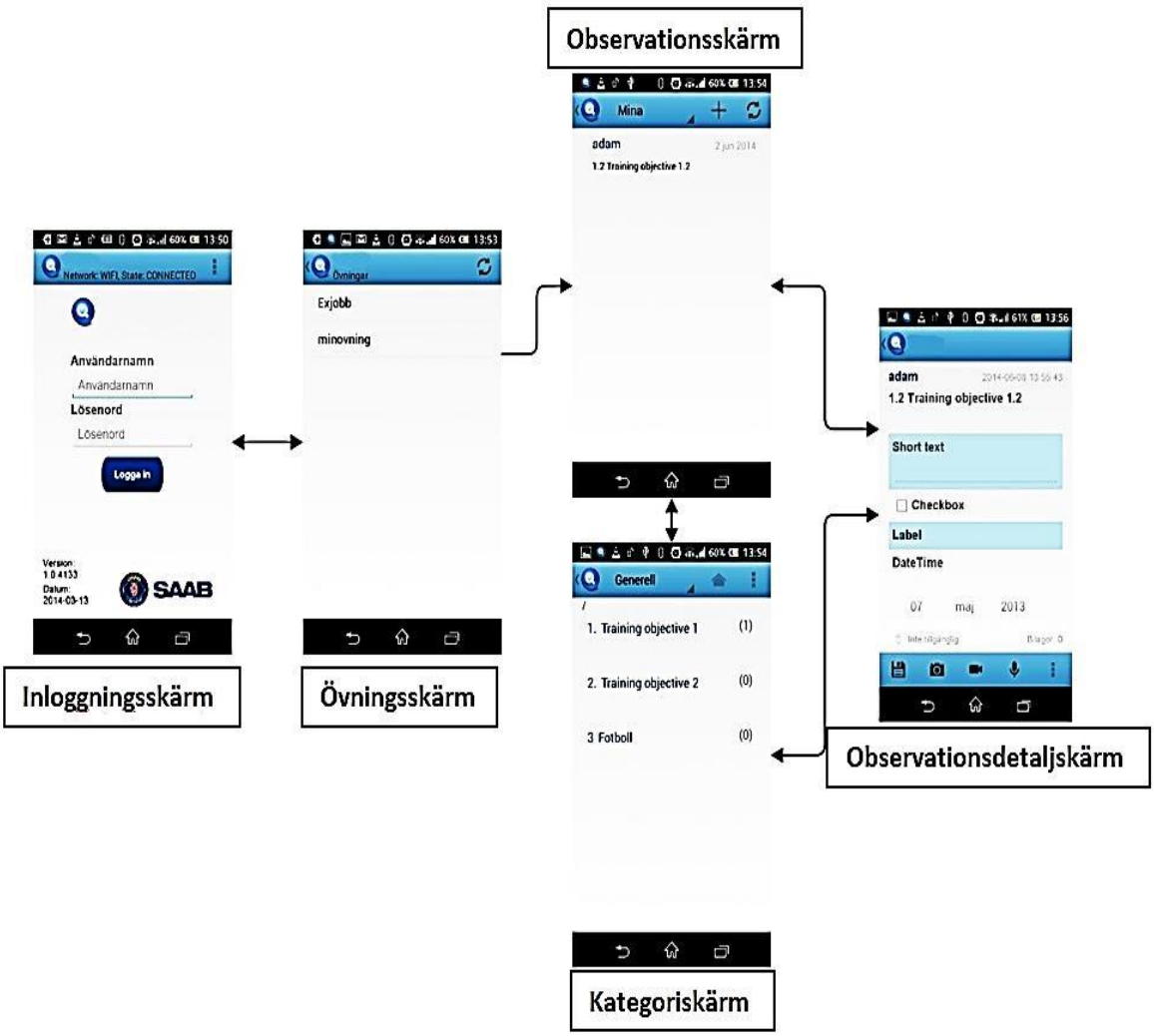


Fig 4.1. *Flödesdiagram för prototypen*

4.2 Resultat

4.2.1 Inloggningsskärm

När man starta applikationen kommer man till en *inloggningsskärm* där man kan skriva in sitt användarnamn och lösenord. Upp till höger finns en knapp för att komma till inställningarna för Internet. Om man lyckades logga in kommer man till *övningsskärmen*.

Fig 4.2. *Inloggningsskärm*

4.2.2 Övningsskärm

Efter man ha loggat in på applikationen kommer man till en sida som visar de övningar man är med i. I högra hörnet finns en knapp för att synkronisera övningarna som man är med i. Upp till vänster finns en knapp som tar en tillbacka till *inloggningsskärmen*. När man väljer en övning kommer man till *observationsskärmen*.

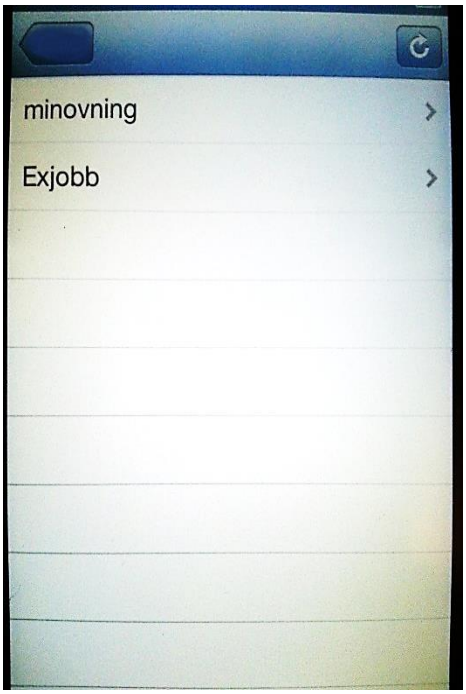


Fig 4.3. *Övningsskärm*

4.2.3 Observationsskärm

Efter att man ha valt övning kommer man till en sida som visar alla observationer för övningen man är med i. En observation är en mall som innehåller data av olika slag. Olika typer av bilagor kan finnas bifogade till observationen såsom bilder och ljud. Om man väljer någon observation kommer man till *observationdetaljskärmen* som visar datan för observationen. Upp till höger finns det en knapp för synkronisering av observationer. Bredvid synkroniseringsknappen finns det en knapp som tar en till *kategoriskärmen*

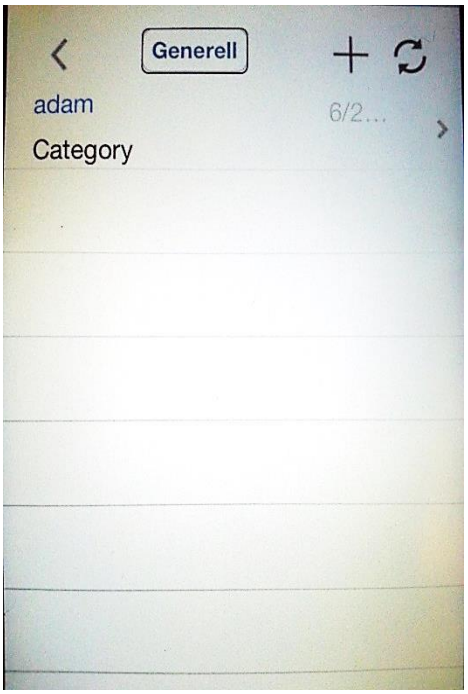


Fig 4.4. *Observationsskärm*

4.2.4 Kategoriskärm

Kategoriskärmen är till för att välja en observationsmall som man vill skapa. Mallarna har olika utseende för olika ändamål. Man trycker på den kategori man vill skapa en observation för. Kategorierna kan ha flera underkategorier. När man trycker på en kategori kommer man till *observationsdetaljskärm*. Knappen upp till vänster tar en tillbaka till *observationsskärmen*.

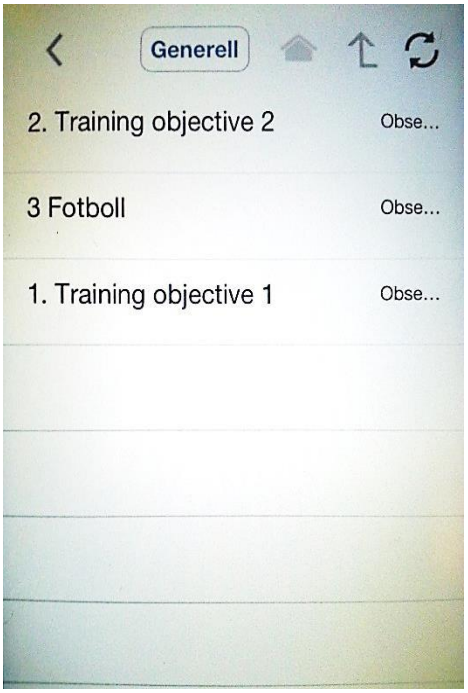


Fig 4.5. *Kategoriskärm*

4.2.5 Observationsdetaljskärm

Observationdetaljskärmen är till för att skapa en observation eller för att se en observation som var skapad tidigare. Observationerna har olika utseende beroende på till vilken ändamål den är skapad för. I Menyn som sitter nederst på skärmen finns funktioner för att ta bilder, spela in ljud, spela in en video och lägga till bilagor. Det finns även en knapp i menyn som sparar observationen. Upp till vänster finns det en knapp som tar en tillbaka till den sidan man var på innan, dvs. antingen *observationsskärmen* eller *kategoriskärmen*.

Fig 4.6. *Observationsdetaljskärm*

4.3 Vidareutveckling

Applikationen som är given av Saab är en stor applikation. Det som porterades i examensarbetet är en liten del av applikationen. Applikationen innehåller många andra funktioner som ej har porterats till prototypen. På grund av konfidentiella skäl kommer vi inte gå in närmare på vilka funktioner som Saab behöver implementerar om de väljer att fortsätta med prototypen.

Kapitel 5

Slutsats och framtida utredningar

5.1 Slutsats

Slutsatsen av analysen är att J2ObjC uppfyller målet med examensarbetet, dvs. att portera en Android applikation till iOS mer effektivt. Från resultatet i avsnitt 3.8 kan man se att J2ObjC kan översätta kod mycket snabbare i jämförelse med när man översatte koden för hand. Verktyget är lätt att använda, speciellt när den användes tillsammans med Eclipse för att göra översättningen. Däremot kunde ingen slutsats dras om användbarheten när J2ObjC används med XCode, eftersom denna kombination aldrig testades.

5.1.1 Resultat och rekommendationer

Med hjälp av informationen från analysen kunde därför frågorna från problemformuleringen besvaras.

Vilken mjukvara krävs för att kunna använda metoden?
Mjukvarukraven för att kunna använda J2ObjC är följande:

Operativsystem

- Mac OS X 10.7 eller högre
- Windows
- Linux

Utvecklingsmiljöer

- Eclipse med JDK 1.6 eller högre
- XCode 4 eller högre

Vilken hårdvara krävs för att använda metoden?
Eftersom XCode 5 är utvecklingsmiljön som används till att utveckla applikationer till iOS plattformar så beror hårdvarukraven på XCode 5. Då XCode 5 inte kan utveckla till lägre än iOS 4.3, medför det att enheterna som kan användas till applikationsutveckling är:

- iPhone 3GS eller högre
- iPod touch 3G eller högre
- Alla iPad och iPad mini är kompatibel

Vilka programmeringsspråk krävs för att kunna göra en portering med metoden?
Från de förberedande undersökningarna och resultatet från utvärderingen av J2ObjC så visade sig att kunskap i både Java och Objective-C är viktigt. Kunskapen i Objective-C är viktigt eftersom den genererade koden från en J2ObjC översättning måste bli modifierad innan den kan exekveras i ett XCode projekt. Att kunna Java är nödvändigt eftersom applikationen som skall bli porterad är skriven i Java. Innan någon modifikation av kod är möjligt måste utvecklaren veta exakt vad den översatta koden försöker åstadkomma.

Kommer metoden att göra porteringen mer effektivt och mindre tidskrävande?
Som framgår i avsnitt 3.8 kan en utvecklare med hjälp av J2ObjC översätta Java kod till Objective-C mycket snabbare än utan J2ObjC. Även om resultaten pekar på att J2ObjC kan översätta koder mycket snabbare behövs fler tester göras så att man inte dra för hastiga slutsatser. Eftersom inga beräkningsintensiva uppgifter testades med J2ObjC under utredningen av verktyget (se avsnitt 3.7), så är det svårt att dra några bra slutsatser utifrån resultatet. Vad som användes till översättningen var grundläggande Java klasser; Array, StringBuilder och Math. Inga slutsatser kunde därför dras om hur väl J2ObjC översätter mer beräkningsintensiva uppgifter. Slutsatsen om kodens prestanda kunde inte heller dras pga. den enkla koden som användes under J2ObjC utvärderingen.

Kommer metoden vara det bästa sättet att portera från Android till iOS?
Även om J2ObjC är förnärvarande i alfa- och beta stadiet så uppfyller verktyget kravet att översätta Java kod till Objective-C kod. J2ObjC är dock inget verktyg som kan översätta en hel applikation direkt som nämndes i avsnitt 3.10. Det är snarare ett verktyg som används i samverkan med utvecklaren under översättningsprocessen. Utvecklaren måste själv överväga om den översatta koden är tillräckligt bra att användas. När det gäller mobiltelefoner är kodens prestanda extra viktigt att tänka på. Eftersom mobiltelefoner har lite minne [20] så är det viktigt att koden som översattes från J2ObjC inte är för minneskrävande.
Eftersom J2ObjC var det enda verktyget som hittades och som uppfyllde både målet med examensarbetet och kravet på prototypen, så anses den vara det bästa verktyget som finns just nu. Även om inga slutsatser kunde dras angående översättningen av mer beräkningsintensiva uppgifter, så visar resultatet i avsnitt 3.8 att J2ObjC ändå är ett effektivt verktyg för att översätta enkla Java koder.

Från erfarenheten med J2ObjC så är det ett rekommenderat verktyg för Saab att använda i kombination med traditionell plattformspecifik utveckling, eftersom verktyget uppfyller deras mål att hitta ett effektivare sätt att portera en Android applikation till iOS.

5.2 Framtida utredningar

Detta examensarbete har undersökt hur man portera en Android applikation, skriven i naturligt Java kod, till en iOS applikation som skall använda sig av naturligt Objective-C kod på ett effektivt sätt. J2ObjC var det enda verktyg som hittades och som kunde åstadkomma dessa mål. Detta verktyg är därför, kombinerad med traditionell plattform-specifik utveckling, den mest passande metod att använda just nu för att portera en Android applikation till iOS.

Det finns dock ett behov att undersöka hur väl J2ObjC översätter mer beräkningsintensiva uppgifter, eftersom detta problem aldrig blev berört. Hur koden påverka minnesanvändandet och prestandan i en iOS applikation är viktigt för att kunna fastställa hur väl J2ObjC översätter koder. Detta lämnas till framtida forskning.

Under examensarbetet användes J2ObjC tillsammans med utvecklingsmiljön Eclipse. Användbarheten när J2ObjC användes med Eclipse var väldigt tillfredsställande. Men hur väl J2ObjC utför sina uppgifter med XCode kommer att lämnas till framtida forskning.
Eftersom J2ObjC är fortfarande under utveckling måste en ny utredning göras för att kunna hitta alla fördelar respektive nackdelar när J2ObjC utvecklarna släpper full versionen. Detta lämnas också till framtida forskning.

Det sista som också lämnas till framtida utredning är att ta reda på hur man integrera J2ObjC i en porteringsprocess för att kunna få ut så mycket av verktyget som möjligt, dvs. att försöka använda sig av J2ObjC på alla möjliga ställen. En undersökning om hur väl J2ObjC översätter olika sorters uppgifter krävs för att kunna göra det. Det kommer att spara mycket tid om J2ObjC kan användas till vissa delar i porteringsprocessen.

6 Terminologi

Applikation	Ett program utvecklat för mobila enheter
Plattform	Operativsystem
Portering	Att göra ett program tillgängligt för en annan plattform
Open-source	Källkoden för programmet är tillgängligt för andra. En modifiering och vida-redistribuering av programmet är tillåten.
Ramverk	Ett utvecklat bibliotek med funktioner som ger möjligheten att utveckla för andra plattformar
System	En rad sammanhängande program i relation med varandra
Android	Ett operativsystem för mobila enheter såsom ”Smartphones” och surfplattor
iOS	Ett operativsystem för Apples mobila enheter såsom ”Smartphones” och surf-plattor
iOS SDK	Ett mjukvaruutvecklingsverktyg för Apples plattformar
Android API	Ett bibliotek som innehåller funktioner för att bygga upp användargränssnittet för Androids mobila enheter
User-Story	En variant av kravskrivning i form av korta berättelser som beskriver vad en användare vill uppnå med kravet

7 Referenser

[1] Almog, S. (2012). *What Is Codename One*
http://codenameone.blogspot.se/2012/01/what-is-codename-one.html [2014-06-04]

[2] Apple Inc. (2012). *Programming with Objective-C*
https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html [2014-03-03]

[3] Apple Inc. (2014). *Designing for iOS 7*
https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/ [2014-06-04]

[4] Ball, T. (2013). *Java Conversions*
https://code.google.com/p/j2objc/wiki/JavaConversions [2014-05-12]

[5] Ball, T. (2014). *Android and iPhones and Web, oh my.*
http://j2objc.blogspot.se/ [2014-05-13]

[6] Ball, T (2014). *Getting Started*
https://code.google.com/p/j2objc/wiki/HowToStart [2015-05-12]

[7] Begemann, O. (2014). *How Dropbox Uses C++ for Cross-Platform iOS and Android Development*
http://oleb.net/blog/2014/05/how-dropbox-uses-cplusplus-cross-platform-development/ [2014-05-25]

[8] Durga, A. (2013). *In Mobile Development, Run Everywhere can mean Debug Everywhere*
http://www.realstorygroup.com/Blog/2569-In-Mobile-Development-Run-Everywhere-can-mean-Debug-Everywhere [2014-03-30]

[9] Gambill, P. (2013). *Scrumban: A different way to be Agile*
http://www.deloittedigital.com/us/blog/scrumban-a-different-way-to-be-agile [2014-06-07]

[10] Holmqvist, A. (2013). *Fakta om informationssökning*
http://www.skolverket.se/skolutveckling/resurser-for-larande/kollakallan/kallkritik/fakta/fakta-om-informationssokning-1.152153 [2014-03-30]

[11] HyperlinkInfoSystem. (no date). *WE CAN DO MOBILE APP PORTING FOR YOU*
http://hyperlinkinfosystem.com/mobile-application-porting.htm [2014-03-24]

[12] iosindex. (no date). *Select Your iOS Device*
http://iosindex.com [2014-03-24]

[13] J2ObjC. (no date). *A Java to iOS Objective-C translation tool and runtime*
https://code.google.com/p/j2objc/ [2014-05-11]

[14] Kvalitativ metod. (no date). *Intervjuer*
http://kvalitativmetod.webs.com/intervjuer.htm [2014-03-24]

[15] Lauesen, S. (2002). *Software Requirements Styles and Techniques*, Edinburghh Gate, England: Pearson Education Limted

[16] Rinaldi, B. (no date). *Web designers become app developers with PhoneGap*
http://www.adobe.com/inspire/2013/11/web-designers-phonegap.html [2014-05-05]

[17] Skop. (no date). *KVALITATIVA METODER*
http://www.skop.se/metoder/kvalitativa-metoder/ [2014-06-08]

[18] Google. (no date). *Googles målsättning är att organisera världens information och göra den tillgänglig och användbar för alla.*
http://www.google.com/about/company/ [2014-06-09]

[19] Viswanathan, P. (2011). *Android OS Vs. Apple iOS - Which is Better for Developers?*
http://mobiledevices.about.com/od/kindattentiondevelopers/tp/Android-Os-Vs-Apple-Ios-Which-Is-Better-For-Developers.htm [2014-05-29]

[20] Assoc. (2012). *Objective-C versus Java*
http://assoc.tumblr.com/post/28261068461/objective-c-versus-java [2014-06-09]

[21] Saab Training and Simulation. (no date). *AKKA DISTRIBUTED OBSERVATION AND ANALYSIS IN REAL TIME*. [ONLINE].
http://www.saabgroup.com/Global/Documents%20and%20Images/Civil%20Security/Police%20and%20Rescue%20Solutions/Training%20and%20Simulation/flyer%20AKKA%20ver%20A%20web.pdf [2014-06-21]

[22] Hanna. (2011). *Vad är en smartphone egentligen?*
http://blogg.telia.se/smartarevardag/2011/09/28/vad-ar-en-smartphone-egentligen/
[2014-06-22]

[23] Sjöberg, O. (2013) *How fast is the mobile revolution changing the world? Ask the Smartphone Generation*
http://www.inma.org/blogs/mobile-tablets/post.cfm/how-fast-is-the-mobile-revolution-changing-the-world-ask-the-smartphone-generation [2014-06-22]

Appendix A

Kod från User-Stories

Det här appendixet innehåller hur Java koden från de olika ”user-stories” ser ut innan översättningen samt efter J2ObjC översättningen. Den innehåller också hur koden ser ut efter modifikationen av den översatta Java koden som kan kompileras i XCode.

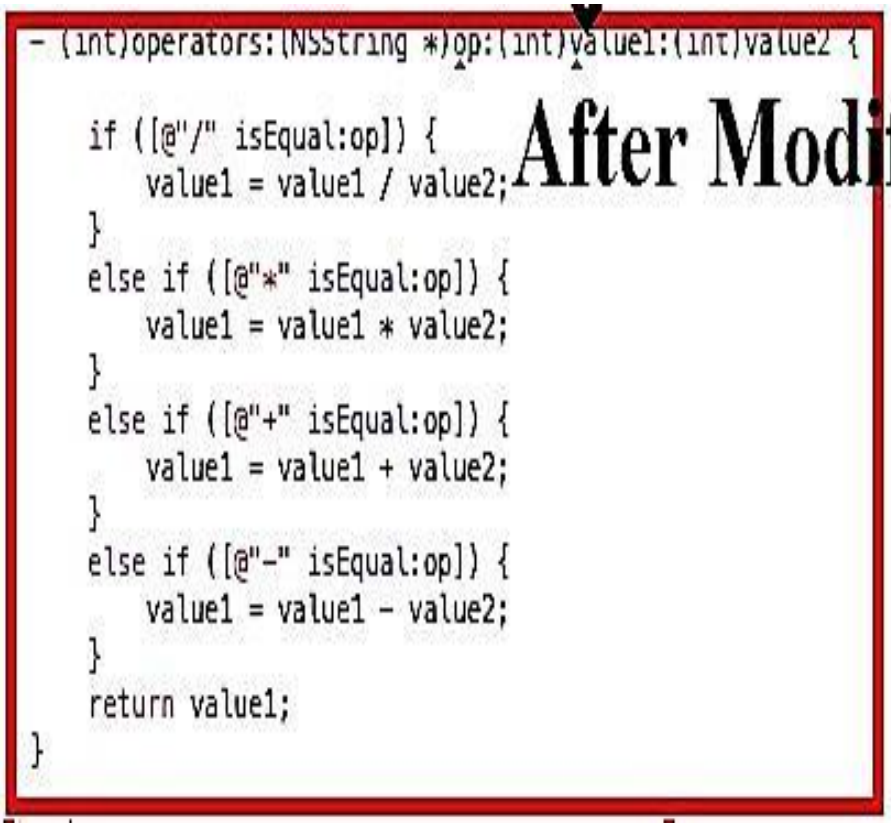
A.1 User-Story 1.2.4

```
    public int operators(String op,int value1,int value2){
        if(op=="/"){
            value1=value1/value2;
        }else if(op=="*"){
            value1=value1*value2;
        }else if(op=="+"){
            value1=value1+value2;
        }else if(op=="-"){
            value1=value1-value2;
        }
        return value1;
    }
}
```

Before Translation

```
#line 9
- (int)operatorsWithNSString:(NSString *)op
                     withInt:(int)value1
                     withInt:(int)value2 {
#line 10
  if ([@"/" isEqual:op]) {
#line 11
    value1 = value1 / value2;
  }
  else if ([@"*" isEqual:op]) {
#line 13
    value1 = value1 * value2;
  }
  else if ([@"+" isEqual:op]) {
#line 15
    value1 = value1 + value2;
  }
  else if ([@"-" isEqual:op]) {
#line 17
    value1 = value1 - value2;
  }
#line 19
  return value1;
}
```

After Translation


- (int)operators:(NSString *)op:(int)value1:(int)value2 {
if ([@"/" isEqual:op]) {
value1 = value1 / value2;
}
else if ([@"*" isEqual:op]) {
value1 = value1 * value2;
}
else if ([@"+" isEqual:op]) {
value1 = value1 + value2;
}
else if ([@"-" isEqual:op]) {
value1 = value1 - value2;
}
return value1;
}
After Modification

A.2 User-Story 1.2.5

```
public int operators(String op, int value1, int value2){
    if(op=="MAX"){
        value1=Math.max(value1, value2);
    }else if(op=="MIN"){
        value1=Math.min(value1, value2);
    }else if(op=="POW"){
        value1=(int) Math.pow(value1, value2);
    }else if(op=="SQUARE ROOT"){
        value1=(int) Math.sqrt(value1+value2);
    }
    return value1;
}
```

Before Translation

```
#line 9
- (int)operatorsWithNSString:(NSString *)op
                    withInt:(int)value1
                    withInt:(int)value2 {
#line 10
  if ([@"MAX" isEqual:op]) {
#line 11
    value1 = [JavaLangMath maxWithInt:value1 withInt:value2];
  }
  else if ([@"MIN" isEqual:op]) {
#line 13
    value1 = [JavaLangMath minWithInt:value1 withInt:value2];
  }
  else if ([@"POW" isEqual:op]) {
#line 15
    value1 = (int) [JavaLangMath powWithDouble:value1 withDouble:value2];
  }
  else if ([@"SQUARE ROOT" isEqual:op]) {
#line 17
    value1 = (int) [JavaLangMath sqrtWithDouble:value1 + value2];
  }
#line 19
  return value1;
}
```

After Translation

```
- (int)operators:(NSString *)op:(int)value1:(int)value2 {

    if ([@"MAX" isEqual:op]) {
        value1 = MAX(value1, value2);
    }
    else if ([@"MIN" isEqual:op]) {
        value1 = MIN(value1, value2);
    }
    else if ([@"POW" isEqual:op]) {
        value1 = (int) pow(value1, value2);
    }
    else if ([@"SQUARE ROOT" isEqual:op]) {
        value1 = (int) sqrt(value1+value2);
    }
    return value1;
}
```

After Modification

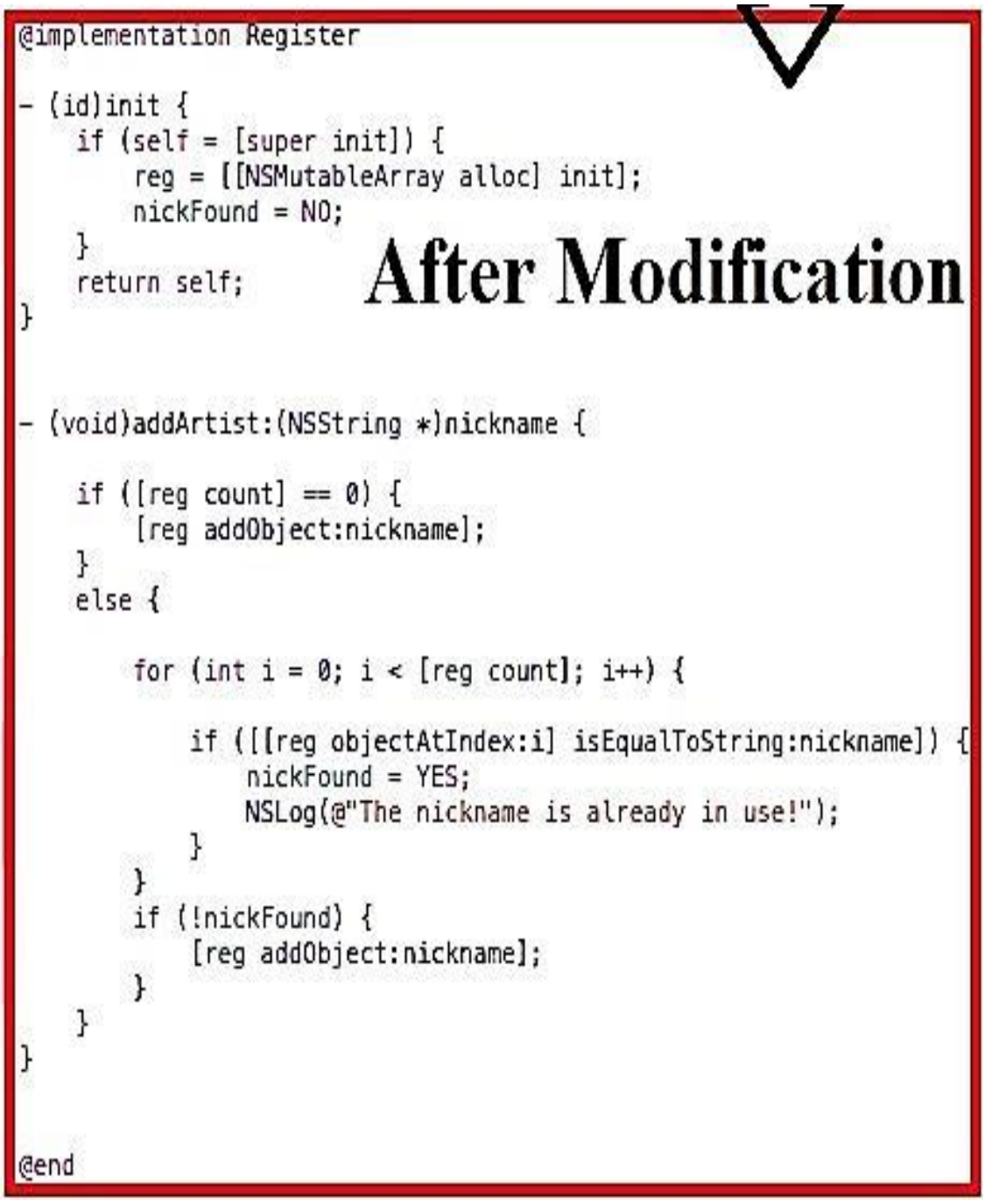
A.3 User-Story 1.2.6

Before Translation

```
import java.util.ArrayList;

public class Register {
    public ArrayList<String> reg;
    boolean nickFound;
    public Register (){
        reg = new ArrayList<String>();
        nickFound=false;
    }
    public void addArtist(String nickname){
        if(reg.size()==0){
            reg.add(nickname);
        }else{
            for(int i=0; i<reg.size(); i++){
                if(reg.get(i).compareTo(nickname)==0){
                    nickFound=true;
                    System.out.println("The nickname is already in use!");
                }
            }
            if(!nickFound){
                reg.add(nickname);
            }
        }
    }
}
```

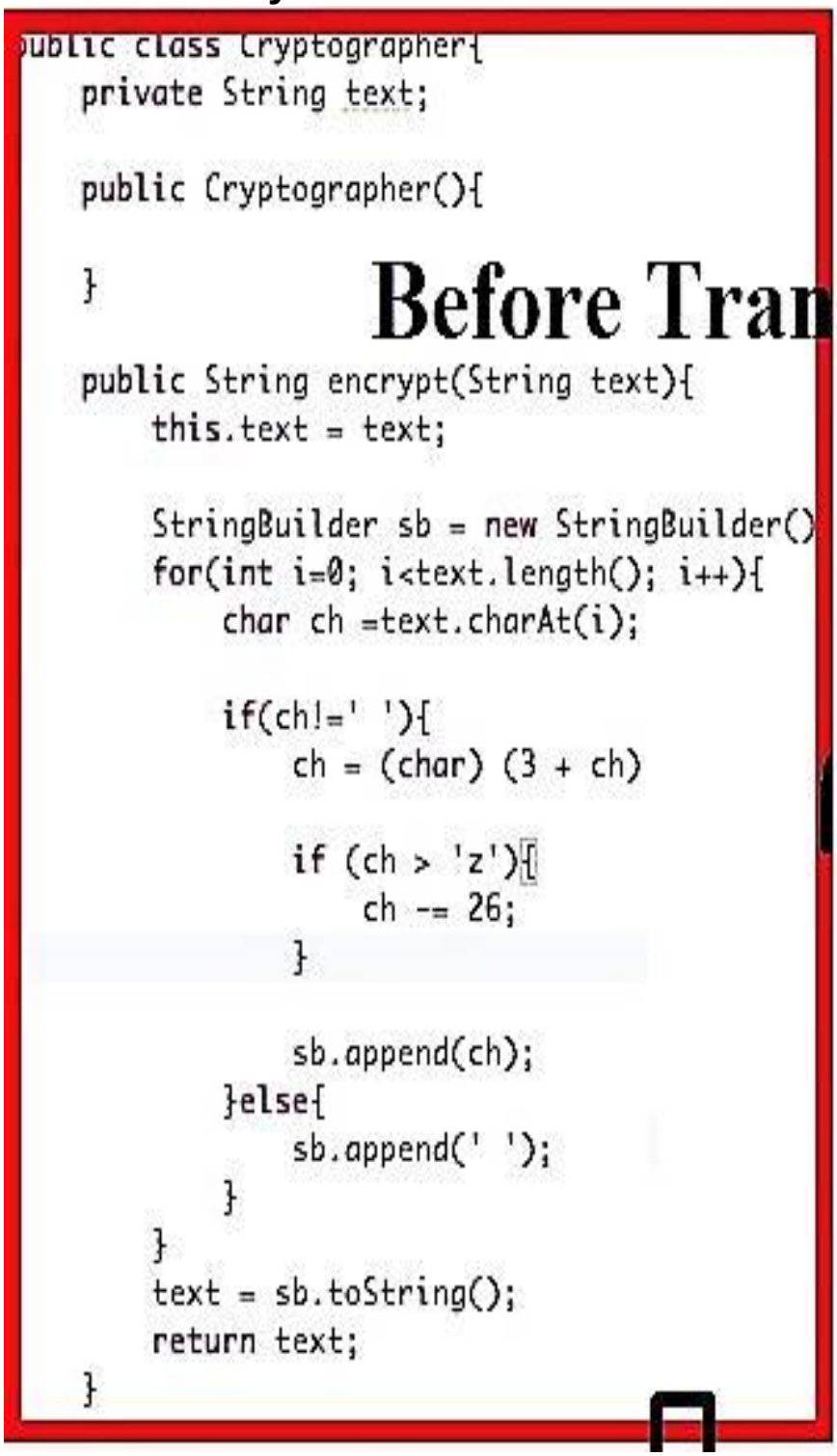
After Translation

```
#line 8
- (id)init {
  if (self = [super init]) {
    reg_ = [[JavaUtilArrayList alloc] init];
    nickFound_ = NO;
  }
  return self;
}

#line 12
- (void)addArtistWithNSString:(NSString *)nickname {
#line 13
  if ([((JavaUtilArrayList *) nil_chk(reg_)) size] == 0) {
#line 14
    [reg_ addWithId:nickname];
  }
  else {
#line 16
    for (int i = 0; i < [reg_ size]; i++) {
#line 17
      if ([((NSString *) nil_chk([reg_ getWithInt:i])) compareToWithId:nickname] == 0
          ) {
#line 18
        nickFound_ = YES;
#line 19
        [((JavaIoPrintStream *) nil_chk([JavaLangSystem out]))
            println1nWithNSString:@"The nickname is already in use!"];
      }
    }
#line 22
    if (!nickFound_) {
#line 23
      [reg_ addWithId:nickname];
    }
  }
}
```

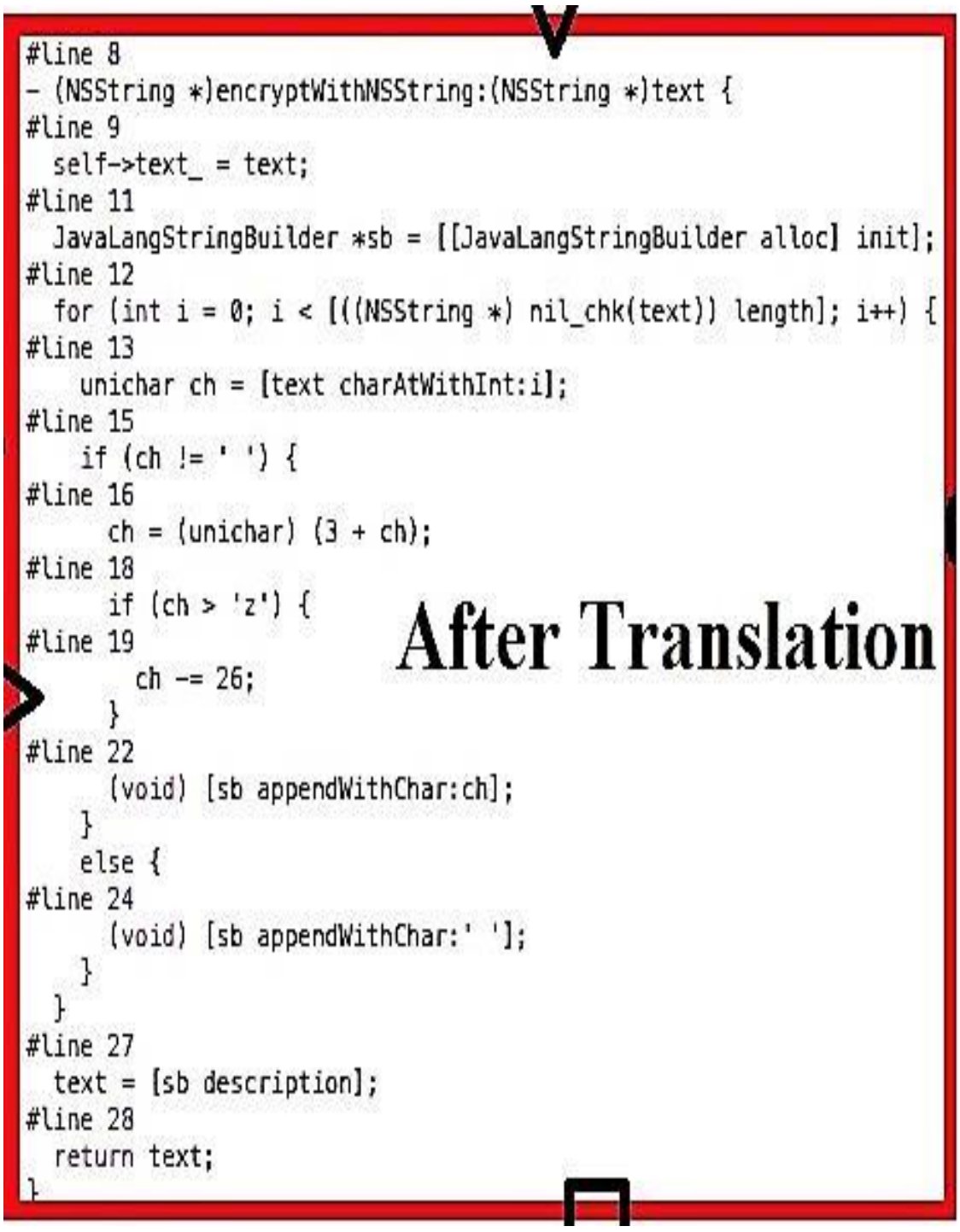


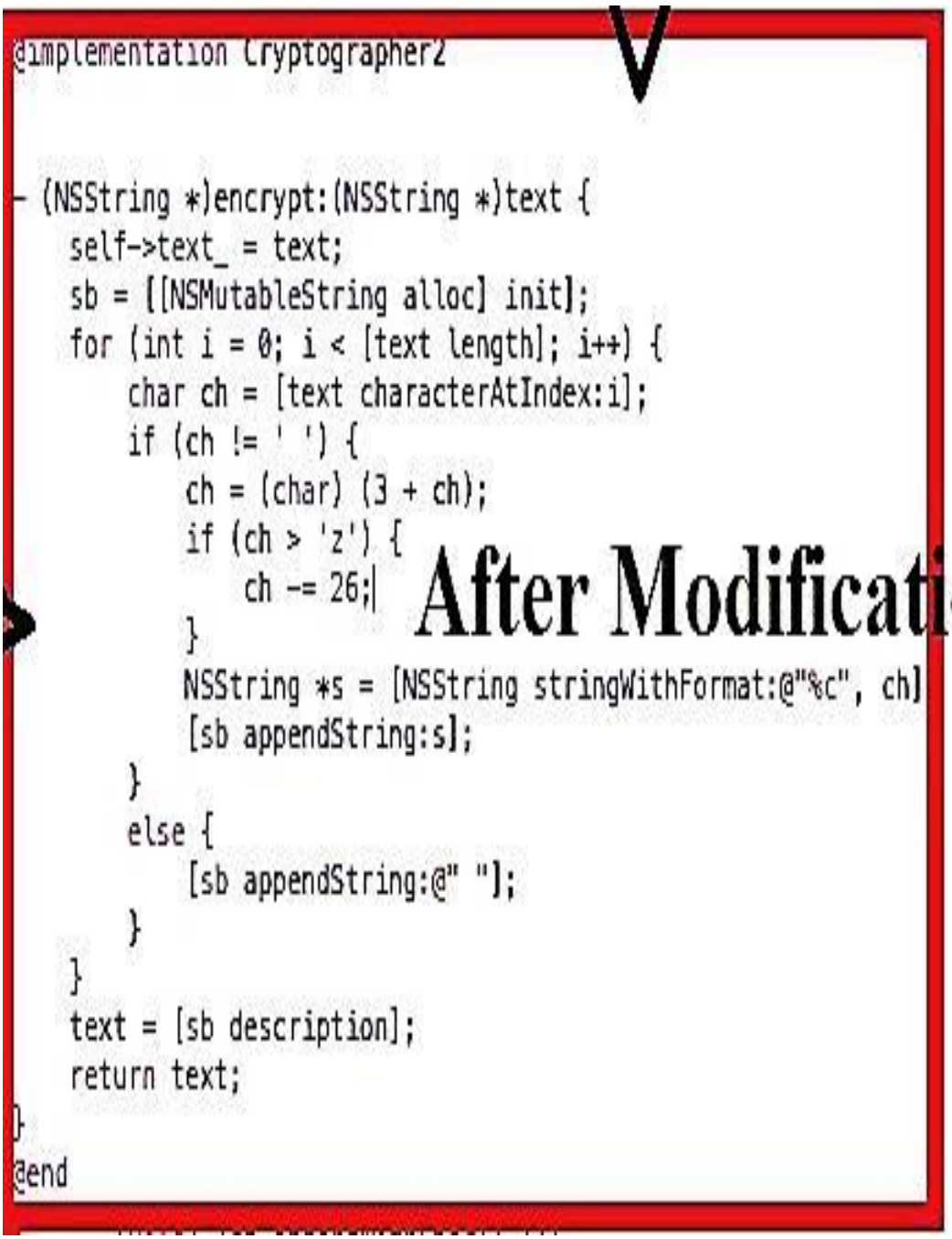
```
@implementation Register

- (id)init {
    if (self = [super init]) {
        reg = [[NSMutableArray alloc] init];
        nickFound = NO;
    }
    return self;
}


- (void)addArtist:(NSString *)nickname {

    if ([reg count] == 0) {
        [reg addObject:nickname];
    }
    else {

        for (int i = 0; i < [reg count]; i++) {

            if ([[reg objectAtIndex:i] isEqualToString:nickname]) {
                nickFound = YES;
                NSLog(@"The nickname is already in use!");
            }
        }
        if (!nickFound) {
            [reg addObject:nickname];
        }
    }
}


@end
```

A.4 User-Story 1.2.7




```
ublic class Cryptographer{
    private String text;

    public Cryptographer(){

    }

    public String encrypt(String text){
        this.text = text;

        StringBuilder sb = new StringBuilder()
        for(int i=0; i<text.length(); i++){
            char ch =text.charAt(i);

            if(ch!=' '){
                ch = (char) (3 + ch)

                if (ch > 'z'){
                    ch -= 26;
                }

                sb.append(ch);
            }else{
                sb.append(' ');
            }
        }
        text = sb.toString();
        return text;
    }
```


#line 8
- (NSString *)encryptWithNSString:(NSString *)text {
#line 9
self->text_ = text;
#line 11
JavaLangStringBuilder *sb = [[JavaLangStringBuilder alloc] init];
#line 12
for (int i = 0; i < [((NSString *) nil_chk(text)) length]; i++) {
#line 13
unichar ch = [text charAtWithInt:i];
#line 15
if (ch != ' ') {
#line 16
ch = (unichar) (3 + ch);
#line 18
if (ch > 'z') {
#line 19
ch -= 26;
}
#line 22
(void) [sb appendWithChar:ch];
}
else {
#line 24
(void) [sb appendWithChar:' '];
}
}
#line 27
text = [sb description];
#line 28
return text;
}
After Translation


@implementation Cryptographer2
- (NSString *)encrypt:(NSString *)text {
self->text_ = text;
sb = [[NSMutableString alloc] init];
for (int i = 0; i < [text length]; i++) {
char ch = [text characterAtIndex:i];
if (ch != ' ') {
ch = (char) (3 + ch);
if (ch > 'z') {
ch -= 26;
}
NSString *s = [NSString stringWithFormat:@"%c", ch]
[sb appendString:s];
}
else {
[sb appendString:@" "];
}
}
text = [sb description];
return text;
}
@end
After Modification

Appendix B
Tidsplan

B.1 Tidsplan


v.09
v.10
v.11
v.12
v.13
v.14
v.15
v.16
v.17
v.18
v.19
v.20
v.21
v.22
v.23
v.24
v.25
Planering
Informationsinsamling
Slutrapport
Analys/Kravinsamling
Kravbehandling
Tenta Databas Mån
Tenta Säkerhet Fre
Tenta Krav Tis
Inläming av Objective-C
Sprint 1
Utveckling av prototyp
Regressionstest
Funktionstest
Opposition
Användartest
Redovisning Campus
Redovisning Saab