Introduksjon og installasjon

Tomas Holt, Institutt for informatikk og e-læring, NTNU
Lærestoffet er utviklet for faget IFUD1042 Applikasjonsutvikling for mobile enheter

Resymé: Denne leksjonen er en introduksjon til programmering av mobile enheter. Først kikker vi på hvilke relevante plattformer som finnes for avanserte mobile enheter og så går vi over på å installere nødvendige verktøy for å lage applikasjoner på Android-plattformen.

Innhold

1.1.Hva er en avansert mobil enhet?

Når vi snakker om avanserte mobile enheter så tenker vi i første rekke på enheter som kan holdes i hånda [Def Handheld]. Vi snakker gjerne om smarttelefoner, PDA'er og nå også nettbrett.

Vanlige tradisjonelle mobiltelefoner holdes her utenfor (men er åpenbart en håndholdt enhet). Disse er mindre avanserte, har mindre funksjonalitet og prosesseringskraft. Disse enhetene dekker i første rekke muligheten for å ringe og sende meldinger, mens de avanserte mobile enhetene vi snakker om er mye nærmere tradisjonelle PC'er, med unntak av at de er en god del mindre. De har gjerne nettleser, epostprogram, mulighet for å vise video osv.

1.2.Introduksjon

Apple lanserte smarttelefonen iPhone i juni 2007. Enheten ble fort populær som en følge av at man kunne bruke enheten som vanlige mobiltelefoner, men også hadde muligheter som tidligere bare var forbeholdt PDA'er og større enheter.

Programmering på mobile enheter har skutt fart etter at lanseringen av iPhone. Apple åpnet for utvikling og distribusjon av applikasjoner via en egen distribusjonskanal. Utviklerverktøy gjorde det mulig å enkelt utvikle mobilapplikasjoner for telefonen. Populariteten til iPhone, den enkle distribusjonen og muligheten for å selge programvare, har etterhvert tiltrukket seg mange utviklere.

Android er et Linux-basert system (open source) for avanserte mobile enheter. Firmaet som startet utviklingen var Android Inc som ble kjøpt opp av Google i 2005. Google har siden vært Android sin største pådriver men utviklingen gjøres gjennom Open Handset Alliance [Open Handset], en gruppe selskaper som samarbeider for utvikling av åpne standarder for mobile enheter. I november 2007 ble Android avdekket for omverdenen. Systemet var i utgangspunktet tiltenkt smarttelefoner, men man ser nå en tendens til at det blir tatt i bruk på alt fra mobiler, nettbrett (tablett maskin), PC'er og TV'er.

Også for Android finnes det en egen distribusjonskanal (Android Marked) for applikasjoner. Denne kanalen tilhører ikke Open Handset Alliance, men Google. Marked lar brukere laste ned applikasjoner på samme måte som Apples App Store. Utviklerverktøyene til Android lar oss utvikle applikasjoner i Java (hvor JVM og API har blitt tilpasset mobile enheter). Selve kjernen til Java er bevart, men det er også en del endringer. Bibliotekene for å lage grafiske grensesnitt er fjernet og nye (bedre tilpassede) lagt til. I tillegg kjøres ikke programmer i Android helt på samme måte som på PC. Utvikling for Android krever derfor både å lære seg nye biblioteker og et nytt rammeverk for utviklingen. Rammeverket setter større krav til å følge konvensjoner enn man er vant til fra Java SE (standard edition).

1.3.Plattformer for avanserte mobile enheter

Det finnes i dag flere alternative plattformer eller operativsystemer (OS) for avanserte mobile enheter. Operativsystemet er en viktige bestanddel i datasystemer da det er denne programvaren som kobler våre programmer mot maskinvaren. Operativsystemet sørger for å gi programmene tilgang til nødvendige ressurser (prosessor, minne, filer, sensorer osv.) og ikke minst å administrere de ulike programmene. Dette innebærer blant annet å starte og stoppe programmer. Operativsystemet kan også prioritere mellom ulike programmer (mer

eksakt prosesser) basert på «viktigheten av» de ulike oppgavene. Åpenbart vil valg av operativsystem kunne ha mye å si om hva slags funksjonalitet som kan tilbys programmererne og ikke minst hvilken maskinvare som kan brukes. For mer informasjon om operativsystemer se [OS].

De mest aktuelle operativsystemene for avanserte mobile enheter er:

- Android
- iOS (brukes på iPhone og iPad)
- Symbian
- Windows i flere utgaver, men det er vel i praksis bare Windows Phone 7 og 8 som er særlig aktuell i dag (for nyere enheter)

La oss se litt nærmere på de forskjellige alternativene.

1.3.1. Android

Open Handset Alliance er gruppen av bedrifter som styrer utviklingen av Android. Mange store firma er involverte [Members]. Android er et operativsystem som er basert på Linux-kjernen. Systemet er åpen kildekode (open source) og tilgjengelig under Apache v2 lisens. Dette vil i praksis si at alle har tilgang til kildekoden og kan gjøre de endringene man vil. Man trenger imidlertid ikke å publisere de endringene som gjøres og det er fullt mulig å bruke systemet i kommersiell sammenheng uten å betale noen form for avgift. Dette gjør systemet attraktivt og det er i dag mange forskjellige leverandører som bruker operativsystemet på sine smarttelefoner.

Android startet som operativsystem for smarttelefoner, men tilbys i større og større grad også for nettbrett. Nettbrettene er på mange måter like smarttelefonene, men med større skjerm. Det varierer også om man har mulighet til å ringe med dem. Tendensen er også at en del av nettbrettene blir mer en krysning mellom bærbar PC og nettbrett, se [Nettbrett PC].

I tillegg til dette tilpasses Android også til x86 prosessorene [x86org], noe som gjør at man også kan bruke Android på PC'er. Dette er spesielt gunstig for PC'er med beskjeden maskinvare, da responsen i et lett system som Android er mye bedre enn i dagens «normale OS». Toshiba er en av produsentene som tilbyr nett-PC med Android [Toshiba] (riktignok ikke med x86 prosessor).

Utvikling på Android-plattformen skjer hovedsaklig gjennom Java, men man kan imidlertid også bruke nativ kode (altså C/C++) om ønskelig. Dette gjøres da ved at man kaller denne koden fra Java-koden (via Java Native Interface). Fordelen med dette er at man stort sett kan lage applikasjonene i Java og får dra nytten av fordelene med dette språket, mens man kan ty til C/C++ der hvor det er gode grunner for det. Hovedårsaker til å velge C/C++ er ytelse eller at man allerede har applikasjoner skrevet i et av disse språkene. Spesielt innenfor spill er dette vanlig, noe som kan gjøre det enklere å få spill over fra andre plattformer til Android.

For grafikkakselerering brukes EGL som er et grensesnitt mellom vindussystemet og rendering API'en (f.eks. OpenGL ES). Dette er en åpen standard. Det at man kan bruke OpenGL ES kan være en fordel da denne API'en allerede er utbredt og kan også gjøre det enklere å oversette programmer allerede skrevet i OpenGL (som er en mye brukt grafikk API).

Applikasjoner som lages kan gjøres tilgjengelig for sluttbrukere gjennom Google Play (tidligere Android Marked). Applikasjonen kan distribueres gratis om man vil det, eller man kan ta et ønsket beløp (det er en betalingstjeneste knyttet til markedet). I sistnevnte tilfelle må man betale en avgift (i skrivende stund 30% av salgssummen).

1.3.2. iOS

iOS brukes av Apple som operativsystem i iPad Touch, iPhone, iPad og Apple TV. Systemet er lukket (i motsetning til Android) og utvikles kun av Apple. Systemet selges kun på Apple sine egen enheter.

Utvikling skjer i Objective-C (noe som også gir mulighet for C/C++). Objective-C gir mulighet for «garbage collection» (som er en av fordelene med Java i forhold til C/C++). En ulempe med Objective-C er at ikke er så mange utviklere som er vant til dette språket i forhold til C/C++.

Akkurat som for Android brukes EGL for å akselerere grafikk.

Applikasjoner kan gjøres tilgjengelig for sluttbrukere gjennom «App Store», og også her må selger betale 30% avgift på salgsummen på alle applikasjoner. I tillegg så er det en avgift for i det hele tatt kunne legge ut applikasjoner på «App Store». Denne er i dag minst $99 pr år. Forøvrig må alle applikasjoner godkjennes av Apple før de kan distribueres.

For mer informasjon om utvikling på denne plattformen se [Dev iOS] .

1.3.3. Symbian

Dette er et system som opprinnelig ble laget av Symbian Ltd og introdusert allerede i 1999. Målgruppen var PDA'er og mobiltelefoner. Både Sony Ericsson og Nokia har brukt dette systemet på sine enheter.

Det virker som om systemet har mistet markedsandeler og ikke lenger er like aktuelt. Symbian Foundation har konverterert systemet og det har nå bli fritt å bruke uten lisenspenger og er åpen kildekode. Organisasjonen har ingen ansatte.

Nokia har vært den store leverandøren bak dette systemet, men valgte så å bruke Microsoft sitt OS. Det var nok ikke noe godt valg som du snart skal se.

1.3.4. Windows Phone

Microsoft har hatt flere operativsystemer (blant annet Windows CE og Windows Mobile) for avanserte mobile enheter. De innså imidlertid at de måtte komme med noe nytt som svar på iOS og Android, da Windows Mobile begynte å bli temmelig akterutseilt. Windows Phone 7 ble derfor utviklet fra bunnen av og lansering skjedde i slutten av oktober 2010.

Flere store produsenter har varslet at de vil benytte systemet på sine enheter (gjerne i tillegg til Android). Vi kan si i dag at Windows 7 Mobile ikke har erobret noen stor andel av markedet.

I juni i 2012 kom Windows Phone 8 (mens 8.1 kom april 2014). Flere leverandører har også støttet denne, blandt annet Nokia som har gått helt bort fra Symbian, men heller ikke denne versjonen av Windows tok særlige markedsandeler. All støtte til disse versjonene er nå fjernet. Microsoft kom riktignok også med Windows 10 Mobile, men utviklingen av denne versjonen er også stoppet - det vil riktignok gis sikkerhetsoppdateringer ut 2019.

1.3.5. Hvorfor Android?

Nå har vi kikket på ulike operativsystemer for avanserte mobile enheter. Et spørsmål som man må stille seg som utvikler er hvilken plattform man tror det er best å bruke. I mange sammenhenger kan det nok være aktuelt å oversette applikasjoner til flere enn en av plattformene, men som det kommer fram av sammenlikningen så er ikke dette rett fram. API'ene som brukes er ulike (stort sett), samt at det brukes ulike programmeringsspråk.

Vi har i dette faget valgt å bruke Android som plattform. Det har vært diskutert om vi kanskje burde dekke flere plattformer i faget, men vi har funnet ut at dette nok ikke er hensiktsmessig utover omtalen de allerede har fått.

Symbian er som omtalt et system som er i en overgangsfase. Systemet er gjort om til åpen kildekode, men det virker som om systemet sliter mot konkurrentene (i hvertfall i sin nåværende form). Derfor anses Symbian som mindre aktuell for et slikt fag.

Windows Phone/Mobile er av åpenbare grunner ikke lenger aktuelle.

Hva så med iOS kontra Android? Dette er begge systemer som har vært en tid på markedet. Android er åpent kontra iOS som er lukket. I tillegg så er det snakk om lisenspenger ved bruk av iOS. Ellers er Android et system som kan brukes fra alt til smarttelefoner til PC'er, noe som gir et stort potensielt bruksområde. Det at systemet er åpent, fritt og basert på Linux kan potensielt gi tilgang til funksjonalitet fra dette miljøet (som f.eks. drivere til eksterne enheter osv.), samt at man ikke er like sårbar for kromspring fra ett enkelt firma (selv om man her også skal være klar over at det er Google som står for det aller meste av utviklingen).

Android programmeres hovedsaklig i Java. Ettersom Java virker å være mest brukt i opplæring så er dette en klar fordel kontra Objective-C.

Android har i det store og hele mange forutsetninger for å være en dominerende aktør i OS-markedet i årene framover. Det er åpent, fritt, fleksibelt og har en solide pådriver.

1.4.Introduksjon til faget og leksjonene

Det vil i dette faget gis ut et antall leksjoner samt øvinger tilknyttet til hver leksjon. Leksjonene er skrevet for å gi oss faglærere muligheten til å framheve ting som er spesielt viktig eller å klargjøre tema som ikke er bra nok dekket av læreboka. Vi vil i hver leksjon spesifisere hva som skal leses i læreboka. I en del tilfeller vil det være naturlig å også referere til stoff fra andre plasser.

For deg som student vil leksjonene og øvingene definere pensum. I tillegg kan det selvsagt være aktuelt for deg å finne informasjon utover dette på egen hånd (kanskje spesielt i forhold til prosjektøvingen som skal utføres). Her kan andre bøker være aktuelle, men det er i mange tilfeller mest nærliggende å bruke web til slike ting. Det har etter hvert blitt mye informasjon tilgjengelig på nett for Android-utviklere.

I leksjonene vil det være en del referanser merket slik [Referanser]. Du vil kunne finne hva denne referansen betyr under kapitlet «Referanser» i slutten av leksjonene. Disse referansene kan f.eks. være relatert til nettlenker, bøker, kode osv.

1.5.Læreboka

Vi har valgt læreboka [Lærebok] etter en sammenlikning med andre aktuelle bøker for Android 5 på våren 2015. Fordelen med boka er at den er forholdsvis kortfattet. Det har riktignok kommet flere versjoner etter Android 5 (det kommer ca. en ny pr. år), men mye er fortsatt relevant. Ikke minst har vi tilgang til god dokumentasjon på nett (se under).

Det er viktig at det ikke blir for mye detaljer når man skal lære temaet. Ettersom faget også skal kunne tas av folk uten mye Java-kompetanse er det viktig at læreterskelen ikke blir for høy. Dette gjør imidlertid at det kan være nødvendig/nyttig å hente informasjon fra andre plasser.

1.6.Andre ressurser

Du bør gjøre deg kjent med ressursene her. Dette vil gjøre det enklere når du trenger mer informasjon enn det som gis gjennom lærematerialet i faget.

Spesielt vil [Android Developer] være en fin plass å finne informasjon. Her kan du finne informasjon, eksempler og ikke minst API-dokumentasjon [API Dok]. **Denne er viktig for å finne ut hvilke metoder og funksjonalitet de ulike klassene kan tilby!**

[Anddev] er et forum for Android-utvikling. Her vil det være mulig å finne svar på mange problemer, det er mulig å stille spørsmål til likesinnede og det finnes veiledninger (tutorials) på hvordan man kan løse ulike oppgaver.

[Developer] er et generelt nettsted for utviklere.

1.7.Om leksjonen

I forbindelse med denne leksjonen forventes det at følgende kapittel i læreboka leses

- Introduction
- Kapittel 1: Getting Started
- De to første underkapitlene (The Activity Lifecycle og ActivityDemo Example) i kapittel 2: Activities.
- Appendix A: Installing the JDK* (leses om du har behov for det)

Introduksjonskapitlet gir en grei innledning i forhold til utvikling for Android. Kapittel 1 viser hvordan man installerer nødvendige verktøy og lager en veldig enkel applikasjon som kjøres i Android-emulator. **Merk!** at det er nødvendig å installere Java JDK (også kalt SDK) før man går igjennom kapittel 1. Dette er beskrevet i Appendix A. Om du ikke allerede har installert Java JDK så gjør dette før du starter med kapittel 1.

1.8.Installasjon

Installasjonsbeskrivelsen i boka er etter mitt skjønn grei. Jeg opplevde imidlertid at etter installasjon av Android Studio (og Android SDK) så fungerte ikke eksempelapplikasjonen beskrevet i kapittel 1. Grunnen til dette var at Java JDK ikke automatisk ble funnet. Løsningen på dette er å gjøre følgende i Android Studio. Velg File -> Project Structure ->

SDK Location -> JDK Location. I dette feltet velger du hvor Java JDK er installert. Rebuild av prosjektet fungerer da fint.

Om du får problemer under installasjon/bruk Android Studio, still spørsmål på fagets diskusjonsforum.

Merk: Emulatoren trenger lang tid på å starte (kan ta flere minutter på treg maskin)! Det kan virke som om systemet har hengt seg når du starter kjøring av koden første gang (og emulatoren må starte). Vær tålmodig, lastingen tar lang tid ved oppstart, men dette gjelder kun ved oppstart. Nye kjøringer vil bruke den samme emulatoren (og vi slipper dermed å laste den på nytt).

1.9.Android-applikasjoner

I første underkapittel (The Activity Lifecycle) av kapittel 2: er det forklart litt dypere hvordan en aktivitet («hovedklassen» for en Android-applikasjon) er bygd opp. På nåværende tidspunkt kan nok en del av dette virke ullent, men det er ingen grunn til å fortvile, dette er noe vi skal jobbe med utover i faget.

Du kan nå gjerne lese det som står i boka og så komme tilbake til det som kommer under etterpå.

1.9.1. Aktiviteter

Som det kommer fram i boka så er alle Android-applikasjoner basert på minst en aktivitet. Det vil si at man bestandig har en klasse som arver (extends) Activity-klassen. Du kan selv se dette i bokas eksempler. Aktiviteten blir startet av Android-operativsystemet, og det vil kalles spesifikke livsyklus-metoder på aktivitetsobjektet, se fig. 2.1 i boka.

Du kan f.eks. se at den første metoden som vil kalles er onCreate(). Du vil også se at du har implementert denne metoden i din egen kode. Hva så med de andre metodene som er beskrevet (onStart(), onResume(), onPause() osv.)? Disse metodene har du kanskje ikke i din egen aktivitetsklasse? Saken er at når du arver fra Activity eller en sub-klasse av denne så arver du også alle metodene i denne klassen. Metodene «finnes» dermed også i din klasse, selv uten at du lager dem! Du bør nå selv sjekke ut [API dok] for Activity.

I de tilfellene hvor vi ønsker å legge til vårt eget innhold i livsyklusmetodene gjør vi dette gjennom å lage metoden på nytt i klassen vår (dette gjør at denne metoden brukes og ikke den som ligger i Activity-klassen). Det er dette du gjorde når du implementerte onCreate()-metoden.

La oss ta et lite eksempel for å avmystifisere det hele litt. Man kan få inntrykk av at programmering for Android er veldig ulikt det man er vant til med fra før (med alle ressursene og rammeverket som brukes). Det er selvsagt grunner til at man har laget rammeverket slik man har gjort for Android, men det er fortsatt mulig å gjøre det meste i kode (selv om det ikke anbefales). La oss se på hvordan man kan lage en enkel meny i kode (og ikke via xml-filer som anbefales).

Ved å kikke i API-dokumentasjonen [API Dok] for Activity-klassen så vil du se at det finnes en metode som heter *onCreateOptionsMenu().* Ved å lese det som står i API-dokumentasjonen for denne metoden finner vi ut at det ikke er så mye som skal til for å legge til en meny til aktiviteten vår. Hver aktivitet lager faktisk en meny selv, og metoden *onCreateOptionsMenu()*

kalles automatisk slik at vi kan legge til egne menyinnslag om vi ønsker det. Prøv selv å legge til følgende kode i aktivitetsklassen din (husk også å importere klassene Log, Menu og MenuItem).

```
public boolean onCreateOptionsMenu(Menu meny){

        super.onCreateOptionsMenu(meny);//kaller metoden som vi arver, er dog ikke nødvendig

        meny.add("Valg 1"); //legger til meny-valg med teksten «Valg 1»

        meny.add("Valg 2");

        Log.i("onCreateOptionsMenu()","meny laget"); //skriver ut til logg, vises i LogCat

        return true; //true her gjør at menyen vil vises

}
```

Når du har lagt til koden over og kjører applikasjonen på nytt vil du med å trykke på Menu-knappen (hvor den er varierer i ulike Android-versjoner – så prøv deg fram) se en meny med to valg («Valg 1» og «Valg 2»). Det var vel ikke så vanskelig :-)

Ved å kikke litt mer i API-dokumentasjonen for *onCreateOptionsMenu()* står det at metoden *onOptionsItemSelected(MenuItem item)* håndterer menyvalg. Kikker vi litt nøyere på det som står (og kanskje også litt på MenuItem) kan vi konkludere med at følgende kode vil gi oss muligheten til å håndtere om brukeren velger menyen «Valg 1»:

```
public boolean onOptionsItemSelected(MenuItem item){

        if (item.getTitle().equals("Valg 1")){

                Log.i("onOptionsItemSelected()","Valg 1 er trykket av brukeren");

        }

        return true; //hvorfor true her? Se API-dokumentasjonen!!

}
```

Legg til koden og kjør på nytt. Se i LogCat-vinduet at du faktisk kan finne utskriften «Valg 1 er trykket av brukeren».

Jeg håper nå å ha avmystifisert Android-applikasjoner litt, og også å framhevet nødvendigheten/viktigheten av API-dokumentasjonen. Dette er en meget aktuell plass å finne nødvendig informasjon. Det er dog ikke alltid like enkelt å finne fram i denne dokumentasjonen (uten at man har en del kunnskap på plass), og den vil derfor være et supplement til boka og andre kilder som f.eks. [Android DevGuide] som innholder en mer forklarende tilnærming enn det API-dokumentasjonen gjør (mens denne går mer i detalj).

Ved å gå inn på [Android DevGuide Menus] kan du se en mer fyldig beskrivelse av hvordan bruke menyer. Vær imidlertid klar over at man der har valgt å skille mye ut i xml-filer i stedet for å kode direkte i Java. Dette har fordeler (mer om dette lenger ut i faget), men øker læreterskelen noe for oss som er vant med å kode direkte i Java.

2. Referanser

[Anddev] - http://anddev.org

[Android Developer] - http://developer.android.com/develop/index.html

[Android DevGuide] - http://developer.android.com/guide/index.html

[Android DevGuide Menus] – http://developer.android.com/guide/topics/ui/menus.html

[Android Install] - http://developer.android.com/sdk/index.html

[API Dok] - http://developer.android.com/reference/classes.html

[Bug Linux] - http://developer.android.com/guide/developing/device.html#setting-up

[Def Handheld] - http://en.wikipedia.org/wiki/Mobile_device

[Developer] - http://developer.com

[Dev iOS] - http://developer.apple.com/

[Geek] - http://www.geek.com/articles/mobile/microsoft-to-gamble-billions-on-windows-phone-7-20100827/

[Handholdt] - http://en.wikipedia.org/wiki/Mobile_device

[Lærebok] - Android Application Development: A Beginner's Tutorioal

[Members] - http://www.openhandsetalliance.com/oha_members.html

[Nettbrett PC] - http://www.dinside.no/859527/asus-lanserer-fire-nettbrett

[Open Handset] - http://www.openhandsetalliance.com/index.html

[OS] - http://en.wikipedia.org/wiki/Operating_system

[Toshiba] - http://www.digi.no/849215/interessant-android-pc-fra-toshiba

[x86org] - http://www.android-x86.org/