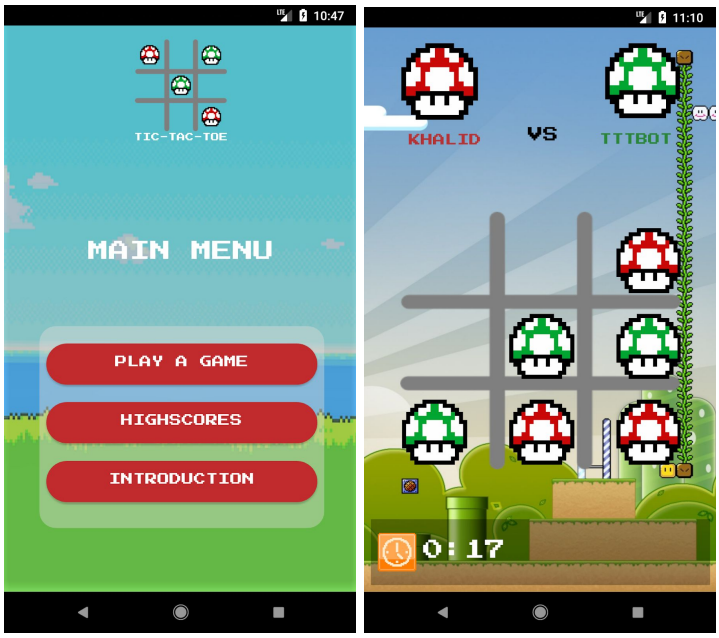
ANDROID PROGRAMMERING

PGR201, 2018 Eksamen

Dokumentasjon av TIC-TAC-TOE Applikasjon

Av

Khalid B. Said



Innholdsfortegnelse

- Innledning
- Start fasen i prosjektet
- Design
  - Designet for alle telefoner.
  - Applikasjonen’s design
  - Reise av Applikasjonen
- Kodefiler og Struktur
  - Beskrivelse av kodefiler
  - Class Diagram
  - Styrker og svakheter
  - Botlogikk
- Tester
  - Enhetstesting
  - Systemtesting
  - Brukertesting

Innledning

I denne dokumentasjonen vil jeg snakke om hvordan jeg har løst Android PGR201 eksamen. Gjennom denne dokumentasjonen så har jeg lyst til å ta deg på en liten reise om hvordan jeg har endt opp med å lage den applikasjonen jeg har levert inn og ikke minst gå forklare problemer jeg støtet på gjennom denne reisen. Jeg skal også snakke om kode filer og struktur og ikke minst modellere og beskrive hvordan en brukers opplevelse gjennom min applikasjon skal være. I oppgaven blir vi bedt om å lage en Tic Tac Toe Android applikasjon. Her har det blitt satt en rekke krav til applikasjonen, og disse lyder som

- Applikasjonen skal ha en Fragment-Arkitektur
- Applikasjonen skal gjøre bruk av et eksternt api
- Applikasjonen skal gjøre bruk av en lokal database.
- Applikasjonen skal ha logikk for å kunne spille alene mot telefonen.

Videre har vi også fått krav om å følge Android sine designprinsipper, samt arkitektur og kodestandard. Jeg skal også gå gjennom hvordan jeg har prøvet å opprettholde disse kravene som har blitt satt for meg. Ikke bare dette har jeg prøvd å lage noen ekstra krav for meg selv hvor jeg har lyst å lage en fullverdig applikasjon, som skal fungere på flere telefoner med ulike systemer, størrelser og ikke minst lage en god design, som får brukeren til å besøke applikasjonen flere ganger. Jeg har også prøvet å opprettholde en god mappestruktur med gode navngivning på klasser og variabler slik at hvem, som helst skal ha muligheten til å finne klasser eller metoder som de har lyst å lese på. Jeg skal også gå gjennom ulike tester jeg har gjort på applikasjonen, alt i fra System tester, Unit tester, Integrasjonstester og ikke minst Brukertester.

Mye av dette skal gåes gjennom i dokumentasjon hvor jeg forklarer hvilket tiltak måtte til for å nå disse kravene som har blitt satt av meg selv og ikke minst av foreleser.

Start fasen i prosjektet

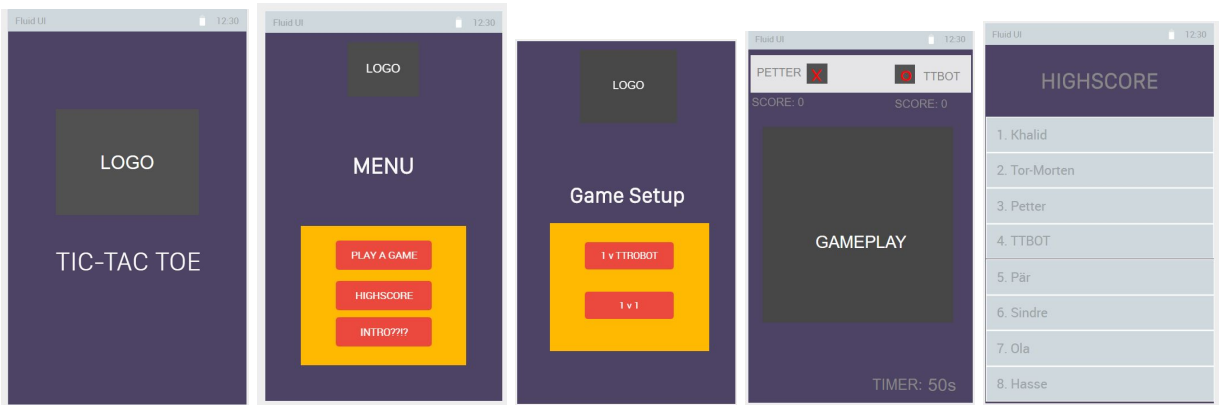
Da vi fikk oppgaven utlevert brukte jeg de dagene til å bli kjent med oppgaven og deres krav som ble satt for meg. Jeg hadde en liten anelse på hvordan jeg ville løse oppgaven. Dermed var bare å sette seg ned med penn å papir og prøve å lage en liten reise gjennom applikasjonen. Her tegnet jeg opp hvordan visse activities og fragments skulle være og hvordan disse skulle oppføre seg. Jeg satt noen hovedmål til meg

selv om å lage en applikasjon som var ganske “clean” i design og ikke minst å lage reisen til brukeren så kort som mulig slik at de kunne komme til deres respektive activities eller fragments.

Etter å ha tegnet opp mye av dette på papir så var det bare å gå over til en datamaskin og prøve å modellere en liten prototype av hvordan applikasjonen min skulle se ut. Jeg hadde ikke noen gode programmer å bruke til dette her, Så endte opp å bruke en delvis gratis tjeneste med navn FluidUI.
Her fikk jeg modellert opp hvordan jeg hadde lyst applikasjonen min skulle se ut med de virkemidlene jeg hadde mulighet til å bruke.

Ettersom mye av dette var på plass brukte jeg min resterende tid til å lese på hvordan vi faktisk bruker fragments, activities på en god måte og ikke minst hvordan kode og arkitektur skal være i Android programmering.
Et av mine største spørsmål, som ble delvis svart på var hvordan vi egentlig skulle bruke Fragments og når vi skulle bruke dette? Jeg fikk ikke noen direkte svar på akkurat dette på internett, men fikk en liten pekepinne på hvordan vi skulle bruke fragments. En av disse svarene var at du bruker fragments når du har en Activity som har samme design gjennom hele applikasjonen, men bare komponenter måtte byttes ut. Jeg synes dette hørtes fornuftig ut og forhørte også med læreren hvor jeg stilte det samme spørsmålet. Han fortalte meg at mye av slike valg var utvikleren som bestemte og at det ikke finnes fasitsvar på det. Dermed gikk jeg tilbake til papiret og modellerte en siste gang på hva i Applikasjonen min skulle være fragments og activities. Mye av dette begynte endelig å sitte på plass og alt som gjenstår nå var bare å finne de riktige bildene jeg ville bruke og lage et Android prosjekt på Android Studio.

Før jeg avslutter dette segmentet i dokumentasjonen, vil jeg legge ved bilder av skisser av hvordan applikasjonen min skulle se ut og ikke minst et av skisse arkene mine som jeg tegnet på papir.



Design

Du husker kanskje at jeg skrev tidligere at et av hovedmålene mine når det kom til design var å kunne lage en applikasjon som hadde en ganske “clean design”. Det er her mye av min tid gikk til å prøve å finne den riktige designet som får andre mennesker til å besøke applikasjonen mer, men samtidig ikke bombardere brukeren med mange bilder og tekst. En kombinasjon av disse to var noe jeg strevet ganske lenge med. Men jeg lot meg ikke bli stoppet av det.
Jeg endte opp med to utkast av applikasjonen. Disse vil jeg komme litt mer inn på i dette segmentet.

Designet for alle telefoner

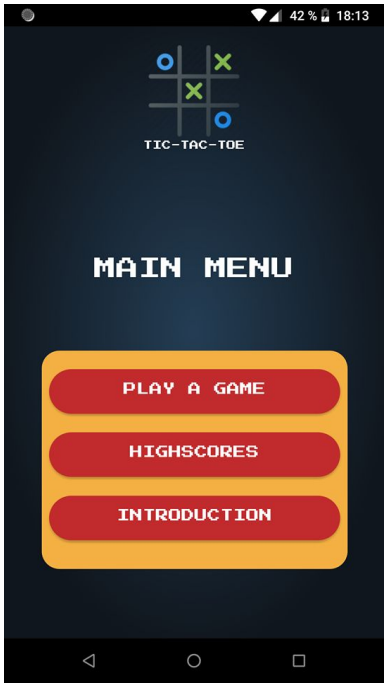
En av hovedmålene mine for applikasjonen var også at den skulle fungere på tvers over mobil-størrelser og måten vi kunne oppnå dette på var å forholde meg til Google’s anbefalte layout, som er da Constraint Layout. Grunnen til at jeg forholdt meg for det meste til Constraint Layout var fordi jeg ville at komponenter som lå på activitiet eller fragmentet skulle forbli der uansett hvilket telefon størrelse mobilapplikasjon kjørte på. Relativ Layout tilbyr oss dessverre ikke dette. Det holdt dessverre ikke for meg å bare bruke Constraint Layout. Jeg la også komponenter inn i type layouts, som Linear Layout for så contrainte layouten. Slik at komponentene aldri hadde mulighet for bevege seg ut av layouten de var i.

Bilder jeg brukte i applikasjonen måtte også taes høyde for. En måte jeg gjør i applikasjonen min for at bilder ikke skulle skalere seg for stort eller for liten var å finne ulike bildestørrelser for mobilens density pixels. Idag har flere mobiltelefoner forskjellige størrelse og ikke minst forskjellig oppløsninger. Jeg har dermed prøvet å ta høyde for dette slik at en bruker med 4,7’’ skjerm vil ha muligheten til å få den samme app opplevelsen, som en bruker som har en mobiltelefon med 5,4’’ skjerm. En av ressursene jeg brukte for nettopp denne type arbeid var en side med navn Image Baker. Her kunne jeg legge inn bilder jeg hadde lyst å bruke i applikasjonen min og få tilbake en ferdig generert mappe med forskjellige størrelser som passer for telefoner med disse type density pixels: hdpi, idpi, mdpi, xhdpi, xxhdpi og xxxhdpi. Disse mappene ligger under res i prosjektet mitt.

Hver gang jeg gjorde endringer eller laget nye fragments, activities testet jeg om disse fungerte kryss over forskjellige skjermstørrelser. Dette er uten tvil en av undervurderte designprinsipper i Android. Det er utrolig viktig applikasjonen har muligheten til å kjøre på telefoner med forskjellig størrelser. Dette kommer fra en person som sitter på en Nexus 6P hvor skjermen min er 5,7”.

Applikasjonen's design

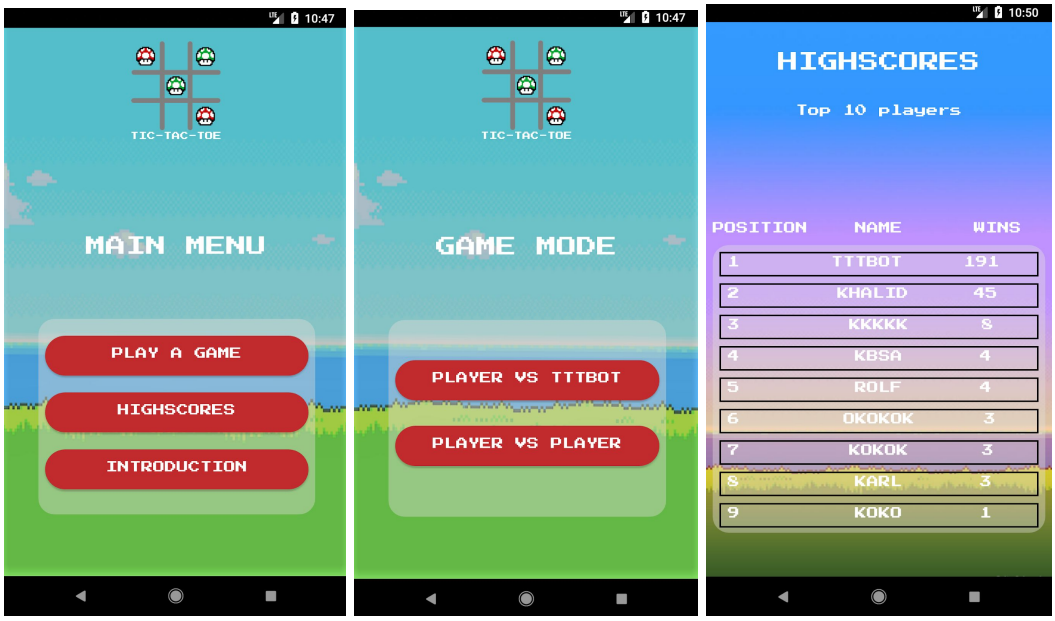
Når skissene var på plass var alt egentlig tilrettelagt for meg å bare begynne på å leke med XML filene i Android Studio. Her prøvde jeg å lage det slik som skissene viser. Jeg hadde allerede en slags design jeg ville gå for i hodet hvor temaets farger skulle være mørke blå, oransj og rød. Jeg måtte være 100 % sikker på at det var denne type design jeg ville absolutt gå for før det skulle bli forsent. Så det jeg gjorde var at jeg startet først og fremst med et lite utkast i Android Studio. Hvor jeg prøvet å replikere hvordan jeg hadde gjort det i skissene, som ble vist ovenfor. Målet mitt var alltid i denne eksamen å få til en applikasjon som hadde en kul design men også var ganske clean og brukervennlig.
Etter jeg hadde gjort ferdig utkastet mitt var det bare å forhøre seg med andre mennesker om hva de syntes om designet. Her spurte jeg nærmeste venner og familie om hva de syntes om applikasjonens design så langt.



(*Et lite bilde av det tidligere designet mitt av applikasjonen.)*

De fleste tilbakemeldingene jeg fikk på dette var at knappene og boksen knappene lå i var flotte, men at dette var en ganske kjedelig design. Jeg ble tipset om å heller velge et slags tema som hadde en slags rød tråd gjennom hele applikasjonen. Hvor jeg burde bruke musikk og bilder for å fremheve dette. Jeg måtte senere sette meg ned å prøve å få en god løsning på dette. Jeg skrev et par kule temaer jeg ville gå for. Det var da jeg fant ut at jeg heller burde gå for et tema jeg interesserer meg for. Da falt valget mitt på 8-Bit tema.

Grunnen til at jeg gikk for 8 bit tema I appen min var fordi de fleste i dag har en form for kjennskap til slike type spill. Her prøver jeg aktivt å resonere med bruker ved å bruke 8 bit bilder, som minner de litt om type 8 bit spill, som Mario.
Jeg var fast bestemt på å opprettholde hvordan jeg hadde satt opp applikasjonen, slik som jeg gjorde det i skissene mine. Alt jeg trengte å gjøre nå var bare å finne de rette bildene og sangene jeg ville bruke. Etter mye testing for å se hva som passet inn med applikasjonen så endte jeg opp men slik design som dette.



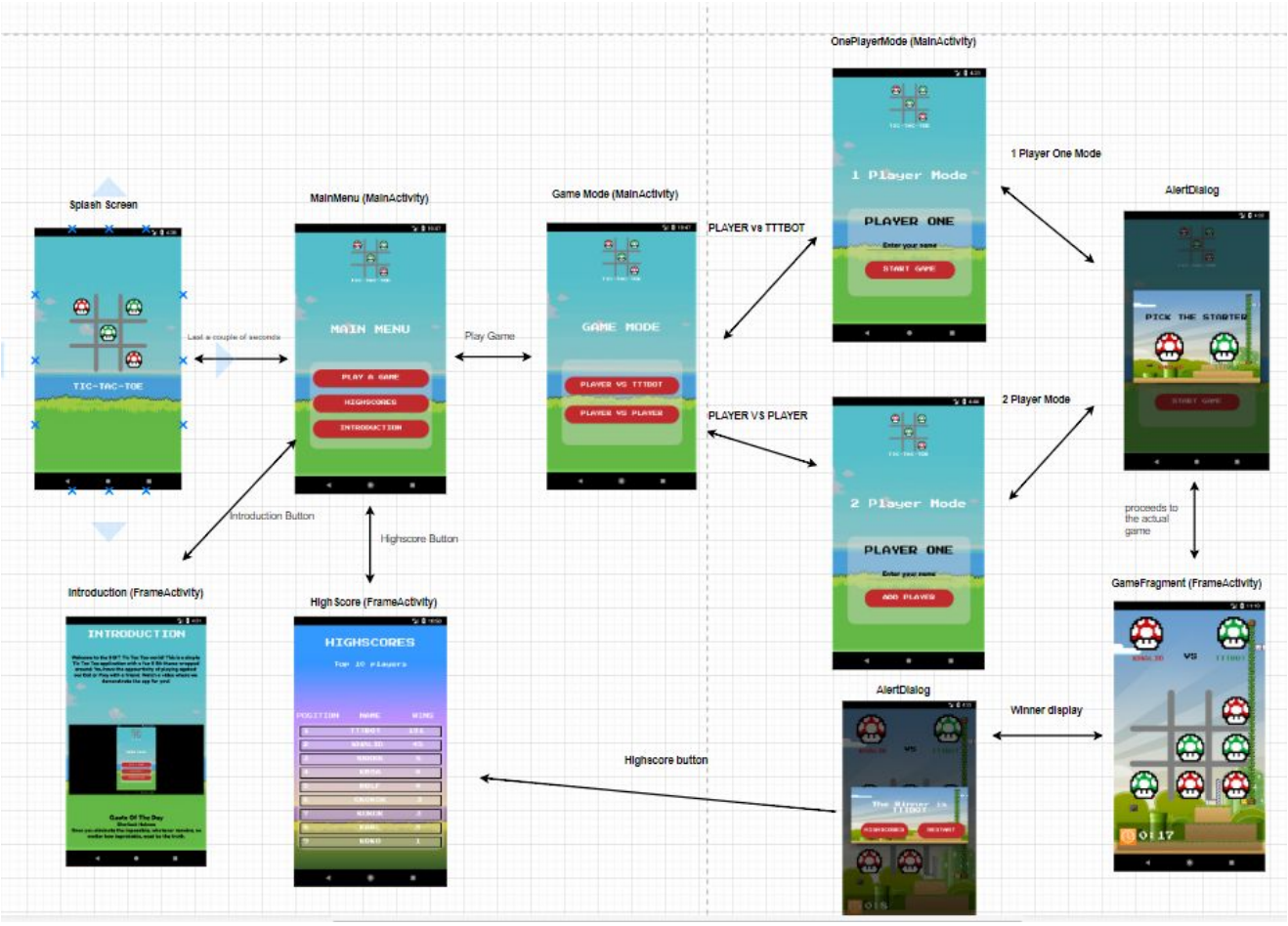
Jeg er relativt fornøyd med designet jeg sitter med Applikasjonen i dag. Jeg mener at jeg oppfyller kravene jeg har satt for meg selv når det kommer til å opprettholde en clean og kul design, som kanskje får flere brukere til å besøke applikasjonen flere ganger. For å fremheve det visuelle så har jeg valgt også spille av musikk gjennom hele applikasjonen. Applikasjonen vil spille av lyd for hver gang du trykker på knapper og ikke minst skifte sang når du kommer til spill delen av applikasjonen. Jeg skal komme nærmere på hvordan AudioPlayer klassen fungerer senere i Dokumentasjonen.

Jeg hadde tidlig i fasen en strukturert måte jeg ville lage denne applikasjonen på. En av disse delene jeg var fast bestemt på var å lage disse knappene på denne måten. Jeg ville ha store knapper i en meny type boks. Jeg ville at brukere ved et kjapt blikk vil forstå at disse knappene er for å samhandle med. Etter par knapp klikking vil brukeren merke at all samhandling med Applikasjonen vil foregå gjennom denne boksen her. Grunnen til at jeg har gjort det som dette er fordi jeg ville at designet mitt skulle være lett og forstå. Slik at alle brukere uansett hvilket

alder de er i. Skal ha muligheten til å finne fram det de trenger. Om det er å starte spill eller om det er se på HighScore.

Reise av Applikasjonen

I dette segmentet av applikasjonen har jeg lyst å skissere en slags flowchart av alle skjermflater av min applikasjon. Hvor jeg viser hvilke knapper i applikasjonen vil ta deg til for hver gang du trykker på de. Som jeg tidligere skrev så var alltid målet mitt å ha en brukervennlig måte på hvordan brukeren skulle integrere med løsningen min.
Jeg ville ha så lite knapper, som mulig for at brukeren kunne forstå hvor han/hun skulle til. Dermed var valgte jeg å gå for en løsning hvor jeg kun hadde maks 3 knapper eller mindre. Planen bak dette var jo mindre knapper jo mindre sannsynlighet skulle det være for brukeren å rote seg vekk i applikasjonen min. Dette resulterte også med at applikasjonen min ikke trengte mye tekst ettersom knapper og menyen var selvforklarende for at de skulle skjønne ting. Dette er også en av de tilbakemeldingene jeg fikk etter jeg utførte flere brukertester, som jeg skal komme til i slutten av dokumentasjonen.



https://drive.google.com/file/d/15mVhYmroPC6NgQBI-4zsCt_sy33_iuNo/view

Kodefiler og Struktur

I dette segmentet vil jeg snakke om valgene jeg har tatt når det kommer til kodefilene mine. Jeg har tidlig i prosjektet mitt bestemt å ha en ordentlig mappe struktur hvor jeg deretter legger kodefilene hvor de hører hjemme. Et av målene mine gjennom prosjektet var å skrive forståelig og lett leselig kode slik at hvem som helst kunne komme inn i koden min og endre på det som måtte trenges. Jeg hadde som mål å ikke ha for mye kodelinjer i klassene mine og begynte relativt tidlig i fasen å dele opp kode i flere klasser. Dette er en av hovedgrunnene til at jeg idag sitter på 26 klasser til årets Android eksamen. Jeg har også brukt Design Patterns, som singleton for å løse Audio problemer i Applikasjonen min. En annen ting jeg gjorde en del av var å alltid se hvilke type klasser, som hadde samme metoder og deretter dele dette her opp ved å lage Interface klasser som tok i bruk metodene.

En av kravene for denne eksamen var at løsningen skulle følge Android sine kodestandard. Dette var jeg veldig usikker på ettersom vi ikke gikk noe særlig innpå hvordan type naming convention Android brukte. Jeg støtet på et artikkel hvor de går gjennom hvordan navngivning av grensesnitt komponenter skulle navngis *Lenke her*. Jeg fulgte dette rådet og gjorde også et ekstensiv søk på navngivning av variable. Her fant jeg et Stackoverflow innlegg hvor det blir forklart at man burde navngi alle medlemsvariabler med “m” og statisk variabler med “s” etterfulgt navn *Lenke HER*. Jeg har dermed prøvd å opprettholde disse kravene gjennom hele prosjektet.

En av kode prinsippene i Android er å aldri å hardkode inn Strings. Det er her vi burde alltid bruke Strings.xml til. Dette gjelder også av farger hvor vi lagrer all fargen i colors.xml. Jeg har i gjennom prosjektet mitt opprettholdt disse prinsippene hvor alle mine Strings, farge ligger i Strings.xml og Colors.xml.

.

Beskrivelse av kodefiler

I dette segmentet skal jeg gi kort beskrivelse på klasser jeg har laget og ikke minst forklare litt hvorfor jeg har delt opp koden flere segmenter av løsningen min i flere klasser. Jeg vil gjerne starte å snakke om disse klassene punktvis dermed følger jeg mappestrukturen jeg har allerede laget.

Activities (Folder)

FrameActivity

Denne klassen her er en vanlig Activity. Jeg bruker denne klassen for det meste til å bytte fragments på den. I XML'en av denne klassen har vi en framelayout som matcher høyden og bredden til parent, som da vi bruker for å legge på forskjellige fragments oppå. En av hovedgrunnene til at jeg gjør dette er fordi jeg ikke vil lage flere Activities og dermed syntes dette var lurt å gjøre enn å måtte sitte på 3 ekstra activities. Jeg bruker følgende fragments i FrameActivity: GameFragment, HighScoreFragment og IntroductionFragment.

IntroActivity

I denne klassen/Activitiet bruker jeg primært bare til å kjøre en liten SplashScreen. Dette vil da være det første som kjøres når applikasjonen kjøres. Her vil logo og navnet på applikasjonen bli presentert i en slags Fade in Animasjon. Dette vil bli kjørt i et par sekunder og aldri igjen med mindre du da velger å slå av applikasjonen og kjøre det på nytt igjen.

MainActivity

I denne klassen/Activitet er her hvor brukeren for det meste integrere mest med applikasjonen. Her har jeg valgt å kjøre en fragment arkitektur hvor brukeren har muligheten til å samhandle med helt til brukeren eventuelt bestemmer for å spille et spill. Du husker kanskje at jeg nevnte at jeg syntes det var lurt å bruke fragments når vi hadde lyst å forandre på deler av activitiet, men ikke hele?
Det er slik måte jeg har gått for å løse MainActivity. Her har jeg en stor hvit/grå lignende boks hvor brukeren da samhandle med knapper. Alle disse knappene ligger i hver sine egne fragments klasser. Disse nevner jeg senere i dette segmentet.
Brukeren vil da samhandle med disse fragment klassene helt til spillet starter, Highscore eller Introduction blir trykket på. For da vil vi igjen forflytte oss til FrameActivity.

API (Folder)

HTTPHandler

Denne klassen bruker vi primært for å gjøre et HTTP-Request til et REST API hvor vi får returnert JSON data, som vi leser inn i en string og returnerer da dette. Kallet på denne klassen vil da skje fra et asynkront slik at vi ikke har for mye på mainthread.

QTDModel (Quote Of The Day for short)

Denne klassen her skal da være en simple Model klasse som vi bruker for å lage objekter av JSON respons vi da får. Her inneholder det bare ganske simple setter og getters metoder. Klassen skal ikke brukes til noe mer enn dette.

Data (Folder)

Database (Folder)

DBConfig

DBConfig består av alle properties, som databasen vår da trenger. Her ligger det bare offentlige statiske variabler hvor Database navnet, versjon, tabell navn ligger.
Jeg la dette i et eget klasse fordi det så mer ryddigere ut og ikke minst lett å finne fram hvis det du skulle være noe som måtte forandres på.

SQLiteHelperClass

Denne klassen skal primært ta seg av å sette opp en Database på telefonen vår med mindre den allerede finnes allerede fra før av. Jeg kunne forsovet legge dette sammen med DB handleren, men syntes det ikke helt hører helt til der. Dermed har jeg gått for å lage den i egen klasse. Når klassen førstet gang oppretter en database på telefonen vil den også ta høyde for å populere Databasen vår med TTTBot slik at brukeren aldri vil ha muligheten til å prøve å spille med det spiller navnet.

DBHandler

I DBHandler klassen har vi alt av CRUD metoder opp mot databasen. Alt ifra fra å hente, legge til, oppdatere, slette og sist men ikke minst hente alle spillere fra databasen. Jeg hadde store planer om å utstyre denne klassen her med en Singleton design pattern ettersom vi da oppretter et objekt av den i noen av klassene, men jeg var usikker på om jeg gikk i mot noen av prinsippene til Android når det kommer til Database. Dermed valgte jeg ikke å gjøre dette.

Player

Denne klassen her er en slags Player Model klasse, som vi da bruker for å opprette objekter. Klassen er bare utstyrt med to fields navn og wins og metoder som setters og getters. Vi bruker primært denne klassen bare for å opprette objekter av spillere fra databasen eller når vi første gang spiller med et navn som ikke finnes i db'en.

CustomViewAdapter

CustomViewAdapter er en klasse som da extender RecyclerView.Adapter klasse. Vi vet allerede at en Adapter er en slags controller for RecyvlerView lista vår. Ettersom jeg har brukt en del tid på design hadde jeg lyst til å også designe hvordan lista skulle se ut. Dette resulterte med at jeg måtte lage en egen CustomAdapter, som deretter brukte ViewHolder klasse som et bindeledd for det XML'en som vi skulle skulle populere.
Vi vil da opprette et objekt av denne klassen i HighScore for så sette inn en liste av topp 10 spillere fra databasen.

Fragments (Folder)

FragmentOptions (Folder)

MainOptions

Denne fragment klassen er også relativt simpel. XML'en til MainOptions består av 3 knapper hvor du har mulighet til å starte prosessen av å spille et spill. De to resterende knappene vil da ta deg til HighScore eller Introduction. De to sistnevnte vil da ta deg til en FrameActivity. Hvor FrameActivity da starter opp deres respektive Fragments, som knappene skal da vise. Jeg bruker Intent for å si ifra hvilken fragment som skal da vises på FrameActivity.

GameModeOptions

Dette er en av de Fragment klassene som blir presentert i boksen i MainActivity, som jeg tidligere snakket om. Denne er klassen er ganske simpel ettersom XML'en består av to knapper hvor brukeren har mulighet til å velge hvilken GameMode de vil spille på. Disse to respektive knappene vil da ta deg til sine egne respektive fragments. Hvor den ene leder deg til OnePlayerMode Fragment og den andre til TwoPlayerMode Fragment.

OnePlayerMode & TwoPlayerMode

I disse fragment klassene skal brukeren ha muligheten til å legge inn et navn før de da starter å spille en runde mot TTTBOT eller mot en venn. Jeg tar litt ekstra høyde på validering i denne klassen. Jeg tar høyde for at brukeren ikke har lov til å spille med navnet TTTBOT i tillegg til at navnet aldri kan være mindre bokstaver enn 4 eller mer en 8. Det sistnevnte gjør jeg egentlig på grunn av designet sitt skyld ettersom navnet begynner da å ta større plass i spill fragmentet.Jeg tar også høyde for at spilleren ikke har noen som helst mulighet til å spille med to av de samme spillere fra Databasen.

Jeg gjør også et sjekk om spilleren allerede finnes i Databasen. Hvis den da finnes i Databasen vil jeg da kalle på en AlertDialog klasse sin metode hvor jeg da informerer brukeren om at spilleren allerede finnes i Databasen og hvis personen har lyst til å spille med spilleren så er det bare trykke start game igjen.

Jeg bruker alt i fra Alertdialog til toast-meldinger for å fortelle brukeren om de har gjort noe galt. Når spillet starter vil da disse to klassene sende videre spiller objektene og hvilken game mode I til GameFragmentet, som ligger da i FrameActivity. Dette gjør jeg da gjennom Intent.

FragmentsGame (Folder)

GameFragments

Denne fragment klassen vil vises på FrameActivity. Jeg hadde som mål for denne klassen å bare holde den så liten som mulig og heller bruke hjelpe klasser for håndtering av setting av bilder på knapper, logikk og logging. Denne klassen her vil sjekke først og fremst hvilken gamemode var det som ble valgt. hvilken spiller som da starter og setting av navnene på spillerne i scoreboardet som blir presentert helt øverst av fragmentet. Jeg tar også høyde for at timeren da starter på OnResume og pauser timeren i OnPause i Fragmentet sitt livssyklus Denne klassen vil her også lage en instans av GameLogic klassen min og begynne å fylle inn informasjon der inne om hvilken spiller som starter og hvilket gamemode det er brukeren har valgt.

GameHighScores

Denne fragment klassen vil vises på FrameActivity. Hovedmålet ved denne klassen er at den skal da starte å populere RecyclerView. Den vil da lage et instans av CustomViewAdapter også starter å populere to 10 beste spillere fra Databasen. Denne klassen her gjør egentlig ikke så mye mer enn dette.

FragmentsOther (Folder)

Introduction

Denne fragmentet klassen vil vises på FrameActivity. Denne klassen er en slags demonstrasjon av Applikasjonen. Hvor brukeren kan få en slags introduksjon av hele Applikasjonen. Klassen gjør ikke så mye enn å starte opp en Youtube video, som jeg har laget. Vi bruker et Youtube API for å få dette til. Jeg gjorde dette før jeg visste at dette ikke følger kravene for eksternt api, Men jeg bestemte å ha dette og heller implementere ett http kall til et REST Api. Introduction klassen vil da ha en privat inner class som gjør et asyncTask på å hente Quote Of The Day fra et REST API lenke her.

Game (Folder)

BotLogic

I denne klassen tar jeg av meg hvordan bot/AI logikken skal fungere. Klassen er utført med metoder for hvordan Boten skal forsvare og angripe og ikke minst hvordan den skal passe seg for kjente scenarioer hvor brukeren prøver da å lure den til å prøve å få den til å tape.
Disse metodene vil alltid returnere et tall, som indikerer hvor boten burde legge på brettet.
Klassens metoder blir kalt på fra GameLogic class.

GameLogic

Denne klassen her vil inneholde all spill logikk for One Player Mode og Two Player mode. Alt i fra hvem som legger på til hvilken type mode det spilles på vil bli håndtert her. Klassen vil også lage en instans av GameObserver klasse og sende hvert trekk som blir gjort i spillet hvor den da logger dette. Denne klassen sitter også på litt av Bot/AI logikken. Vi bruker primært sett BotLogic klassen for å gjøre sjekk på hvilke trekk som burde gjøres.

Klassen bruker også et hjelper klasse kalt for GameResultLogic hvor den sjekker hver gang det har blitt lagt på 5 på brettet om det finnes noen vinnere eller ikke. Hvis det skulle da være seg at det fantes en spiller som vant bruker vi AlertDialog klasse hvor vi viser brukeren hvem som har vunnet.

GameObserver

Denne klassen er ganske simpel. Den er utstyrt med 3 HashMaps i fields hvor disse blir brukt til å logge hver gang en spiller tar et valg. Vi bruker en HashMap til brettet, En til spiller 1 og en til Spiller 2. Grunnen til jeg gjorde dette her var fordi jeg hadde planer om å lage en AI i slutten av løsningen om jeg fikk tid. Når jeg endelig fikk til AI så var det supert at jeg logget all trekk som blir gjort. Dette hjalp meg utrolig mye for å optimalisere AI'en min. Klassen er bare utstyrt med setter og getters metode og ikke noe mer enn dette.

GameResultLogic

Denne klassen blir primært brukt for å sjekke om det er noen vinnere eller ikke på brettet. Klassen er bare utstyrt med en metode hvor vi bruker 2 dimensjonal array med alle mulige måter å vinne på og sjekker brettet vår på dette. Metoden vil da returnere 0 om det ikke er noen vinner, 2 hvis spiller 2 har vunnet og 1 hvis spiller 1 har vunnet. GameLogic vil da ta seg av resterende biten av å presentere dette for brukeren.

GameNotifiers (Folder)

AfterGameAlert, StarterPlayerAlert, UserExistAlert

Disse tre klassene er relativt ganske like. Jeg valgte å dele opp mine Alert Dialogs i forskjellige klasser slik at det var lettere å forandre på disse om det skulle være noe. Vi kan godt starte med å snakke om StarterPlayerAlert dialog. Denne har en enkel metode hvor du må velge hvem som skal starte. Du vil bli møtt av to bilder og navn på spillerene. Metoden vil senda videre informasjonen til GameFragmentet og fortelle hvem som skal starte av spiller 1 og spiller 2. Her bruker jeg eksplisitt intent.

UserExistAlert er ganske lik. Jeg bestemte å lage en AlertDialog når brukeren prøver å spille med en eksisterende spiller fra Databasen. Her forklarer jeg brukeren at de prøver å spille med en spiller, som allerede finnes. Og at hvis du ikke har lyst til å spille med denne spilleren så bør de bytte navn. Hvis du derimot har lyst å spille med spilleren må du bare trykke Start Game engang til.

AfterGameAlert derimot blir kalt på fra GameLogic. Så snart det er en vinner eller uavgjort vil dette da presenteres til brukeren. Hvis det skulle da være seg at det var en vinner vil AfterGameAlert klassen ta seg av oppdatering av spillerens nye score til Databasen. AlertDialog XML'en er utstyrt med to knapper hvor du har muligheten til å sjekke highscore eller restarte spillet.

Interfaces (Folder)

DatabaseMethods & InterfaceFragments

Disse to Interface klassene ble laget fordi å komprimere litt mer kode. Klasser som har mye av de samme metodene skal da implementere disse metodene. DatabaseMethods Interfacet brukes av OnePlayerMode og TwoPlayerMode klasser. Begge disse to klassene sjekker om en spiller finnes i Databasen og ikke minst legge til i Databasen. Da er det lettere for disse klassene å bare implementere disse metodene fra DatabaseMethods interfacet.

Det samme går også for InterfaceFragments Interfacet. Jeg laget dette primært fordi jeg la merke til at flere klasser som startet opp et fragment hadde primært like metoder. Jeg valgte heller legge denne metoden i et eget interface klasse slik alle disse klassene som setter opp et nytt fragment kunne bare hente det fra interfacet.

Jeg kunne også la hver av disse klassene extende et bestemt klasse som deretter extender Fragment og heller kalle på en bestemt metode. Men jeg var så godt i gang at jeg ikke kunne gjøre så store drastiske endringer på løsningen min.

Media (Folder)

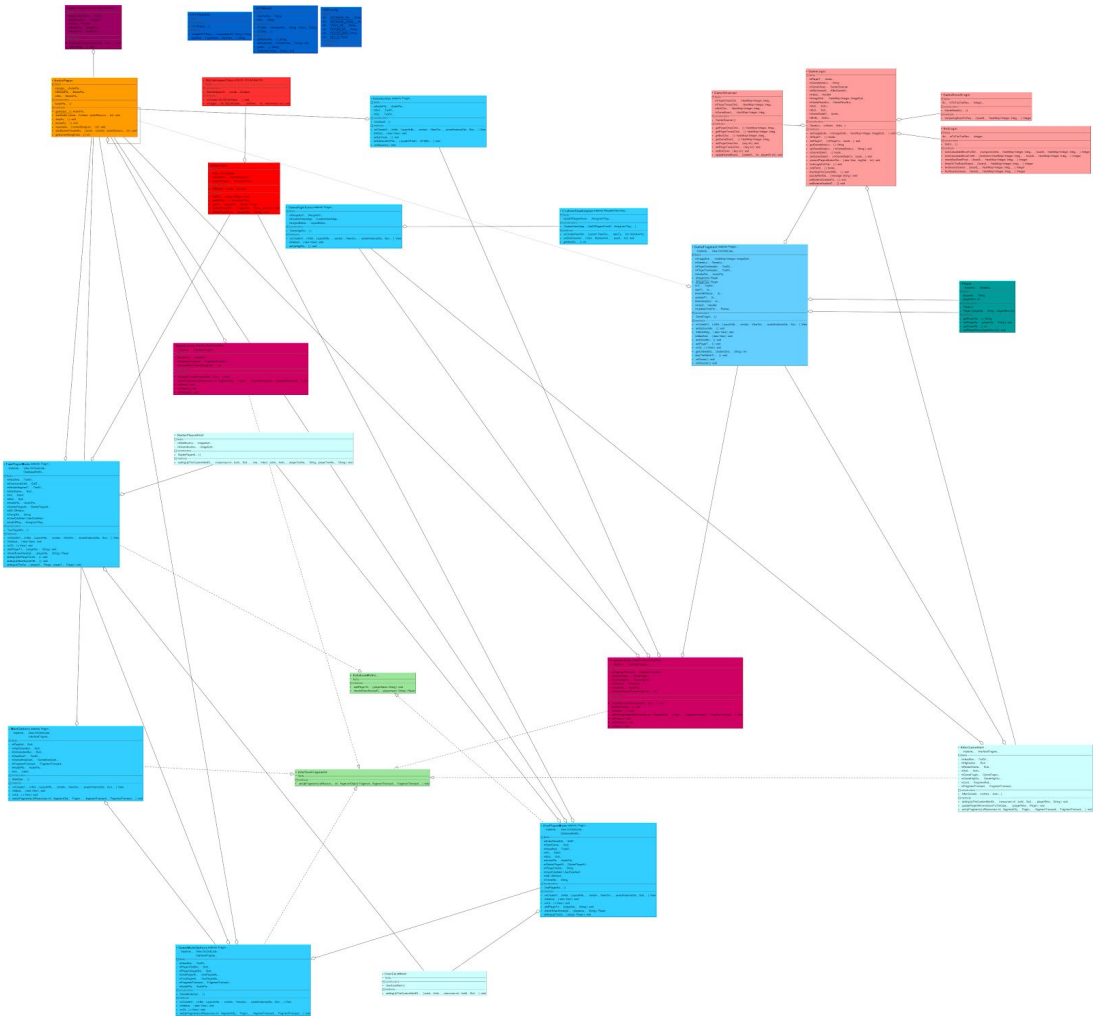
AudioPlayer

AudioPlayer er primært for å spille audio. Jeg bestemte å lage en egen klasse for dette og ikke minst utstyre denne klassen med et Design Pattern, Singleton. Grunnen til jeg gjorde dette var fordi jeg ville at flere klasser kunne bruke det samme objekter av AudioPlayer istedenfor å måtte lage et helt nytt et. Musikken ville dermed oppføre seg veldig rart om du hver gang laget et nytt objekt. Klassen er utstyrt med visse metoder hvor du kan stoppe sangen pause sangen og ikke minst starte å spille av sangen hvor du første gang pauset den. Dette vil si hvis du for eksempel går ut av appen. Vil klassen da huske hvor langt du hadde kommet i sangen og deretter spille dette av når du har kommet tilbake igjen. Ettersom jeg spiller forskjellige sanger i Activites så var det lurt å løse dette på denne måten.

Class Diagram

I dette segmentet skal jeg vise deg en slags Klasse Diagram over alle klassene jeg har i prosjektet mitt og hvordan disse klassene er da koblet opp mot hverandre. Jeg hadde gjerne lyst å gjøre dette gjennom LucidChart, men ettersom Mye av tiden min gikk til løsningen fant jeg et PlugIn, som genererte dette for meg. Her har jeg brukt noe som kalles for SimpleUMLCE, som jeg da måtte laste ned og legge inn i Android. Det er verdt å nevne at det kanskje ser litt rotete ut, men jeg har valgt å legge på farger på visse klasser. Jeg legger også dette som et vedlegg i PDF format om du skulle ha lyst til å ta et ekstra titt.

Lilla - Activities
Oransje = Medi
Mørkeblå = Klasser som ikke har noen tilkobling
Lyse blå = Alert Dialogs
Blå: Fragment klasser
Rødt = Database relaterte klasser
Grønn = Interface Klasser
Rosa = Alt som har med spill logikk å gjøre.



Styrker og Svakheter

I dette segmentet skal jeg gå gjennom mine styrker og svakheter i kodefilene og løsningen jeg har laget. Jeg nevnte tidligere at målet mitt gjennom prosjektet var alltid å prøve å dele ting litt opp i flere klasser og metoder. Slik at visse klasser ikke var overflødig med kode. Jeg har prøvet å opprettholde målet mitt gjennom prosjektet, men mer på dette skal jeg komme litt innpå

Styrker i løsningen og kodefilene

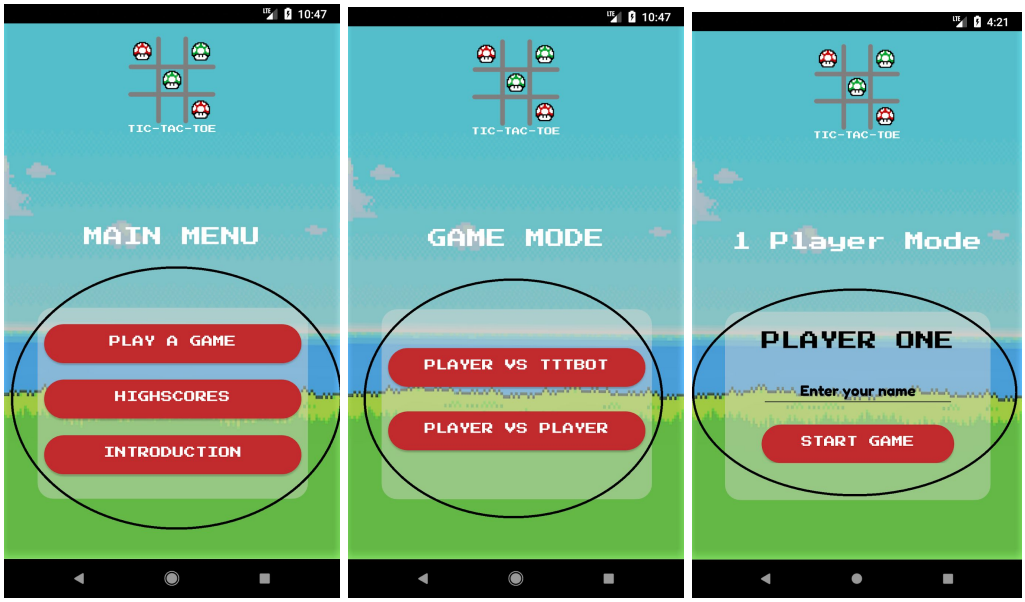
Jeg vil si at jeg har en ganske god mappestruktur hvor av alt du trenger å finne er bare et par tastetrykk unna. Målet mitt var alltid å dele ting opp i riktig klasser og gi gode navn på disse klassene slik at du ikke hadde problemer med å finne disse. Jeg hadde selv som et mål hvor jeg prøvde å unngå å ha mer enn 300 linje kode i klasser. Et godt eksempel på god deling av kode er måten jeg har løst på spill logikken. GameFragmentet mitt gjør så lite som mulig og heller bruker eksterne hjelper klasser for å håndtere mye av spill logikken (*GameObserver, Gamelogic, Botogic og GameResultLogic)*. Jeg har også brukt Interfaces for klasser, som har bruker samme metoder. Slik at disse klassene kan da bare implementere disse metoder.

Jeg har gjennom prosjektet alltid prøvd å følge Android sine kodestandard hvor jeg alltid legger strings hvor de hører hjemme og oppfyller kravene for navngivning av medlem, statiske variabler. Jeg prøver også følge en god naming convention i komponenter og Layout i XML'ene også.
Jeg tar også ekstra høyde for små ting i spillet f.e at du ikke kan spille med TTTBOT navnet at du heller ikke har lovt til å spille med to av de samme spiller.

Gjennom prosjektet har jeg også lest meg opp på hva Google egentlig anbefaler oss utviklere å bruke. Et godt eksempel på dette er RecyclerView. Google anbefaler oss utviklere å heller bruke dette enn ListView ettersom RecyclerView bruker mindre ressurser. I tillegg til at vi burde helst jobbe med ConstraintLayout istedenfor RelativeLayout.

Jeg føler også en av styrkene av løsningen er måten jeg har brukt fragments på. Et godt eksempel på dette kan være hvordan brukeren samhandle med knappene mine i menyen hvor disse knappene ligger i Fragments. Hvor jeg heller bare skifter ut hovedteksten i Acitivitien for å indikere at brukeren er nå f.e. I "Game Mode". Jeg føler dette er en riktig måte bruke Fragments på. Hvor du ikke trenger å skifte ut mye av Activitien, men bare enkelte deler.

Under vil du se ett enkelt demonstrasjon på dette.



Svakheter i løsningen og kodefilene

Det er selvsagt visse ting jeg ikke er fornøyd med løsningen min. Noen av disse tingene er at jeg har 3 Activities. Når jeg burde kanskje forholdt meg til 2. IntroActivitet, som kjører et SplashScreen i 5 sekund og aldri igjen er litt unødvendig. Jeg kunne gjerne gjort dette om til et fragment og heller brukt det i FrameAcitvity. Dette hadde tatt mindre ressurser enn å faktisk måtte laste opp et helt Activity.

En annen ting jeg ikke er helt fornøyd med er at jeg har deler av bot logikken min hos GameLogic. Jeg kunne gjerne tatt dette her ut å legge det i sammen med BotLogic klassen. Grunnen til at jeg ikke gjorde dette var fordi jeg i bunn å grunn ikke hadde planer om å gå for AI. De siste to dagene før levering av eksamen prøvde jeg bare å lage litt logikk for boten hvor den kunne forsvare seg og angripe. Dette utviklet til jeg absolutt ville prøve å lage en fullstendig AI. Dette valget slo litt på designet måten jeg har designet. Jeg har nå ingen form for easy bot eller hard bot. Jeg skulle kanskje planlagt dette litt bedre.

En av svakhetene jeg syntes er valget av et eksternt API jeg har tatt. Jeg hadde opprinnelig lyst til bruke noe som kunne være samsvar med applikasjonen. Men her endte jeg opp heller å velge vise fram Quote of the Day fra et Rest API. Jeg føler ikke at dette er noe helt passer inn i Applikasjonen, men dessverre fant jeg ikke noe som kunne ha passet inn med applikasjonen.

Botlogikk (AI)

Jeg nevnte tidligere at jeg ikke hadde noen planer å lage en fullstendig logikk for den. Jeg syntes ikke at det var riktig av meg å bare implementere en MinMax algoritme eller lage 500 linjer med If-Statements. Hvis jeg skulle absolutt gå for en AI så skulle dette være noe som var mitt. To dager før eksamen satt jeg meg ned å bestemte å prøve å lage litt logikk for boten min hvor den hadde mulighet for å forsvare og angripe. Jeg har i dette tilfellet to metoder som gjør dette for meg. Jeg satt plutselig på en løsning hvor jeg hadde mulighet til å implementere en AI løsning. Jeg sa tidligere i dokumentasjonen at jeg logget mye av hva som foregikk på brettet. Dette kom heldigvis til min unsetting. Koden min logger begge spillerne sine hånd og brettet. Jeg brukte disse delene til å å få implementert en vell fungerende AI. Det er verdt å nevne at AI'en vil i enkelte tilfeller velge tilfeldig. Men etter en god del spilling vil disse valgene mest sannsynlig være til hans fordel.

Etter en god del brukertesting hvor venner og familie testet TTBOT's logikk fant jeg et par hull eller scenarioer hvor jeg burde tette igjen med litt hardkoding. Dermed har jeg laget noe som kalles for ThreeBoardScenarioes, TwoBoardsScenarioes og FourBoardScenarioes metoder i Botlogikk klassen min. Jeg gjør et sjekk opp mot disse scenarioer hver gang en brikke blir lagt på brettet og prøve å stoppe dette hvis disse oppstår.

Jeg vil si at jeg er relativt fornøyd med AI'en jeg har bygget opp. Den opererer på en dynamisk måte. Jeg har ikke brukt noe form for ressurser eller noe lignende for denne delen av min løsning. Her har jeg bare prøvd å tenke utenfor boksen og heller lage noe som er mitt. Jeg følte at det var dette Tor-Morten var ute etter istedenfor å bare bruke en algoritme fra nett eller å hardkode inn 500 linjer med IF-Statements.

Tester (Enhetstester, Systemtester, Brukertester)

Enhetstester (JUNIT)

Jeg hadde ikke noe særlig erfaring med JUnit ettersom jeg da går Intelligente Systemer, men lot ikke dette stoppe meg fra å utføre noen tester i løsningen min. Etter å ha sett på en god del Youtube videoer hvordan vi kan utføre tester i JUnit har jeg prøvet meg på dette i løsningen min. Her har jeg valgt å kjøre JUnit tester enkle plasser hvor det passet inn. Jeg føler det er litt vanskelig å kjøre JUnit Tester på Android i forhold til Java. Jeg valgte å utføre tester på metoder i DatabaseHandleren min. Her testet jeg adding av person, hente spiller, endre spiller og sletting av spiller. Jeg har også utført et par tester på GameResultLogic klassen min hvor jeg sendte inn mockups av spillebrettet og sjekket om det var vinnere eller uavgjort i disse.

Systemtester (Espresso)

Jeg hadde litt kjennskap med Espresso tidligere før eksamen ble utlevert. Jeg prøvde å lese meg opp på hvordan man skulle teste i Espresso. Hva som var kjennetegnet på en god test. Jeg fant dessverre ikke noen gode forklaringer på dette og valgte heller å teste systemet litt på min måte. Jeg prøvde å simulere en hel espresso test hvor du skulle gå fra A til Å i applikasjonen, men her støtet jeg litt på store utfordringer. En av disse var at jeg ikke helt fikk simulert et helt spill. Jeg tror noe av dette var på grunn av spillet lå i en egen activity. Jeg valgte å se bort i fra dette og heller prøvde å teste så mye jeg kunne klare.

Jeg valgte å skippe Splash Screen og sjekket om Espresso testen min kunne trykke seg inn på forskjellige knappene som lå i boksen. Måten jeg testet om disse knappene ledet meg til der jeg skulle var å sjekke alltid en bestemt del av eksempelvis textviews i den fragmentet den var på.

Jeg kunne gjerne ønsket å gjøre flere tester, men vi har ikke særlig gått gjennom Espresso i timene og prøvet å få det beste ut av det jeg allerede kunne fra før av. Jeg fikk hvertfall testet alt utenom spillet.

Brukertesting

Jeg utførte en del brukertesting gjennom prosjektet mitt. Her har jeg for det meste snakket med venner, familiemedlemmer. Jeg fikk for det meste gode tilbakemeldinger på applikasjonen min. De fleste sa at det føltes virkelig ut som en fullverdig applikasjon, som de kunne ha lastet ned fra Google Play Store. Dette var en av målene mine for prosjektet. Det var så klart visse ting de ikke var fornøyd med eksempelvis at du måtte alltid trykke på Android Systemets pilknapp for å

gå tilbake. De mente at det hadde vært bedre om det fantes fysisk knapp i applikasjonen, som kunne ta deg tilbake til menyen. Denne kommentaren fikk jeg relativt sent inn og angrer på at jeg ikke fikk implementert dette.

Hvordan utførte jeg brukertesting?

Jeg hadde ikke så store erfaringer med hvordan vi utfører brukertesting. Jeg måtte tenke litt tilbake på hvordan vi utførte brukertester under Google Design Sprint i Smidig prosjekt. Her hadde vi laget en del spørsmål hvor vi ga brukeren enkle oppgaver, som å finne noe spesifikt i applikasjonen. Jeg ønsket å følge en slik struktur på hvordan jeg skulle utføre brukertester på applikasjonen min. Jeg skrev ned en rekke spørsmål til brukeren som:

- Du skal spille et Tic Tac Toe spill mot TTTBot. Hvordan hadde du gått frem?
- Du skal nå sjekke HighScore listen på appen.
- Du skal nå finne en spille modus hvor jeg og deg kan spille mot hverandre.

Etter disse oppgavene ble utført spurte jeg hva brukeren syntes om applikasjonen. Hva de likte med det og hva de ikke likte med det. Her var det bare å være så kritisk som mulig.
Jeg fikk for det meste gode tilbakemeldinger på design og hvor brukervennlig applikasjonen var. Det var også visse ting som de ikke var fornøyd med når det kom til design, som jeg fikk heldigvis endret fort.

Jeg er relativt fornøyd med løsningen jeg leverer inn til årets Android Programmerings eksamen. Jeg satt høye krav for meg selv og føler jeg har hvertfall nådd på et par av de. Målet mitt var alltid å lage en fullverdig applikasjon, som skulle være klar til å legges ut på Google Play Store. Jeg har selvfølgelig gjort det så godt jeg kunne for å nå dette målet.
Jeg har også prøvet godt gjennom hele prosjektet å holde en god kodestandard og god struktur. Jeg har også prøvet å oppfylle kravene som ble satt for meg i årets eksamen og føler jeg har truffet greit på alle disse. Jeg har også planer om å videreutvikle applikasjonen min slik at jeg får kanskje lagt den ut på Google Play Store.
Jeg sitter idag med en god kompetanse i Android og kunne lett tenkt meg å jobbe videre med dette i fremtiden.

KILDER

- **Splash Intro, Inspirasjon**: https://www.youtube.com/watch?v=h_hTuaEpc-8
- **Rounded corners, punkt å prikke:** https://www.youtube.com/watch?v=R3qCJQUKVO0
- **8 bit Background for MainActivity** http://www.backgroundcheckall.com/8bit-background-2-2/
- **8bit Background for GameFragment:** http://www.idownloadblog.com/2015/05/10/8-bit-video-game-wallpaper-iphone-ipad/
- **Sound for button:** https://www.youtube.com/watch?v=ZrP-30MmpvM
- **Sound playing on app:** https://www.youtube.com/watch?v=RSi959Xyw-Q&t=114s
- **Song for GameFragment and HighScore:** https://www.youtube.com/watch?v=faNTXJnKCH4
- **Songs for MainActivity:** https://www.youtube.com/watch?v=pL6CNH4t7P4
- **Mushroom red :** https://nathanmarino.deviantart.com/art/8-Bit-Mushroom-281851369
- **Mushroom green:** https://nathanmarino.deviantart.com/art/8-Bit-1Up-Mushroom-281852701
- **Parse Character til en Integer, Følgte dette punkt å prikke:** https://stackoverflow.com/questions/4968323/java-parse-int-value-from-a-char
- **Board picture:** https://commons.wikimedia.org/wiki/File:Tic-tac-toe.png
- **Wallpaper for alertDialog and gameboard pick starter;** https://www.pinterest.com/zamber305/pixel/
- **Alert Dialog, følgte dette punkt å prikke:** https://www.youtube.com/watch?v=plnLs6aST1M
- **Bug problem used this in my sulotion "Click button bug":** https://stackoverflow.com/questions/16662777/how-to-resolve-double-tap-on-button-issue-in-android
- **Updating the DB with player Score, Used the code:** https://stackoverflow.com/questions/9798473/sqlite-in-android-how-to-update-a-specific-row
- **TIMER FOR SPILLET: MYE AV KODEN ER HENTET HERFRA:** https://www.youtube.com/watch?v=Dr-VtCbev10
- **Sjekket litt ut her for hvordan YoutubeSupportFragment fungerer**:https://gist.github.com/takeshiyako2/e776bbaf2966c6501c4f
- **Video for implementering av Youtube API'et:** https://www.youtube.com/watch?v=a4NT5iBFuZs
- **HighScore Wallpaper:** https://www.desktop-background.com/wallpaper/update-new-version-of-the-8bit-day-wallpapers-set-pixel-58843
-
- **HTTP request in android** https://stackoverflow.com/questions/24399294/android-asynctask-to-make-an-http-get-re

quest/24399320?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
- https://alvinalexander.com/android/android-asynctask-http-client-rest-example-tutorial