Minigames

En android app med små reaktion og logik spil

22-08-2018

Christian Alexander Sauer Mark
Studienummer: S164833

DTU Diplom
Center for Diplomingeniøruddannelse

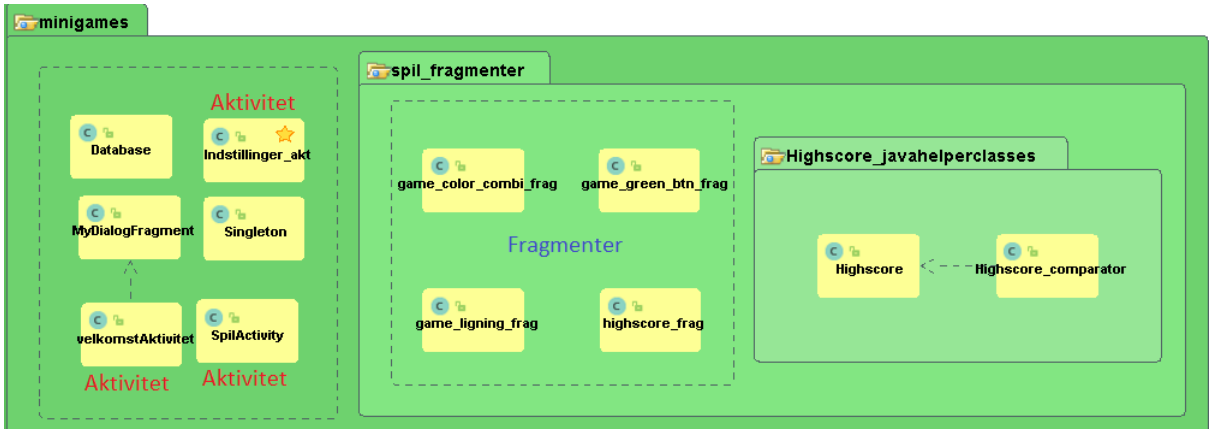
1 Problemformulering

Et reaktions og logik spil ønskes udviklet til telefoner der kører styresystemet Android (Min. Android 5.0 - Lollipop). Spillet skulle bestå af en mængde små opgaver/spil som spilleren på tid skulle løse. Point skal baseres på, hvor mange små del-opgaver spilleren kunne nå at løse på den givne tid. En global og lokal highscore liste skal også implementeres, hvormed app'en skulle kunne håndtere en mængde lokal lagret data og foretage netværkskommunikation.

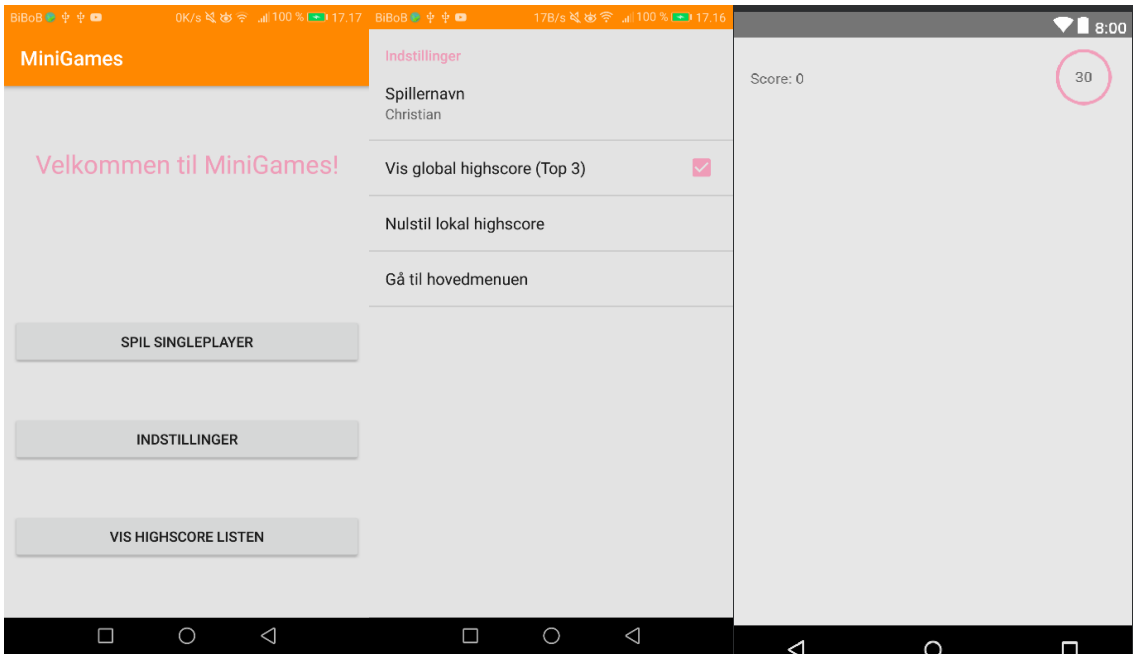
2 Appens design og implementering

1. Appens opbygning

På Figur 1 - "Program struktur" ses java klasserne og opdelingen af aktiviteter og fragmenter. Appen er opbygget af 3 aktiviteter som kan ses nederst på siden.



Figur 1 - "Program struktur"



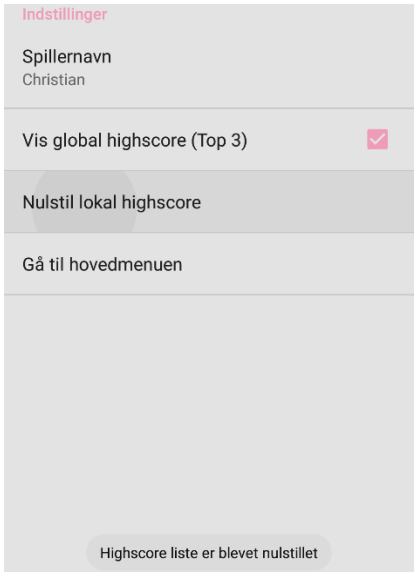
Aktivitet 1 - VelkomstAktivitet *Aktivitet 2 - Indstillinger* *Aktivitet 3 - SpilActivity*

Aktivitet 1 er ”Velkomstaktivitet” som er hovedmenuen og det som spilleren ser først når han åbner appen. Her har spilleren 3 valgmuligheder:

1. Gå til Aktivitet 2 - ”Indstillinger”.
2. Gå til Aktivitet 3 - ”SpilActivity”.
3. Se highscore listen (Beskrevet i ”Highscore - Lokal og Global” på side 6)

Aktivitet 2 er ”Indstillinger” siden, hvor brugeren har 4 muligheder for at ændre på indstillinger i appen:

1. Skifte navn som bruges på den lokale og globale highscore liste.
2. Slå visning og deltagelse på den global highscore til og fra.
3. Nulstille den lokale highscore som er lageret på spillerens telefon (som det kan ses på billedet herunder).
4. Gå tilbage til hovedmenuen.



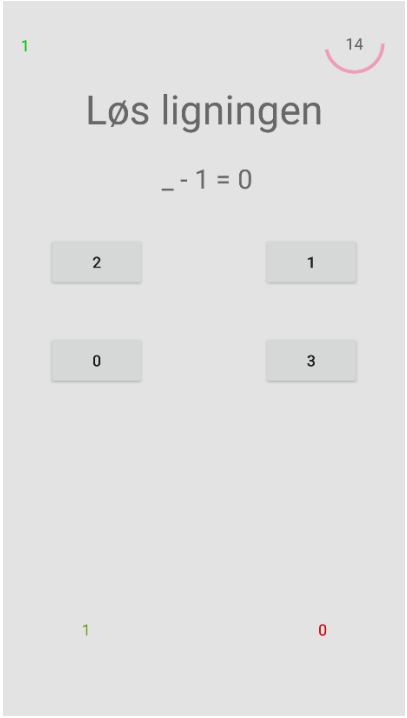
Screenshot 1 - Indstillinger siden, hvor scoren bliver nulstillet

Aktivitet 3 er ”SpilActivity” som er en aktivitet med et framelayout, to ”textviews” og en ”progressbar”. Framelayoutet bruges til at vise de forskellige spil fragmenter som, hver især repræsentere et spil. Det ene textview til venstre viser spillerens score og det til højre viser sammen med progressbaren, hvor meget tid der er tilbage i det nuværende spil.

2. Spillene

I appen er der implementeret 3 små spil som fragmenter. Det første spil er et klassisk matematikspil, hvor der bliver generet nogle tilfældige værdier som sætter en ligning op i et textview og viser det til spilleren. Spilleren skal så hurtigt han kan trykke på den knap der viser det rigtige svar ud af 4 knapper (Se Screenshot 2 - Ligningsløsnings spil på side 4). Spilleren vælger

enten den rigtige eller forkerte (Screenshot 3 eller 4), får visuelt feedback og vil få/miste et point. Først ved rigtigt svare vil der generes en ny ligning og spillet fortsætter indtil tiden løber ud. Specielt for dette spil er, at man kan eksplicit se, hvor mange point og minuspoint man har fået i netop dette spil (røde- og grønne tal i bunden af Screenshot 2,3 og 4). Derudover vises der de to små billeder med kryds eller flueben, som en AsyncTask tråd fjerner igen efter 2 sekunders visning.

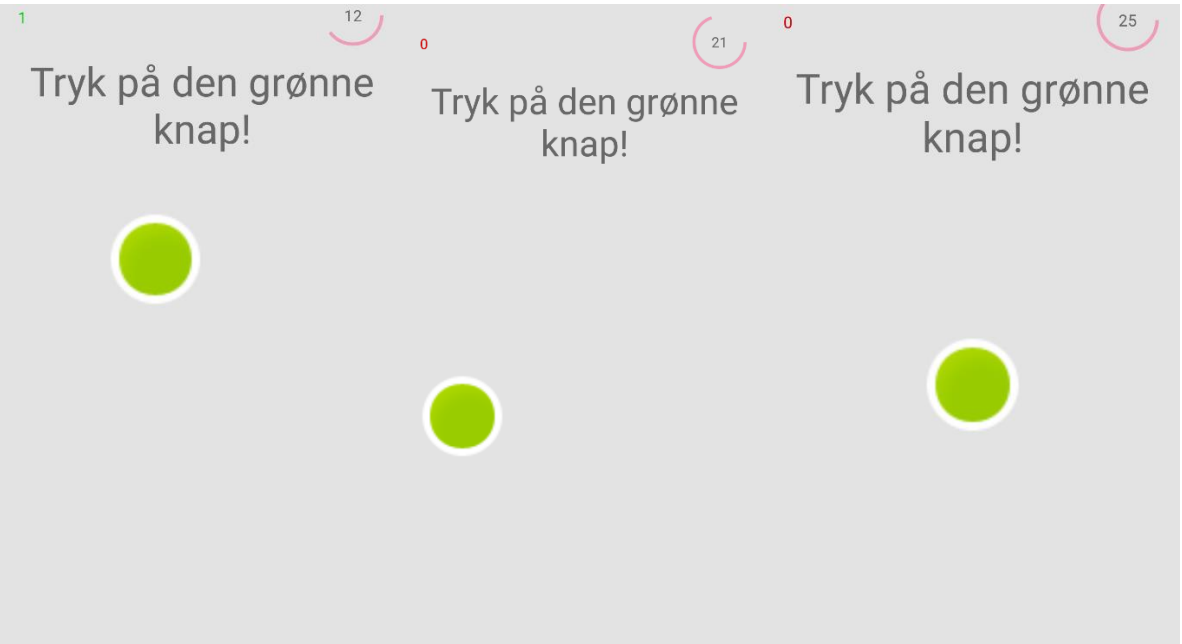


Screenshot 2 - Ligningsløsnings spil

Screenshot 3 - Ligningspil forkert svar

Screenshot 4 - Ligningspil korrekt svar

Det andet spil er et reaktionsspil, hvor en grøn knap - som et imageview - på skærmen skal trykkes på. Ved et tryk på knappen gives et point og ved tryk ethvert andet sted på skærmen vil give 1 minuspoint. Efter ethvert skærmtryk får knappen en ny position som er bestemt udfra størrelse på skærmen i "portræt" positionen (portrait mode)[1] generet med java's random(). Spillet kan ses på "Screenshot 5 - Grøn knap reaktion spil" herunder.

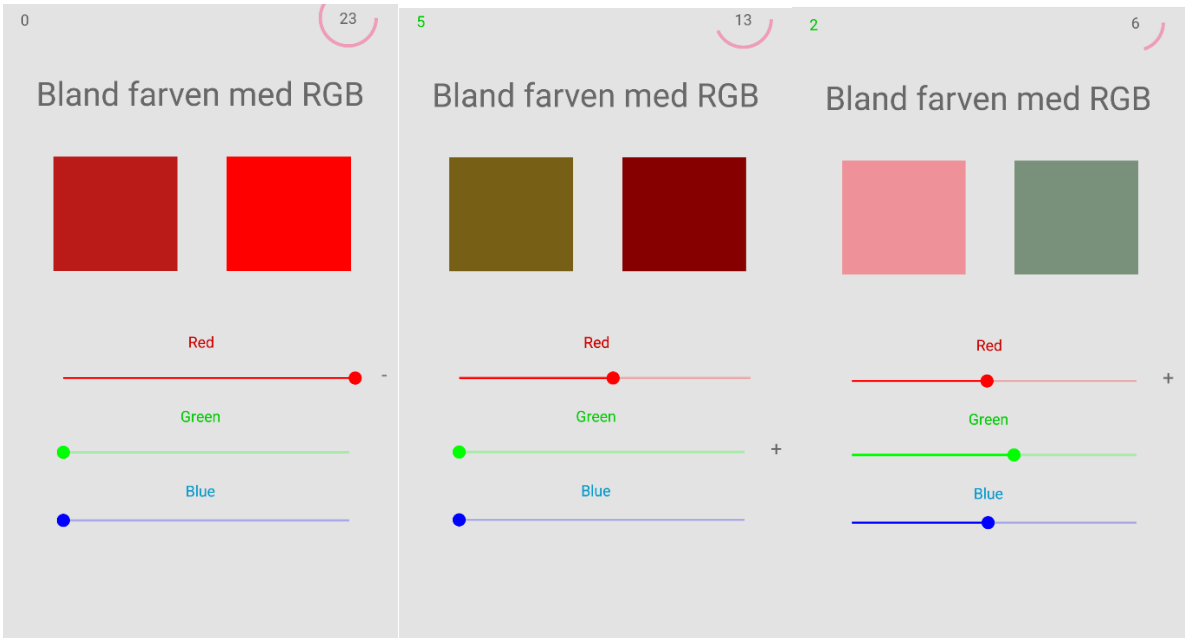


Screenshot 5 - Grøn knap reaktion spil

Det sidste spil er et farvespil som kan ses på Screenshot 6 - Farvespil" på side 6, hvor spilleren får opgivet et Imageview (det til venstre) med en tilfældig RGB farve som baggrund. RGB farven består af 3 farvekanaler (R-G-B) som er tilfældige 8-bit værdier mellem 0 og 255 som bliver generet af random() metoden. Spillere skal så ved hjælp af 3 "seekbars"/slidere kombinere og gætte den rigtige farve som blev generet ved spilstart. Slideren repræsentere hver en af de 3 RGB farvekannaler som styre baggrundsfarven i et andet Imageview, som brugeren skal prøve at få til at matche det første Imageviews's farve. Hver gang at brugeren slipper en af sliderne vil der blive udregnet en "ColourDistance" eller "Colour difference"[2] udfra en bestemt algoritme som kan beskrive, hvor "langt" to RGB-farver er fra hinanden. Denne værdi bruges til at sammenligne farverne, da alternativet var (primitivt) at man opstillede en masse if statements på de 3 RGB int værdier også så om de lå indenfor et vist interval fra hinanden.
Hvis spilleren finder en farve som, ligger under en hvis "farve længde" fra den rigtige farve vil "gættet" blive accepteret og brugeren vil få 5 point. For at spillet ikke skulle være for svært/grænsen hvor man vandt ikke skulle være for "hård" er der visuelt hjælp, hvis brugeren ikke er for langt væk fra farven, men bare mangler den sidste lille justering på en af farvekana-

[1] Se afsnit "Singleton side 9"
[2] https://en.wikipedia.org/wiki/Color_difference

lerne. Et lille ”+” eller ”-” vil angive om der skal skrues op eller ned for den enkelte farvekanal, hvor forskellen mellem den aktuelle farveværdi og den farveværdi som skal findes, er størst.

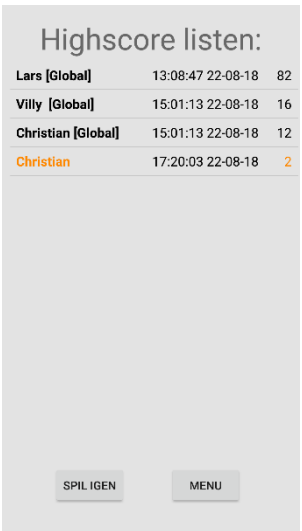


Screenshot 6 - Farvespil

3. Highscore - Lokal og Global

Lokal

Når spilleren har været igennem de 3 spil, vil et highscore fragment blive vist på Screenshot 7 herunder. Dette fragment har et textview til titlen og et listview der viser en ArrayAdapter som viser highscore listen. Highscore listen viser (afhængig af indstillingerne sat af brugeren) først og fremmest ens lokale gemte highscore også den globale highscore liste, hvis dette er slået til.



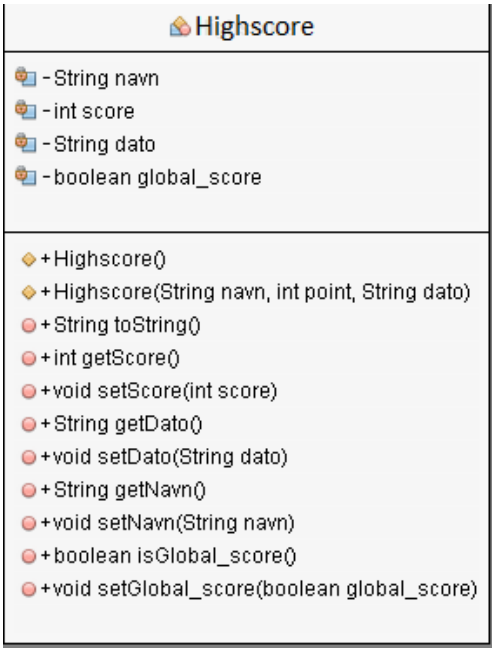
Screenshot 7 - Highscorelisten efter endt spil

Den lokale highscore ligger gemt som nøgle værdi par i PreferenceManageren på android enheden i følgende format:

Type	Nøgle	Retur værdi
String	`"spillernavn_key"`	Spillernavnet brugeren har indtastet*
Int	Spillernavnet	Spillernavnets highscore
StringSet	`"navne"`	Navne i sættet**
String	`"date_"+ Spillernavnet`	Datoen Spillernavnet har sat highscoren

*Da et nøglen i et nøgleværdipar skal være unikt, overskriver "Spillernavnet" kun en ældre score, hvis den nye score er højere end den gamle score.
**StringSet kan indeholde et sæt af strenge, men de ligger usorteret.

Da et Strengsæt er usorteret ligger den lokale highscore altid gemt usorteret og skal sorteres før det vises på highscore listen. Dette gøres ved, at oprette Highscore objekter af typen Highscore.



Figur 2 - UML digram over Highscore klassen

Herefter bliver objekterne indsat i en liste som sortere efter score med Highscore comparator'en. Herefter kan Arrayadapteren få en sorteret highscore liste med navne, dato og score.

Global

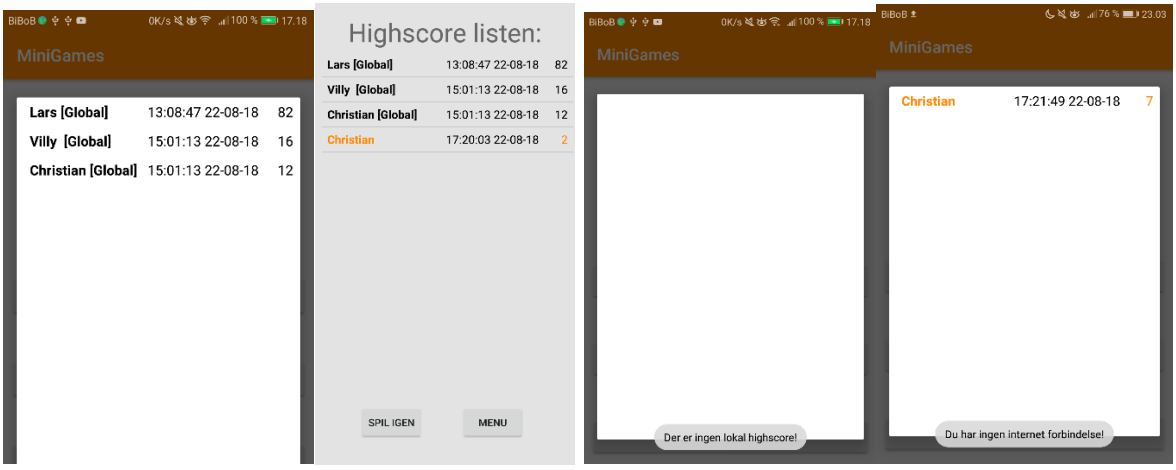
Hvis spilleren har sat hak i "Vis global highscore (Top 3)" på indstillinger siden vil den globale highscore også medtages i visningen af highscoren. Dette gøres ved hjælp af databasen Firebase[3] som det kan ses på Screenshot 8 - Firebase database på side 8. Databasen indeholder 3 Highscore objekter som holder på de 3 højeste globale highscores der er blevet lavet af spillere.

[3] https://firebase.google.com/

Fra databasen bliver der både hentet Highscore objekter når highscore listen skal hentes og sendt, hvis der sættes en ny global highscore.

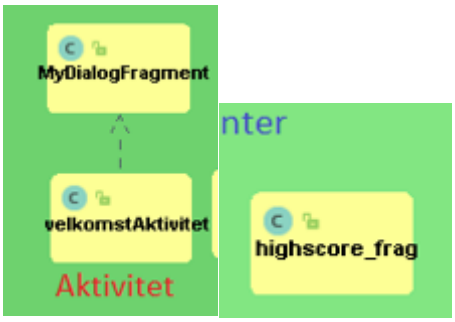
Screenshot 8 - Firebase database

Highscoren bliver hentet og opdateret (hvis ændringer) når en spiller er igennem alle spillene eller spilleren trykker på "VIS HIGHSCORE LISTEN" fra hovedmenuen ("Aktivitet 1 - VelkomstAktivitet"). På "Screenshot 9 - Highscore med lokal og global score vist" ses forskellige scenarier. (Fra venstre) Global highscore vises, men der er ingen lokale highscore lavet endnu. Herefter vises listen når et spil er spillet færdig og en lokal score vises med globalscore. Herefter ses et screenshot, hvor den ikke skal vise globalscore og lokal highscore lige er blevet nulstillet. Til sidst vises, at global highscore skulle vises, men da der ikke er internet, kan den ikke vises.



Screenshot 9 - Highscore med lokal og global score vist

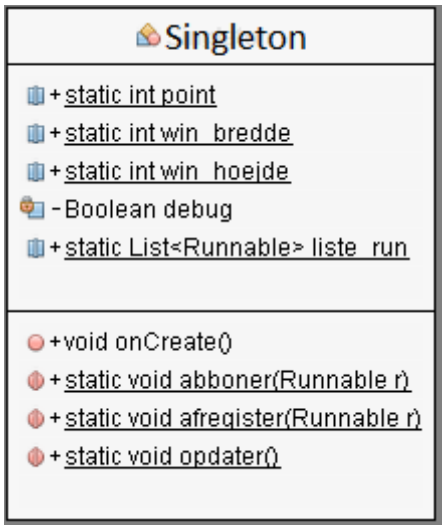
Highscore listen fragmentet bliver vist to forskellige steder på to forskellige måder. Som det kan ses på udklippet fra Figur 1 - ”Program struktur” bruger Aktivitet 1: velkomstAktivitet MyDialogFragment som er et DialogFragment i modsætning til SpilActivity som viser fragmenter og bruger fragmentet highscore_frag til at vise highscore listen.



Screenshot 10 - Udklip fra Figur1

4. Singleton

Hvis man i android gerne vil have noget data alle aktiviteter kan dele (globale variabler) og sikre sig at der kun er én instans af klassen kan man lave det der hedder en Singleton.
Den ”extends” Application og man kan være sikker på, at der altid kun er en af dem i ens applikation. Applikationens singleton indeholder det man kan se på UML diagrammet herunder. ”win_bredde” og ”win_hoejde” er variabler der beskriver højden og bredden på mobilens skærm. Disse variabler sættes i Aktivitet 1 og bruges i grønknap spillet når spilleren trykker ”SPIL SINGLEPLAYER” fra hovedmenuen.

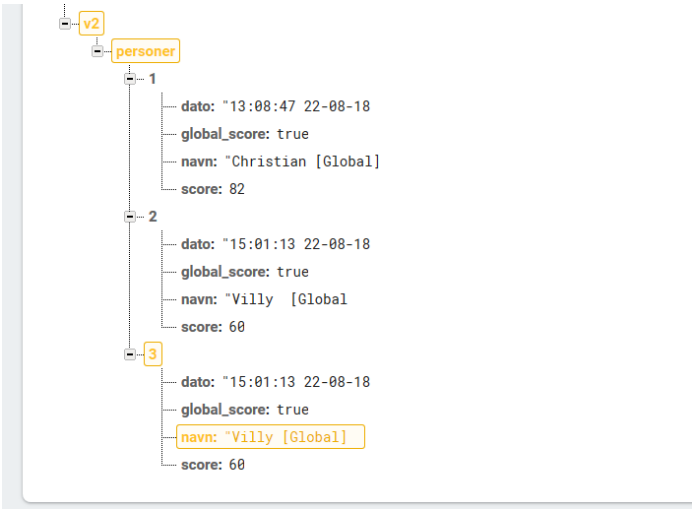


Figur 3 - UML Digram Singleton

3 Afprøvning

Til afprøvning af appen blev android studio's "testmonkey" afprøvet med 10.000 tryk. Desværre blev den ved med at gå ud af appen og begyndte at bruge andre apps og pille ved indstillinger. Så appen er testet ved udlevering af telefon med app installeret til en anden ingeniør studerende/værelseskammeraten. Testpersonen stødte hurtigt ind i første "problem" (se Screenshot 11 - Forsøgt på at spille i landscape mode), at man ikke kan spille spillet i portrait mode. Hvis spilleren prøver at trykke "SPIL SINGLEPLAYER" mens skærmen er vendt forkert, vil et halvt-transparent "vend telefonen"-ikon komme frem på skærmen. Ydermere blev en logik fejl fundet ved den globale highscore, hvor "Villy" med 2. pladsen også vil få 3. pladsen, hvis han spillede et spil igen med samme navn. Dette skyldes den logik der altid prøver, at opdatere den globale highscore ikke tager forbehold for, at den højeste lokale score altid vil bliver uploadet til firebase, hvis den er højere end den i firebase. Eksempel kan ses på Screenshot 12 - Firebase "logikfejl"

Screenshot 11 - Forsøgt på at spille i landscape mode



Screenshot 12 - Firebase "logikfejl"

Spillene blev ellers godkendt og specielt farvespillet var en "unik" lille opgave.

4 Konklusion

I projektet er der brugt meget tid på at arbejde på brugeroplevelsen af app'en og dens potentiale for, at blive udvidet med mange flere små spil. App'en tager både højde for aktiviteternes livscyklus og benytter netværkskommunikation med forbehold for manglende forbindelse. Man kunne dog godt have tilføjet en implementering af noget cache af det data der hentes fra firebase. Logikfejlen beskrevet i "3 Afprøvning" med den globale highscore blev desværre fundet meget sent, så denne fejl er ikke blevet rettet. Implementeringen af lokal highscore liste med PreferenceManageren viste sig også at være meget besværlig og knudret. Hvis man skulle lave det om, vil man nok skrive navne, score og datoer ned i en fil som man så læste og skrev til. Arbejdet med projektet gjorde, at det meste af pensum til kurset er blevet gennemarbejdet eller i hvert fald forsøgt implementeret.