Hiof

TextgameFramework rapport

Dokumentasjon for design

Thomas Alexander Bøhn Eidsvaag
20.05.2020

Tekstspill Rammeverk rapport

Av Thomas Eidsvaag

1. Høynivå beskrivelse av rammeverket

Når vi fikk beskjed om at prosjektet skulle omhandle et rammeverk vi kunne velge fritt, så ville jeg lage et rammeverk som skapte noe interaktivt. Ideen jeg da fikk, var at jeg kunne skape et tekstspill som kunne fungere som et rammeverk. Dette rammeverket ville jeg at man skal kunne ta i bruk uten særlig stort behov for programmeringskunnskaper.

Funksjonaliteten i rammeverket kommer fra at man kan ha et fungerende, kjørende java applikasjon laget i Javafx ved hjelp av objekter man oppretter. Disse objektene blir brukt i rammeverket for å skape scener og knapper som spillet skal ta i bruk. Selve rammeverkets «gameplay» kan bli oppsummert med hvordan «Grøsserne» bøkene fungerer. En leser av en slik bok får valgmuligheter til å gå til spesifikke sider i boken, gjør de riktige valg får de fortsette lesingen. Gjør man feil valg i boken, så ender boken med en dårlig slutt. Dette blir sånn tilsvarende spillmodusen «adventure» blir i mitt rammeverk.

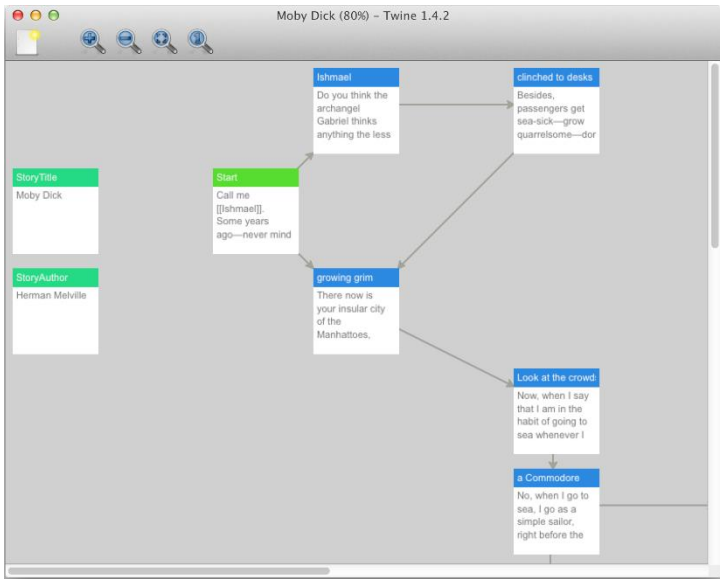
2. Kort beskrivelse av gruppen

Jeg jobber alene med dette prosjektet og jeg har nå fått en god del erfaring med å programmere innenfor Java gjennom min skolegang på Hiof. Grunnen til jeg valgte å basere grensesnittet i JavaFX var fordi jeg lærte å bruke det i emnet objekts-orientert programmering og brukte det aktivt i emne Software Engineering. Tenkte det å bruke et annet programmeringsspråk med et annet grensesnitt rammeverk ville vært altfor tidskrevende. Selv det arbeidet jeg har fått gjort i dette faget har blitt litt strupet av andre emner som fikk mer krevende arbeidskrav etter Corona-viruset.

3. Kort bakgrunn om eksisterende løsninger, og hva de kan og ikke kan

Jeg har søkt meg rundt og prøvd å finne tilsvarende rammeverk laget med Java, men fant ingen. Det jeg fant var en god del små-prosjekter som var veldig hardkodede til å fungere som enkle spill innenfor sin egen kode. Når jeg søkte generelt rundt som ikke var basert på Java, så fant jeg to rammeverk som het «Inform» og «Twine». Disse rammeverkene er kraftige og fleksible, men krever at man lærer deres fastsatte språk innenfor deres rammeverk. Dette synes jeg kan skape en liten læringskurve.

For eksempel Twine så har de en visuell representasjon for scenarioer i spillet. Dette er veldig kult og gir brukeren av rammeverket en god oversikt over hvilket valg en utvikler har lagt ut og hvor de peker til. Bilde her er et eksempel av oversikten i Twine:



Dette er en langt bedre løsning enn det jeg har, for i IDE-er så blir min løsning dårlig. Det blir fort vanskelig å holde en god oversikt over hvor scener og knapper fører til ved å bare se på teksten. Når man tar i bruk mitt rammeverk så må man nesten lage en visuell representasjon over spillet man har lyst til å designe i et annet verktøy.

Så mitt rammeverk er simpel i forhold til deres, men krever langt mindre teknisk forståelse fra brukeren av rammeverket. I tillegg til dette så ser det ikke ut til å være andre rammeverk designet for å lage tekstspill i Java tilgjengelig på nettet.

4. Design prosessen for rammeverket

Definisjoner

FXML
FXML er grunnlaget til grensesnitt laget gjennom JavaFX, som er et grensesnitt rammeverk for Java applikasjoner. Denne FXML-en i mitt rammeverk har felt med FXID-er som blir manipulert av mitt rammeverk, slik at den kan legge inn informasjonen gitt av bruker av rammeverket (som skjer når en utvikler av rammeverket oppretter objekter).

Exceptions
Exceptions er feilmeldinger gitt av Java sin kompilator. Gjennom mitt rammeverk så har jeg satt inn flere exceptions som prøver å fortelle en bruker av rammeverket hva som gikk galt.

Controller
Klasser for JavaFX som manipulerer grensesnittet. Disse controller klassene er basisen for at man kan lage logikk ut ifra grensesnittet som er bygget opp med FXML, som er veldig tilsvarende XML-filer.

EventHandler
En metodetype man kan bruke i JavaFX for å legge til en handling når en spesifikk handling i grensesnittet blir gjort.

Introduksjon

Tekstspill Rammeverket jeg har laget har en lav barriere for å starte å produsere Java applikasjoner for å skape tekstbasert underholdning. Den bruker rammeverket JavaFX for å skape grensesnittet og dette rammeverket bruker FXML-er som et grunnlag for å gjøre dette. API-et kan fungere «rett ut av boksen» ved å importere rammeverket til din foretrukne IDE, for den følger med allerede definerte FXML-er du kan videre konfigurere med CSS. Det jeg hadde veldig lyst til å få til var å også kunne lage selve grensesnittet gjennom rammeverket mitt, men dette ble altfor krevende for meg alene. Hvis jeg skulle fått til det så måtte jeg ha laget et XML-rammeverk for å få til det, for så å designe det rammeverket jeg har laget nå rundt dette. Det å lage et XML-rammeverk for å lage FXML-er er også etter min mening redundant, ettersom JavaFX allerede har verktøy som grafisk kan la en bruker opprette grensesnittet sitt gjennom et program som heter scene builder (og som er pakket inn med IntelliJ).

Scenario drevet design punkter brukt

Scenarioer jeg gikk etter under designing av rammeverket:

1. Bruker av rammeverket skal kunne lage scener som forteller en historie
2. Bruker av rammeverket skal ha muligheten til å ha med knapper i hver scene slik at en spiller kan gjøre handlinger innenfor spillet
3. Bruker av rammeverket skal kunne legge til bakgrunn på spillet
4. Bruker av rammeverket skal ha muligheten til å velge fra forhåndsdefinerte spillmoduser
5. Bruker av rammeverket skal ha muligheten til å implementere sin egen spillmodus
6. Muligheten til å egendefinere små detaljer om hvordan applikasjonen blir representert når den blir kjørt på et operativsystem (f.eks. tittel på vinduet i et operativsystem).

Design prinsipper brukt generelt

- Konsekvent navngiving av klasser og variabler

Beskrivelse:
Jeg har prøvd å konsekvent navngi klasser og variabler for å være beskrivende av funksjonen variabelen/klassen har i rammeverket. Som et eksempel på dette: tidlig i designingen av rammeverket så kalte jeg knapper for «Buttons». Dette var ekstremt forvirrende for meg selv alene, for den hadde flere implementasjoner innenfor JavaFX og skapte en god del problemer. Løsningen for dette var å kalle den «ActionButton». Knapper fikk navnet «ActionButton» fordi den beskriver at det er en knapp og har en handling assosiert med seg innenfor rammeverkets spill komponent.

- Sette «private» eller protected på klasser og variabler som ikke en bruker av rammeverket skal bry seg om

Beskrivelse:
Skjerpe inn hva en bruker av rammeverket har tilgang til, mest for å minimalisere forvirring og feil bruk. Et konkret eksempel er det å sette konstruktøren på objekter til å være private hvis man bruker builder pattern. Når man benytter builder pattern så vil utvikler av rammeverket at alle objekter skal bli laget gjennom builder patternet, og forfatter av rammeverket kan kommunisere hvert felt som skal bli definert for objektet gjennom metode-navn.

- En utvikler av rammeverket skal få konkret og beskrivende tilbakemelding hvis de bruker rammeverket feil.

Beskrivelse:
Kaster exceptions der hvor det er mulig med en beskrivende feilmelding av hva som gikk galt.

Design mønstre brukt

Builder-pattern
Builder-pattern gir en enklere og mer oversiktlig måte å lage nye objekter på.

Singleton
Singleton design mønstret kan brukes når man har et tilfelle av et objekt som bare trenger å instansers en gang.

4.1 Pseudo kode og begrunnelse av scenarioer

1.Bruker av rammeverket skal kunne lage scener som forteller en historie

API for bruker:

```
GameScene.builder()
        .withSceneId(4)
        .withTitle("You've died.")
        .withStory("Your arm just kept bleeding and you've bled out.")
        .buildSceneNoBtns();
```

Kode for å lage en scene uten knapper. Dette punktet er grunnlaget for å fortelle deler av historien basert på valg en spiller gjør.

2.Bruker av rammeverket skal ha muligheten til å ha med knapper i hver scene slik at en spiller kan gjøre handlinger innenfor spillet

API for bruker:

```
ActionButton.builder()
        .withLabel("Go on with life with a rabbit attached to your hand")
        .damagePlayerWillTake(5)
        .pathToASound("")
        .buttonPointsToSceneId(3)
        .build());
```

Kode for å lage en simpel ActionButton. En ActionButton er hovedgrensesnittet som en spiller har for å gjøre valg i applikasjonen. Her gjør de valg basert på hvilken handling de har lyst til å utføre innen konteksten av historien i scenen. Valget knappen står for blir fortalt gjennom knappen sin etikett (label).

3.Bruker av rammeverket skal kunne legge til bakgrunn på spillet

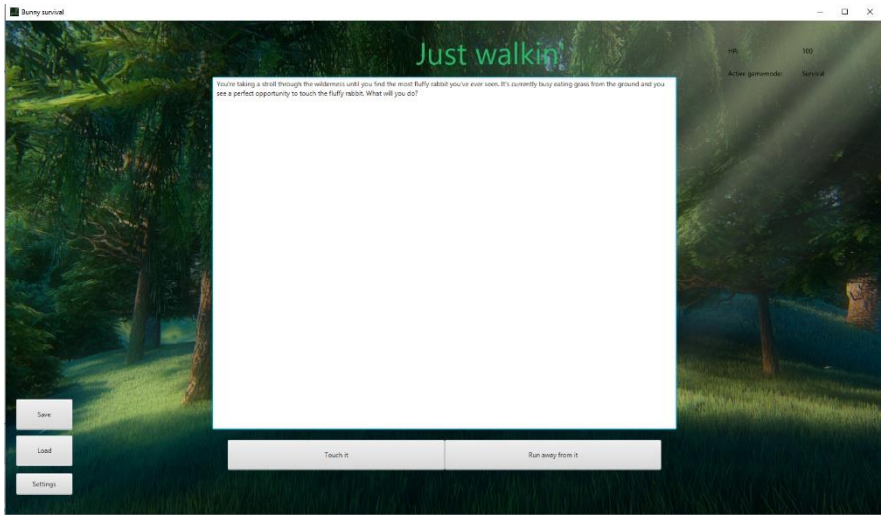
Api for bruker:

GameController.setApplicationBackground(“picture.jpg”);

```
GameSetupDetails.createGameSetupDetails("Survival", "A freaking bunny",
        "An epic tale of epic proportions when you encounter a fluffy bunny while
walking in the wilderness!",
        "forestBG.jpg");
```

Beskrivelse av scenario:
Ideen med dette punktet er veldig simpel. Legge til en bakgrunn på applikasjonen slik at den ikke er grå og trist. Gå fra det her:

Til noe mer fargerikt som det her:



4. Bruker av rammeverket skal ha muligheten til å velge fra forhåndsdefinerte spillmoduser

API for bruker:

Setting av spillmodus blir gjort gjennom GameSetupDetails objektet.

```
GameSetupDetails.createGameSetupDetails("Survival", "A freaking bunny",
        "An epic tale of epic proportions when you encounter a fluffy bunny while
walking in the wilderness!",
        "forestBG.jpg");
```

Beskrivelse:
Første input til denne metodens parameter setter spillmodusen. Originalt så var det planlagt å ha survival (når HP på spiller blir 0 eller mindre så taper man), adventure (Veldig lik grøsserne bøkene, hvor man gjør et dårlig valg, så taper man) og hardcore survival (hvor man ikke har muligheten til å lagre/laste inn framgang). På dette tidspunkt så har jeg bare laget spillmodusen survival. Når HP når null eller lavere så stoppes framgang i spillet og man får en pop-up som forteller at spilleren har dødd.

5. Bruker av rammeverket skal ha muligheten til å implementere sin egen spillmodus

Api for bruker:
@Override
public void customGamemode() {

    doSomething;

}

Beskrivelse:
Et mere avansert punkt. Her så vil jeg at en bruker av rammeverket skal kunne hente og sette variabler i grensesnittet og implementere fritt hva slags regler som skal bli brukt for å definere tap/vinning i sitt spill. Det jeg har tenkt til å gjøre her blir at en utvikler skal kunne hente og manipulere grensesnittet gjennom definerte metoder jeg gjør klart. Hvis GameSetupDetails objektet sitt «GameMode» blir satt til «custom» så skal den ta i bruk en ekstern klasse med en implementasjon for en spillmodus.

For å

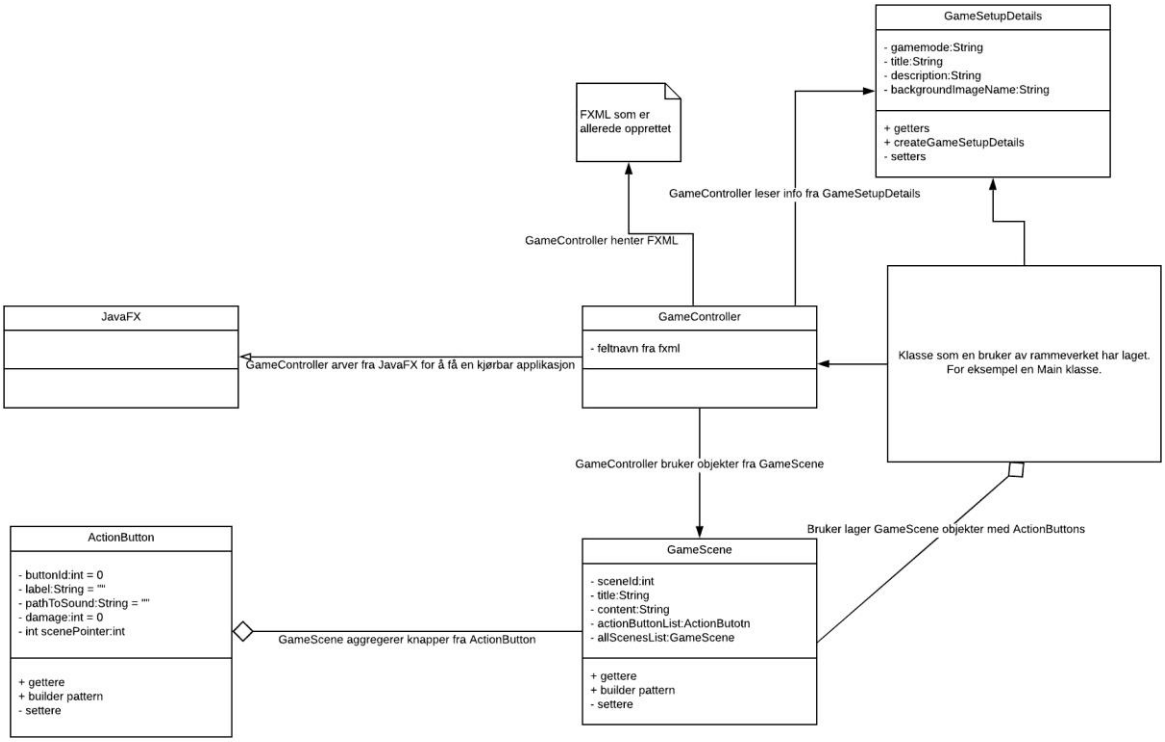
6. Egendefinere små detaljer om hvordan applikasjonen blir representert når den blir kjørt på et operativsystem (f.eks. tittel på vinduet i et operativsystem).

Api for bruker:

GameController.setApplicationTitle(‘’Tittel Her’’);

Beskrivelse:
Ganske simpelt punkt, la en bruker av rammeverk definere tittel på vinduet for applikasjonen som kjører i et operativsystem som Windows.

4.2 Spesifikasjon – UML og beskrivelse av resulterende grensesnitt



En bruker av rammeverket får direkte tilgang til å opprette to forskjellige objekter. UML-en er litt feil ettersom en bruker av rammeverket har full tilgang til å lage en ActionButton direkte, istedenfor gjennom opprettelse av et GameScene objekt. Beste praksis er å lage knapper gjennom opprettelse av GameScenes, men jeg vil at det skal være åpent for å lage ActionButtons til gjenbruk uten å måtte lage en ny scene. Bruker av rammeverket får også tilgang til å definere GameSetupDetails objektet gjennom en metode sin parameter.

GameController arver fra JavaFX sin «application» klasse for å arve funksjonaliteten med å ha en kjørbar applikasjon. FXML følger med i rammeverk prosjektet. GameController får data fra GameScene, ActionButton og GameSetupDetails.

Det resulterende interfacet som bruker kommer til å få kommer for det meste til å være oppretting av objekter som skal bli tatt i bruk av applikasjonen. Dette blir en veldig objekts-orientert design.

4.3 Spesifikasjon – Implementasjonen

Her går jeg gjennom objekter og metoder en utvikler må tenke på når man tar i bruk mitt rammeverk.

GameSetupDetails

Objekt som holder på innstillinger som spillet skal begynne med og hvilken spillmodus som skal være aktivt. Dette objektet er **nødvendig** å definere for at rammeverket skal fungere. Du trenger bare å definere dette objektet en gang og klassen benytter singleton prinsippet for å optimalisere applikasjonen (noe som i praksis ikke har så stort inntrykk på ytelsen).

Felt navn	Type	Beskrivelse
Gamemode	String	Hvilken type spillmodus du vil at spillet skal bruke. I nåværende tilstand så er bare «survival» en implementert spillmodus. Denne spillmodusen vil ta hensyn til hvor mye HP spilleren har, og lukke ut spilleren fra å fortsette spillet når HP-en til spiller blir 0. Andre «godkjente» gamemodes som ikke er implementert er: «hardcoresurvival», «adventure» og «custom».
Title	String	Tittel som dukker opp så fort man starter spillet. Denne tittelen dukker opp over hovedteksten i grensesnittet.
Description	String	Beskrivelse av spillet. Her kan du beskrive generelt hva historien av spillet går ut på.
backgroundImageName	String	Her definerer du et potensielt bilde du vil bruke som bakgrunn av ditt spill. Denne pathen kan være tom, men feltet må fortsatt være med når du lager objektet (hvis du ikke vil ha bakgrunn, lag en tom streng i dette feltet). Du skal bare skrive inn navnet på bildet, og legg det inn i \TextgameFramework\src\main\resources\images. Implementasjonen vil da bruke bildets navn du legger i dette feltet.

Definere GameSetupDetails objektet eksempel

```
GameSetupDetails.createGameSetupDetails("Survival", "A freaking bunny",
        "An epic tale of epic proportions when you encounter a fluffy bunny while
walking in the wilderness!",
        "forestBG.jpg");
```

ActionButton

Objekt som definerer en knapp innenfor rammeverket. En «ActionButton» er musklene av rammeverket. Det er med disse knappene en spiller gjør sine valg i grensesnittet, som skaper hoved essensen av det spillet er basert på. «Builder pattern» har blitt brukt for dette objektet. Dette er gjort for å skille konstruksjonen av et komplekst objekt fra dets representasjon, slik at den samme konstruksjonsprosessen kan skape forskjellige representasjoner. Disse knappene må bli sendt sammen med en gamescene, slik at de blir assosiert med en gamescene.

For å starte lagingen av et ActionButton objekt så må man kalle på «ActionButton.builder()» deretter hvilket felt du har lyst til å definere. Unødvendige felt å ta med er markert med kursivt i felt navn.

Når man har definert alle felt man har lyst til å definere gjennom builder pattern så ender man kallet med «.build();».

Felt navn	Type	Builder kall	Beskrivelse
buttonId	int	.withButtonId(tall)	Kan definere id for knapp. Dette feltet er ikke nødvendig, men hvis du vil så kan du definere dette feltet.
Label	String	.withLabel(''Label'')	Knappens etikett tekst. Dette bør være en beskrivelse av handlingen en spiller gjør ved å trykke på knappen innenfor historien som blir fortalt.
pathToSound	String	.pathToASound(''String'')	Banen til en lyd du vil spille av når knappen trykkes på. Dette er i skrivende stund ikke implementert, så denne har jeg satt slik at man ikke trenger å definere det.
Damage	int	.damagePlayerWillTake(tall)	Hvis spillmodusen går ut på å for eksempel overleve, så er damage feltet der hvor man definerer hvor mye damage en spiller skal ta, når de gjør et valg med en knapp. Survival som er implementert tar denne i betraktning. Hvis man for eksempel har lyst på en type eventyrs modus så kan man bare ignorere dette feltet fordi standard verdi er 0. Gjør dette til et negativt tall så får spilleren HP istedenfor.
scenePointer	Int	.buttonPointsToSceneId(tall)	I dette feltet så skriver man inn sceneId knappen skal peke til. Scenen med id som matcher det man har satt her blir vist når en spiller trykker på knappen. Hvis man har laget to scener med lik id, så blir første scene med denne id-en brukt. Peker knappen til en scene som ikke har blitt laget blir det kastet en exception som forteller en utvikler om det. Dette feltet er nødvendig.

Lage et ActionButton objekt eksempel

```
ActionButton.builder()
        .withButtonId(1)
        .withLabel("Run away from it")
        .damagePlayerWillTake(0)
        .buttonPointsToSceneId(5)
        .build());
```

GameScene

Dette objektet holder historien og knappene som grensesnittet skal fylles ut med. GameScene klassen bruker også builder pattern. Dette objektet må bli laget med knapper slik at grensesnittet vet hvilken knapper som skal bli brukt for scenen. Eneste GameScene du må lage med en spesifikk id er scenen du vil skal være start GameScenen. Din start GameScene **må** ha sceneid 1. Dette objektet bruker også «builder pattern» av samme grunn som en action button.

For å starte lagingen av et GameScene objekt så må man kalle på «GameScene.builder()» deretter hvilket felt du har lyst til å definere. Unødvendige felt å ta med, er markert med kursivt i felt navn.

Felt navn	**Type**	**Builder kall**	**Beskrivelse**
sceneId	Int	.withSceneId(tall)	Id-en til scenen. Dette er identifikatoren for hver enkelt scene. For at scenen skal bli vist så må en knapp peke til denne scenen i en ActionButton sin «scenePointer». Nødvendig. Dette feltet må være unikt for hver scene.
Title	String	.withTitle(''String'')	Tittelen på scenen (stor tekst over hovedteksten i grensesnittet). Kan være tom, så ikke nødvendig.
Content	String	.withStory(''String'')	Hovedteksten, her skal selve historien for scenen være. Dette blir brukt i det store tekstfeltet i grensesnittet. Kan være tom, men dette er ikke anbefalt for hovedteksten skaper konteksten av en scene
actionButtonList	ArrayList<ActionButton>		Dette er en liste med knapper som blir laget automatisk av rammeverket når en scene blir laget med knapper. Denne trenger ikke en utvikler å tenke på.
allScenesList	ArrayList<GameScene>		Liste med alle scener laget gjennom rammeverket. En utvikler trenger ikke å tenke på dette feltet.

Her måtte jeg designe klassen annerledes. Jeg ville at man skal kunne velge fritt mellom å ha 0-4 knapper. Dette gjorde jeg ved å overlaste konstruktøren gjennom builder pattern. For å lage en GameScene med et spesifikt antall knapper, så må man bruke metoden for eksempel «buildSceneTwoBtns(2 ActionButtons her)» for å lage to knapper. Ha med fire knapper med en GameScene så må man bruke metoden «buildSceneFourBtns(4 ActionButtons her)».

Lage en GameScene med ingen knapper:

```
GameScene.builder()
        .withSceneId(5)
        .withTitle("Ran away")
        .withStory("You bravely ran away from the rabbit. Your quick thinking
clearly deduced that this " +
                "might've been a reference to the scary rabbit from Monthy
Python.")
        .buildSceneNoBtns();
```

Lage en GameScene med to knapper:

```
GameScene.builder()
        .withSceneId(1)
        .withTitle("Just walkin'")
        .withStory("You're taking a stroll through the wilderness until you find
the most fluffy" +
                " rabbit you've ever seen. It's currently busy eating grass from
the ground and you see a perfect" +
                " opportunity to touch the fluffy rabbit. What will you do?")
        .buildSceneTwoBtns(
                ActionButton.builder()
                        .withLabel("Touch it")
                        .damagePlayerWillTake(10)
                        .buttonPointsToSceneId(2)
                        .build(),
                ActionButton.builder()
                        .withLabel("Run away from it")
                        .damagePlayerWillTake(0)
                        .buttonPointsToSceneId(5)
                        .build());
```

Metoder for å egen definere smådetaljer på applikasjonen

Ikke en ideell løsning, men fikk lagt til et par metoder for å egendefinere ikonet og tittelen på applikasjonen som blir brukt i operativsystemer.

Sette tittel på applikasjonen:

```
GameController.setApplicationitle("Bunny survival");
```

Sette ikon på applikasjonen:

```
GameController.setApplicationIcon("navnetPåBildetHerSomLiggerIResourceMappen.jpg")
;
```

Denne metoden henter bilder fra samme mappe som GameSetupDetails, som er \TextgameFramework\src\main\resources\images\.

Starte programmet

I enden av klassen du skal kjøre applikasjonen fra, så må man kjøre denne metoden:

```
GameController.runGame();
```

Når denne metoden blir kjørt, så blir alle objekter tatt i bruk av applikasjonen automatisk. Denne setningen er **nødvendig** for at applikasjonen skal bli kjørt. Kode satt etter denne koden blir ikke brukt av rammeverket.

4.4 Bruker testing

Testingen foregikk noen dager før innlevering. Ville helst ikke plage andre grupper som er i samme emne i «crunch-tiden» deres, fordi jeg var treg med å komme i gang med dette punktet. Derfor valgte jeg å finne to andre som ikke har gått gjennom samme utdanningsløpet jeg har gjort.

Test case 1, erfaren Java utvikler

Jeg skjenner en person som har sikret seg en fast jobb innenfor programmerings bransjen som en Java utvikler etter sin skolegang, og sitter for tiden med sin master. Jeg fikk han til å importere mitt rammeverk og han fikk til å gjøre så og si alle handlinger som rammeverket har å tilby:

1. Bruker av rammeverket skal kunne lage scener som forteller en historie. ✓
2. Bruker av rammeverket skal ha muligheten til å ha med knapper i hver scene slik at en spiller kan gjøre handlinger innenfor spillet ✓
3. Bruker av rammeverket skal kunne legge til bakgrunn på spillet ✓
4. Bruker av rammeverket skal ha muligheten til å velge fra forhåndsdefinerte spillmoduser ✓
5. Bruker av rammeverket skal ha muligheten til å implementere sin egen spillmodus ✗
6. Muligheten til å egendefinere små detaljer om hvordan applikasjonen blir representert når den blir kjørt på et operativsystem (f.eks. tittel på vinduet i et operativsystem). ✓

Hvorfor punkt 5 av mine scenario-drevet design punkter feilet går jeg i mer detalj i punkt 5 i dokumentet hvor jeg presenterer den endelige koden for rammeverket.

Han hadde ikke noe konkret tilbakemelding å gi meg dessverre, men fint at han klarte å ta i bruk rammeverket.

Test case 2, min far

Rammeverket mitt er etter min mening relativ simpel å ta i bruk. For å bevise dette, så skal jeg gjøre det utenkelige. La min far prøve å lage et spill ut ifra mitt rammeverk. Han er en kuldeingeniør og har så å si ingen programmeringserfaring. Det jeg har tenkt til å gjøre er å la han få lese dokumentasjonen, og ikke lære han direkte hvordan rammeverket skal brukes. I enden av dette dokumentet så har jeg også lagt til en bruksanvisning til hvordan man setter opp et nytt prosjekt i IntelliJ for å ta i bruk rammeverket.

Jeg satt meg ned med min far etter å ha installert IntelliJ Community Edition på hans laptop. Han leste først instruksene på hvordan man setter opp IntelliJ for å ta i bruk rammeverket (det er i enden av dokumentet). Dette gikk smertefritt.

Når det kom til å faktisk bruke rammeverket så var den tekstlige beskrivelsen i dette dokumentet vanskelig for han å forstå (noe som er forståelig, mange fremmedord hvis man aldri har holdt på med programmering). Det som virkelig hjalp han med å komme i gang, var å se på hvordan min eksempelkode var lagt opp i «sample» pakken. Da forstod han sammenhengen og lagde sitt eget lille spill, med alle scenarioene jeg har satt opp (unntatt scenario-drevet design punkt 5).

Et sted hvor han ga meg direkte tilbakemelding var presentasjonen av applikasjonen sin tekst. Han synes at teksten var altfor liten, så jeg økte tekst størrelsen. Jeg lagde deretter en metode en utvikler kan kalle for å endre tekststørrelsen selv gjennom interfacet.

5. Presenter den endelige koden for rammeverket

Istedenfor å legge til koden på alle punkter, så vil jeg referere til linjer av koden eller hvilket filer hvor implementasjonene ligger for de forskjellige scenarioene. Noen av disse punktene ble voldsomt store og ville ikke bli oversiktlig innen dette dokumentet.

1.Bruker av rammeverket skal kunne lage scener som forteller en historie

2.Bruker av rammeverket skal ha muligheten til å ha med knapper i hver scene slik at en spiller kan gjøre handlinger innenfor spillet

Disse to scenario-drevet design punktene ble smeltet sammen når jeg skulle lage implementasjonen. For at en bruker av rammeverket skulle på lettest mulig skape en sammenheng mellom knapper og en scene, så var den alternative måten jeg kom på at bruker av rammeverket måtte lage knapper og legge disse knappene manuelt inn i et ArrayList, for så å sende denne listen med som parameter med for å lage en ny GameScene. Dette hadde vært unødvendig komplisert å gjøre for hver enkelt scene som skal bli laget. Selv om det resulterende API-et jeg har laget for å opprette en GameScene med ActionButtons kan være litt innviklet, så synes jeg dette var den simpleste interface løsningen jeg kunne laget.

Builder pattern ble brukt på både «GameScene» og «ActionButton». Dette muliggjorde å lage beskrivende kall for felt innenfor for å lage et nytt objekt. I tillegg til dette så forhindrer builder patternet at en bruker av rammeverket må definere hvert nye objekt som skal bli laget med «new». Denne implementasjonen kan du se i «Models/GameScene» og «Models/ActionButton».

Selve implementasjonen for logikken for å bruke GameScener og ActionButtons ligger i klassen «GameController». På linje 172 så starter metoden «showFirstActiveScene». Denne blir kjørt én gang så fort en spiller trykker på «Start» knappen i applikasjonen. Her blir GameScene med id 1 brukt først. Dette vil si at en utvikler som bruker rammeverket må definere en GameScene med id 1 som skal være starten av historien. Dette kunne vært bedre definert i interfacet, men fant ingen direkte løsning som ikke var overkomplisert.

Linje 189-199 i GameController ligger metoden «findActiveScene». Denne metoden iterer gjennom alle GameScenes som har blitt laget for å finne aktive scenen som skal bli vist neste og returnerer den. Blir ikke denne scenen funnet så returner metoden en exception som forteller hvilken scene metoden fikk beskjed om å finne, og at denne ikke eksisterer.

Linje 206-339 har metoden «enableActionButtonsForScene». Denne metoden burde fått noen «refractoringer» for å splitte opp funksjonaliteten. Dette er metoden som lager «spillet». Metoden printer tekst til grensesnittet basert på hvilken sceneId den får sendt som parameter. Når det er gjort så aktiverer den knapper basert på hvor mange knapper scenen har. Hver av disse knappene har en «eventHandler» som kjører metoden «enableActionButtonsForScene» for knappen som blir trykket på sin scene peker.

3.Bruker av rammeverket skal kunne legge til bakgrunn på spillet

Implementert ved å ha et stort «ImageView» som er et felt innenfor JavaFX grensesnitt hvor man kan vise bilder. Burde i grunn vært at man kan definere path til et bilde i GameSetupDetails objektet, men synes denne løsningen var mer oversiktlig og lettere for en bruker å ta i bruk for å definere bilder. Det en utvikler som bruker rammeverket må være obs på er å ha lik mappestruktur som

rammeverket krever. Senere i dokumentet ligger en beskrivelse på hvordan man setter opp strukturen i prosjektet.

```
private void setBackgroundImage() throws ImageNotFoundException {

    String backgroundImageName = gsd.getBackgroundImageName();

    if(!backgroundImageName.equals("")) {
        try {
            FileInputStream input = new
FileInputStream("src/main/resources/images/" + backgroundImageName + "/");
            Image image = new Image(input);
            backgroundImg.setImage(image);
        } catch (FileNotFoundException e) {
            throw new ImageNotFoundException("The image you've specified in your
GameSetupDetails wasn't found");
        }
    }
}
```

4.Bruker av rammeverket skal ha muligheten til å velge fra forhåndsdefinerte spillmoduser

Dette blir gjort gjennom defineringen av GameSetupDetails objektet. Når man setter «Gamemode» til å være «survival» som er en implementert spillmodus, da blir «setGamemodeSurvival()» metoden kjørt.

Fra linje 345 til 437 i klassen «GameController» så ligger denne implementasjonen. Enderesultatet fra gamemode metodene er at det blir satt en «changeListener» satt på HP. Hver gang HP-en endrer seg, så sjekker denne «changeListener» metoden om HP er 0 eller lavere.

5.Bruker av rammeverket skal ha muligheten til å implementere sin egen spillmodus

Dette punktet ble jeg lei meg for at jeg ikke fikk til. Dette kommer av at JavaFX sine felt innenfor grensesnittet ikke kan være «static». Kunne ikke gjøre det mulig for at man kunne overkjøre en metode fra «GameController» klassen gjennom arving heller, for da må man reimplementere en kjørbar applikasjon i den klassen en utvikler hadde ønsket å lage en ny spillmodus. For å ha fått dette til å fungere så måtte jeg ha redesignet hele prosjektet og det hadde jeg ikke tid til.

Interfacet ville vært det jeg lagde i «CustomGamemode» klassen som ligger i pakken «sample» i prosjektet. Jeg laget settere og gettere for feltene som blir brukt i applikasjonen i «GameController» klassen, slik at man kunne arve klassen «GameController» for å manipulere grensesnittet med en utviklers eget spillmodus.

6.Egendefinere små detaljer om hvordan applikasjonen blir representert når den blir kjørt på et operativsystem (f.eks tittel på vinduet i et operativsystem).

Legge til tittel på en JavaFX applikasjon var relativt simpelt å implementere, men jeg liker at en bruker av rammeverket kan definere dette for å legge til sitt eget preg på sitt prosjekt.

```
public static void setApplicationTitle(String title) {
    windowTitle = title;
}
```

Det å la en bruker legge til applikasjonsikon var litt mer krevende, men her gjenbrukte jeg en del av koden fra det å sette bakgrunnsbilde på applikasjonen:

```
public static Image setApplicationIcon(String image) {
    windowIcon = image;

    //If empty, don't do anything
    if (!windowIcon.equals("")) {
        FileInputStream input = null;
        try {
            input = new FileInputStream("src/main/resources/images/" + image +
"/");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        Image icon = new Image(input);

        return icon;
    }
    return null;
}
```

I denne metoden blir samme filbane som det scenario-drevet design punktet 3 brukt igjen.

7.Endre tekststørrelse på hovedtekst og knapp etiketter i grensesnittet gjennom interfacet

Dette er et ekstra punkt som jeg la til i listen for scenario-drevet design på grunn av tilbakemelding fra min far. Jeg vil at man skal kunne definere tekst størrelse gjennom mitt API. Implementasjonen var litt vrien å få til på grunn av at en annen klasse kan ikke direkte manipulere et element innenfor JavaFX som ikke er en «controller» av FXML-en.

Når metoden blir kalt, så setter metoden «mainTextSize» til å være størrelsen bruker har satt.

```
public static void setTextSizeMainText(int size) {
    mainTextSize = size;
}
```

Deretter for at tekstørrelsen skal bli satt, så kjører metoden «setAllSettings». Denne kjører etter at selve applikasjonen starter. Hadde jeg brukt samme framgangsmåte som med punkt 6, så hadde den produsert «null pointer exception».

```
private void setAllSettings() throws GamemodeNotFoundException,
ImageNotFoundException {
    setBackgroundImage();
    setGameMode();
    //Sets the font size
    mainText.setFont(Font.font(mainTextSize));
    actionBtn1.setFont(Font.font(actionButtonLabelTextSize));
    actionBtn2.setFont(Font.font(actionButtonLabelTextSize));
    actionBtn3.setFont(Font.font(actionButtonLabelTextSize));
    actionBtn4.setFont(Font.font(actionButtonLabelTextSize));
}
```

Hvis ikke disse blir metodene blir satt av utvikler, så blir standard verdi 16 brukt som tekst størrelse.

6.Konklusjon

Som konklusjon så er jeg fornøyd med det jeg har fått til med prosjektet. Design-driven scenarioet som var å la en bruker av rammeverket kunne definere sin egen spillmodus, plager meg at jeg ikke fikk til. Hadde jeg fått det til så ville jeg ikke hatt noen problem med å kalle dette et ordentlig rammeverk. Akkurat nå så blir den veldig simpel, men funksjonell. Det som ikke er så positivt er at det kan bli veldig lange parameter strenger sendt gjennom lagingen av nye objekter, så lenge man skal lage en historie ut av objektene man lager. I retrospekt så burde jeg tatt noe simplere, det ble litt komplisert å lage et rammeverk rundt et grensesnitts rammeverk.

«GameSetupDetails» objektet som brukte «Singleton» prinsippet ble litt rar i lengden. Ideen var å ha et objekt som skulle holde på viktige egenskaper pakket inn i et objekt, slik at en bruker av rammeverket ikke måtte gå gjennom en smørbrødliste for hva de må definere på forhånd. Så lenge det ble pakket inn i et objekt så kom det tydelig frem hva en utvikler må definere for å få applikasjonen til å kjøre «riktig». Singleton prinsippet i praksis med GameSetupDetails objektet ble nærmest et anti-pattern, fordi det krever et eksplisitt kall til klassen for å definere de forskjellige feltene på GameSetupDetails objektet, og skaper mer forvirring enn det Singleton-patternet er verdt.

For å argumentere for at dette er et rammeverk framfor «konfigurerbar programvare» så er det jeg har laget her et system designet rundt å ta i bruk en kjørbar applikasjon som jeg har designet selv. Hadde rammeverket vært bestående av «setDitt» og «setDatt» på applikasjonen gjennom interfacet som hovedfunksjonalitet, så hadde dette gått under kategorien «konfigurerbar programvare».

Sette opp IntelliJ for å ta i bruk rammeverket

Maven er grunnstrukturen av rammeverket, mest for at den kan automatisk innhente avhengigheter fra JavaFX automatisk. Det ble ikke tatt hensyn til hvordan mappestrukturen ble før det var for sent, men det her er letteste måte å kopiere rammeverket over til nytt annet prosjekt som jeg fant.

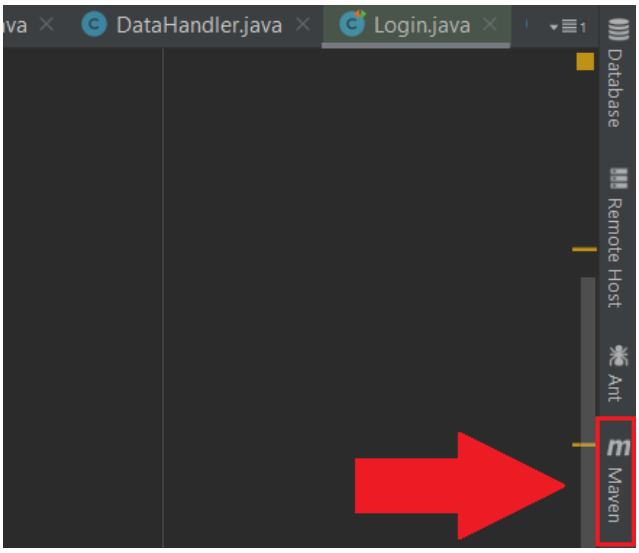
1. I IntelliJ, trykk på file og deretter mus over «new» og trykk «project».
2. Velg Maven fra listen på venstre side, velg Java versjon 11.0.4 hvis du har den installert (IntelliJ kan laste ned nyeste Java SDK for deg her, prøvde å kjøre rammeverket med SDK 14.0.2 og det fungerte fint) så trykk «next».
3. Skriv inn din foretrukne «groupid», «artifactid» og «version». Trykk «next».
4. Skriv prosjekt navn du vil ha og definer hvor du vil at prosjektet skal bli lagret på din PC.
5. Når du får opp prosjektet ditt, finn din «pom.xml» som er i roten av prosjektet og åpne den. Her skal du kopiere inn avhengigheter satt i rammeverket mitt. Finn «pom.xml» som ligger i rammeverkets filer som ligger i roten av «TextgameFramework» og kopier alt fra linje 11 til enden. (Pass på at du ikke får dobbelt opp med </project> i filen din).
6. Nå skal du legge til klasse filene av rammeverket til ditt prosjekt. Banen er TextgameFramework/src/main/java/. Her skal du kopiere mappen "TextgameFrameworkClasses" til din «java» mappe som ligger i ditt prosjekt.
7. Her skal du legge inn ressurser og det som kommer til å bli mappestrukturen på bilder du vil bruke i ditt prosjekt gjennom rammeverket. Du skal kopiere innholdet til mappen som befinner seg på denne banen: TextgameFramework/src/main/resources/. Disse to mappene skal du kopiere til ditt prosjekt sin «resources» mappe.
8. Forsikre deg om at Maven har lastet inn avhengigheter som ble lagt inn i din «pom.xml» fil ved å trykke på «Maven» på høyre side av IntelliJ. Når dette er gjort så får du et ekstra vindu. Her trykker du på «refresh» knappen (I enden av dokumentet så har jeg lagt til hvordan man gjør dette i mer detalj).
9. For å teste om alt fungerer så kan du prøve å kjøre «SampleMain» klassen, som er et eksempel på hvordan rammeverket kan bli brukt. Dette gjør du ved å finne «SampleMain» i ditt prosjekt i IntelliJ, som befinner seg i filbanen src/main/java/TextgameFrameworkClasses/sample/. Høyreklikk på «SampleMain» og trykk run. Forhåpentligvis nå så kjører applikasjonen.

For å bruke en annen Main fil å kjøre rammeverket fra, så tar du bare å høyre klikker på din egen «Main» fil og trykker «Run (dinKlasse)».

Reimportere avhengigheter i Maven

Hvis Maven ikke henter avhengighetene automatisk, så må du få den til å oppdatere manuelt. Da må du:

1. Gå til høyre side av IntelliJ, så skal Maven være langs høyre side. Kan hende Maven er litt seig og ikke vil dukke opp med mindre du starter IntelliJ på nytt.



2. Trykk på Maven vist på bildet over og trykk på «Reimport All Maven Projects» som har symbolet av to piler som går i en sirkel (som vist på bilde under).