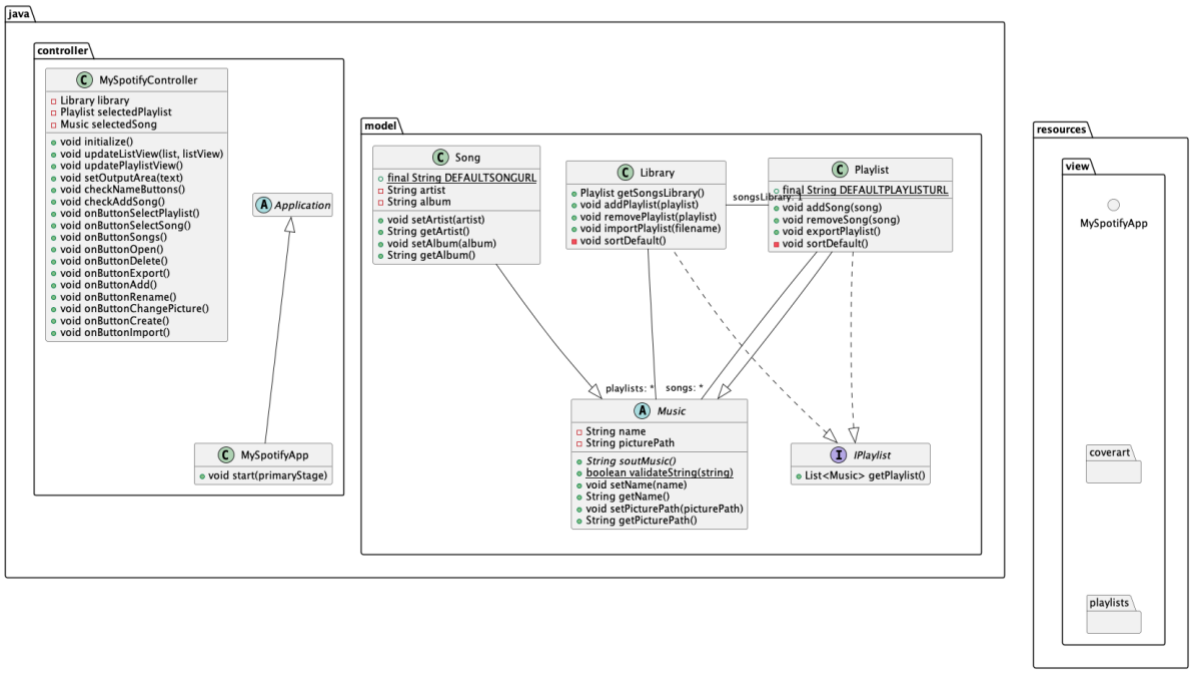
Dokumentasjon

Min applikasjon skal etterligne Spotify. Brukeren skal ha mulighet til å kunne velge fra en stor mengde med forhåndsbestemte sanger ("Song"-objekter), som de deretter skal kunne legge inn i sine egne spillelister ("Playlist"-objekter). Dersom man trykker på ulike knapper i applikasjonen, skal ulike metoder fra de respektive klassene kalles. Dette inkluderer muligheter som å kunne lage nye spillelister, slette spillelister, eksportere spillelister, importere spillelister, gi spillelister nytt navn, endre forsidebilde på spillelister og legge til sanger i spillelister.



Klassediagrammet viser hvordan applikasjonen er satt opp. Merk at mens «controller»-mappen og «model»-mappen er inneholdt i «java»-mappen, er FXML-filen «MySpotifyController» plassert i «resources/view». Dette er for å følge «Model-View-Controller»-prinsippet. I tillegg, inneholder både «Library»-klassen og «Playlist»-klassen en liste med «Music»-objekter. Det siste som er verdt å merke seg er at «Library» også inneholder et «Playlist»-objekt «songsLibrary». En liten tilleggsopplysning er at «MySpotifyController»-klassen også inneholder utallige «FXML»-felter, men disse er ikke en del av diagrammet. Dette er for å spare plass.

Mange deler av pensum brukes. Den abstrakte klassen «Music» blir arvet av både «Song»-klassen og «Playlist»-klassen («extends Music»). Dette er fordi det ikke gir noe mening å ha

mulighet til å instansiere «Music»-klassen, i tillegg til at implementasjonen er lik hos begge subklassene. En annen del av pensum som dekkes er grensesnitt gjennom «IPlaylist». Både «Playlist»-klassen og «Library»-klassen implementerer dette grensesnittet («implements IPlaylist»). Dette er for å kunne kalle på begge klassene om hverandre, siden begge inneholder en type spilleliste, der «Playlist» inneholder «Song»-objekter mens «Library» inneholder «Playlist»-objekter. «IPlaylist» er ikke et funksjonelt grensesnitt selv om det kun inneholder én metode, fordi det tar hensyn til intern tilstand. I tillegg er korrekt innkapsling brukt gjennom hele applikasjonen. Metoder er, med enkelte unntak, «public», mens alle felter er «private». Dette inkluderer feltene i superklassen «Music» og FXML-feltene i «MySpotifyController». Selv om både «Song»-objekter og «Playlist»-objekter sorteres i applikasjonen, arver ingen av de fra «Comparable»-superklassen. Istedenfor, sorteres de gjennom metoden «sortDefault», som utnytter lambda-uttrykk for å sortere. Derfor finnes det heller ingen metoder som implementerer «Comparator»-grensesnittet.

Det brukes ikke observatør-observert-prinsippet i min applikasjon. Dette kunne blitt brukt ved å si ifra til observatørene hver gang en sang ble lagt til eller fjernet fra en spesifikk spilleliste. Grunnen til at jeg valgte å ikke inkludere dette var fordi dette hadde gjort applikasjonen enda mer komplisert. I tillegg hadde det ikke vært veldig givende for brukeren. I tillegg brukes ikke «Iterator» eller «Iterable» i applikasjonen, da det aldri var noe nødvendighet for det i noen av funksjonene som appen skulle ha. Til slutt ble det heller ikke brukt delegering i applikasjonen. Dette kunne blitt gjennomført ved å, for eksempel, ha en slags «queue» med sanger som skulle spilles av, hvor flere spillelister delegerte noen sanger til denne.

Koden forholder seg fint til «Model-View-Controller»-prinsippet sine standarder. Personlig, brukte jeg mye tid på å finne relativ filsti for klasser gjennom min utvikling av applikasjonen.

Jeg har valgt å teste alle delene av applikasjonen min for å sørge for korrekt tilstand til alle mulige objekter. Når jeg har utviklet disse testene, har jeg tenkt ut de fleste tingene som kan gå galt, og sørge for at disse utløser nødvendige unntak, slik at applikasjonen kan håndtere disse. Enkelte ting som ble testet ga feil resultat, slik at jeg kunne finne ut hva som førte til at feil tilstand ble oppnådd. Noe jeg ikke tester er innholdet i eksporterte filer i testene for «Playlist»-klassen, men jeg tester på den andre siden om det som blir importert er korrekt gjennom «Library»-klassen. Dette har ført til at dersom «exportPlaylist»-metoden ikke fungerer som den skal, kommer aldri testene til «importPlaylist»-metoden å bli oppnådd.