Dokumentasjon Snake

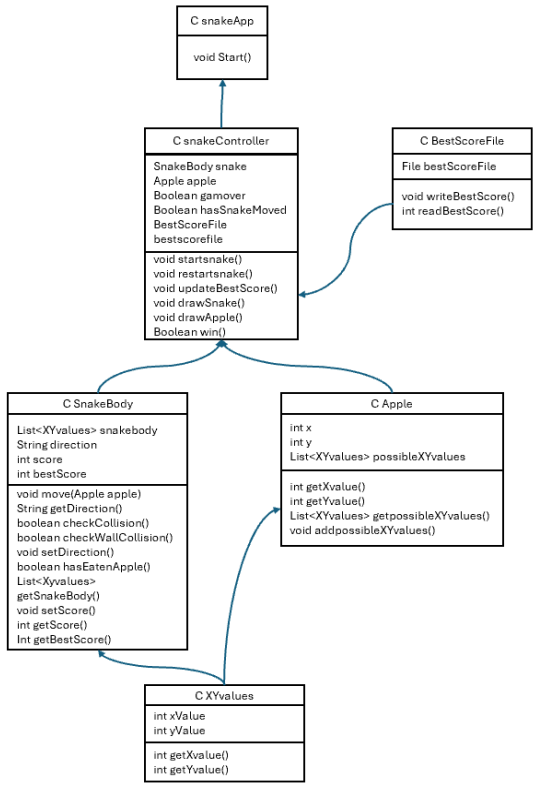
av Birk Thomassen

Beskrivelse av appen

For å gjøre dette prosjektet så spennende som mulig og passe utfordrende valgte jeg å lage et spill. Spillet jeg laget var snake som etterligner det klassiske Snake-spillet, men er noe forenklet. For å starte spillet trykker brukeren på start knappen som dukker opp på skjermen. Da vil det dukke opp en slange og et eple på spillbrettet.Når spillet er i gang styrer spilleren en slange med tastaturet og slangen vil bevege seg rundt på et 10x10 stort rutenett. Målet med spillet er å samle/spise flest mulig epler som vises tilfeldig på skjermen. Når slangen spiser et eple, vokser den og scoren øker. Spillet avsluttes og spilleren har tapt dersom slangen kolliderer med veggene eller seg selv.

Snake appen inneholder også funksjonalitet for å registrere og vise den nåværende poengsummen, men også den beste poengsummen som er registrert. Hvis spilleren setter en ny rekord, blir dette markert. Jeg har også lagt til en vinn-tilstand, der spillet slutter når hele rutenettet er fylt av slangen.

Diagram



Spørsmål

1. Hvilke deler av pensum i emnet dekkes i prosjektet, og på hvilken måte? (For eksempel bruk av arv, interface, delegering osv.)
Prosjektet dekker flere deler av pensum i objektorientert programmering. Det mest sentrale er bruker av klasser som representerer ulike komponenter i spillet. Disse klassene inneholder både metoder og attributter som definerer deres tilstand og oppførsel.

Grensesnitt brukes i spillet. Ettersom spillet baserer seg på et rutenett/GridPane var det sentralt med getter metoder for posisjon i flere av klassene. Ettersom dette ble brukt såpass mye lagde jeg et interface for disse metodene. Dette interfacet ble kalt SnakeInterface og både XYvalues og Apple klassen implementerer dette grensesnittet.

For å kunne lagre en BestScore i spillet valgte jeg å benytte meg av filhåndtering. For å gjøre dette laget jeg klassen BestScoreFile. Her både skrives det og leses fra fil slik at det alltid vises hva som er den høyest oppnådde scoren. Denne scoren oppdateres dersom rekorden blir slått.

Snake spillet består av mange objekter og klasser som jobber sammen og da er assosiasjoner relevant. SnakeBody objekt består av en liste med flere XYvalues objekt altså 1-n assosiasjon.

For å lage snake har jeg brukt validering og unntakshåndtering. Dette for å blant annet sjekke at det hele tiden er gyldige verdier for slangen og eplene. Jeg har også brukt try-catch når jeg skriver til og leser fra fil. Ved å benytte meg av validering og unntakshåndtering har det blitt lettere å oppdage feil underveis og sikre gyldige tilstander.

2. Dersom deler av pensum ikke er dekket i prosjektet deres, hvordan kunne dere brukt disse delene av pensum i appen?
Med tanke på arv kunne man opprettet en overordnet klasse som definerer grunnleggende egenskaper og atferd som deles av både slangen og eplet. Deretter kan vi opprette underordnede klasser for hver av disse komponentene som arver egenskaper og metoder fra den overordnede klassen, samtidig som de kan ha sine egne unike egenskaper og metoder. Men dette er altså ikke gjort og jeg brukte interface i stedet.

Videre har jeg heller ikke brukt observatør-observert prinsippet. Men dersom dette skulle vært implementert kunne jeg for eksempel brukt det til å observere SnakeBody klassen og hvis slangen treffer veggen eller treffer seg selv gir den beskjed til observatøren. Det samme gjelder dersom slangen f.eks har spist et eple.

3. Hvordan forholder koden deres seg til Model-View-Controller-prinsippet? (Merk: det er ikke nødvendig at koden er helt perfekt i forhold til Model-View-Controller standarder. Det er mulig (og bra) å reflektere rundt svakheter i egen kode)
Snake spillet forholder seg i ganske stor grad til MVC-prinsippet, men det er også rom for forbedring. Spesielt med tanke på at snakeController gjør mye av View prinsippet.

1, Model: “SnakeBody”, “Apple”, “XYvalues” og “BestScoreFiles” er alle klasser som går under modellen i spillet. Disse klassene håndterer logikken og tilstanden i spillet som for eksempel bevegelse av slangen, generering av epler og håndtering ved kollisjoner osv.

2, View: SnakeController klassen og FXML klassen håndterer visningen av spillet .

3, Controller: snakeController fungerer som kontrollerern i spillet. Denne klassen håndterern hendelser fra brukergrensesnittet som for eksempel tastetrykk fra brukeren.

4. Hvordan har dere gått frem når dere skulle teste appen deres, og hvorfor har dere valgt de testene dere har? Har dere testet alle deler av koden? Hvis ikke, hvordan har dere prioritert hvilke deler som testes og ikke? (Her er tanken at dere skal reflektere rundt egen bruk av tester)
For å teste spillet har jeg brukt ulike former for testing både gjennom manuell testing og enhetstesting. Jeg har valgt å først fokusere på grunnleggende funksjonalitet og deretter litt mere avanserte scenarioer. Den manuelle testingen foregikk gjennom at jeg som kjørte spillet og spilte som en ekte bruker ville gjort. Her observerte jeg spillet under ulike scenarier som f.eks hva skjedde når slangen kolliderte, spiste et eple osv. Her var det lett å observere visuelle feil og uventet adferd. Jeg har også laget flere Junit tester for å teste funksjonalitet. Disse Junit testene består både av grunnleggende funksjonalitet for klassene, men også andre scenarioer som kunne ført til uventet oppførsel. Jeg testet f.eks at dersom BestScore ble slått ble denne nye scoren lagret korrekt i filen.