Projektbeskrivning

<Memory Game>

Innehåll

Projektrapport

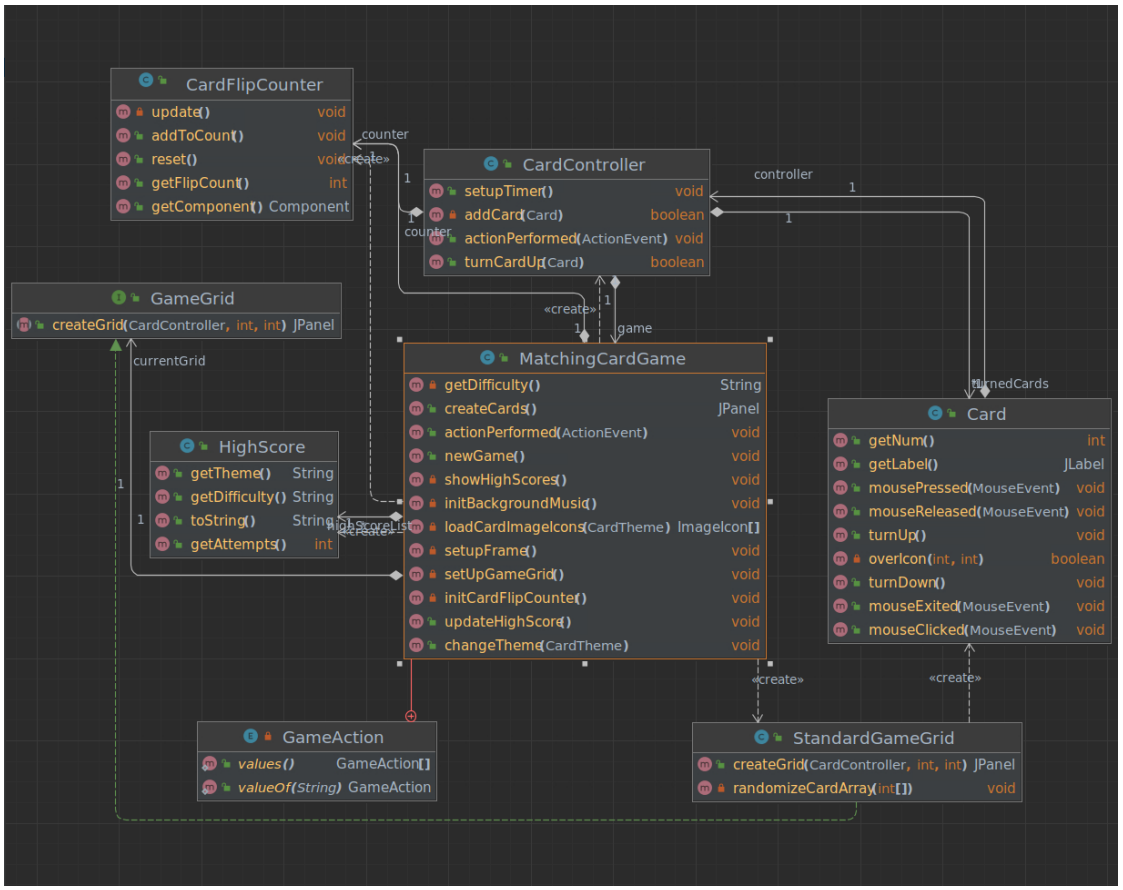
1. Implementationsbeskrivning

1.1. Dokumentation för programstruktur, med UML-diagram

- **Övergripande programstruktur**

Jag har implementerat ett kortspel med tre centrala moduler: spellogik, användargränssnitt och tema. Spellogikmodulen inkluderar viktiga klasser*, se Figur 3*, som **CardController** för kortlogik, **CardFlipCounter** för att räkna spelarens försök, **GameGrid**-gränssnittet för att hantera spelbrädets layout och **StandardGameGrid** för konkret implementation. **HighScore** sparar och hanterar högsta poängen medan **MatchingCardGame** styrspelet, från att skapa fönster till att hantera kort och visa slutresultat. Användargränssnittet hanteras av **GameUtils**-modulen med klasser som **BackgroundMusic**, **GameMenu** och **WelcomeMenu**, medan **Theme**-modulen möjliggör enkel tillägg av nya teman genom abstrakta och specifika klasser som **FlagCardTheme** och **PlaceCardTheme**.

- **Översikter över relaterade klasser** och hur de hänger ihop.



Figur 3: UML diagram som illustrerar hur klasserna i spellogik modulen hör ihop.

Spelet består av flera klasser som fungerar tillsammans för att skapa ett matchande

kortspel. Här är en beskrivning av hur dessa klasser är ihopkopplade:

Card: representerar enskilda kort i spelet och hanterar musrelaterade händelser med hjälp av "MouseAdapter". Den har metoder för att vända korten upp och ner baserat på spellogiken och övervakar musens position för interaktioner som musklick och musdragningar.

CardController: Hanterar logiken för korten genom att övervaka vilka kort som är uppvända och jämföra dem för att avgöra om de matchar varandra. Den styr interaktionerna mellan spelarens val och resultatet av kortmatchningarna.

CardFlipCounter: Ansvarar för att räkna antalet försök spelaren gör genom att vända på kort. Detta antal visas för spelaren för att ge feedback om antalet försök under spelets gång. Klassen är kopplad till CardController för att räkna varje försök.

GameGrid (gränssnitt): Specificerar hur spelkortens rutnät ska skapas och hanteras i spelet. Det ger flexibilitet för olika implementationer av spelbrädet och gör det möjligt att utöka med fler typer av rutnät i framtiden. StandardGameGrid är en konkret implementation som använder detta gränssnitt för att definiera spelbrädets layout och interaktioner.

HighScore: Ansvarar för att spara och hantera spelens högsta poäng. Den lagrar information såsom tema, svårighetsgrad, antal försök och tidpunkt för varje högsta poäng. HighScore klassen är kopplad till MatchingCardGame för att lagra och visa slutresultatet när alla kort har matchats.

MatchingCardGame: Huvudklassen som styr spelet. Den skapar spelets fönster, initierar och hanterar korten samt visar slutresultatet när alla kort har matchats. Den använder CardController för att hantera kortens logik och HighScore för att lagra spelresultatet.

StandardGameGrid: En konkret implementation av GameGrid-gränssnittet. Den skapar ett standardrutnät av spelkort och hanterar interaktionerna på spelbrädet baserat på användarens val och spellogiken som styrs av CardController och MatchingCardGame.

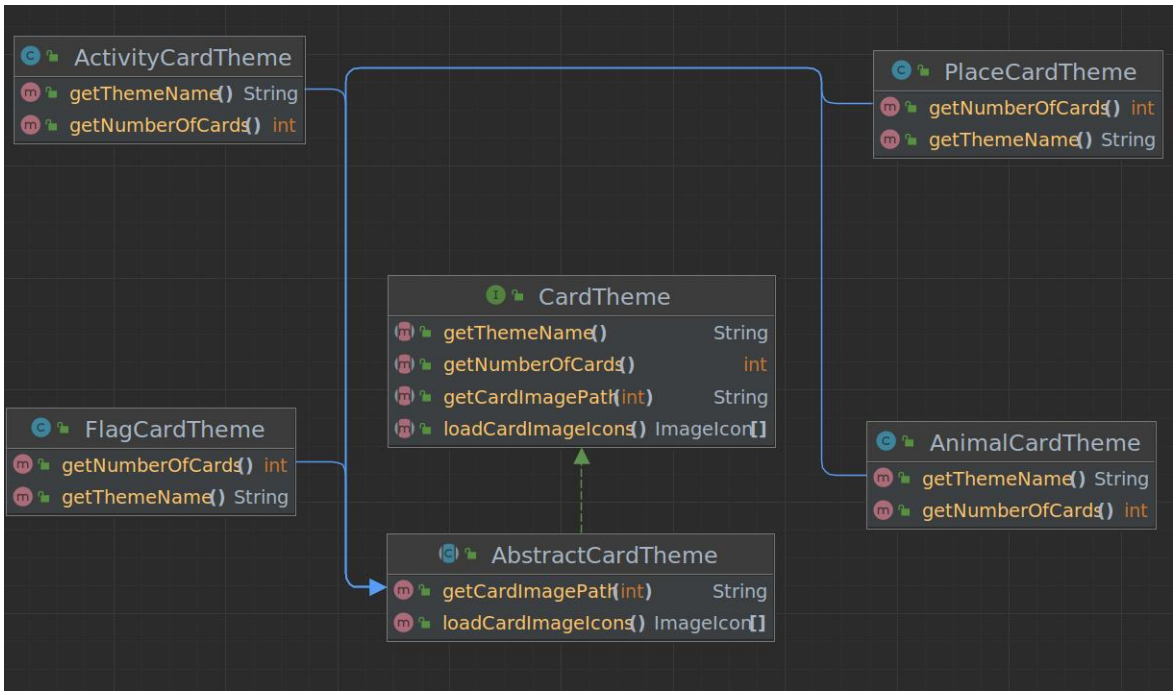
BackgroundMusic: Ansvarar för att hantera bakgrundsmusiken i spelet. Klassen laddar in och spelar musik som skapar en atmosfär för spelaren. Den kan styras via användargränssnittet för att pausa, spela eller justera ljudnivån under spelets gång.

GameMenu: Hanterar spelets huvudmeny och ger spelaren olika alternativ såsom att starta ett nytt spel, återgå till menyn eller avsluta spelet. Det möjliggör också framtida expansion genom att inkludera fler spelalternativ eller anpassningar.

WelcomeMenu: Skapar en välkomstskärm för spelaren vid spelets start. Det erbjuder grundläggande information om spelet, möjligheter att välja tema eller svårighetsgrad samt att navigera till huvudmenyn för att börja spela.

AbstractCardTheme: Detta är en abstrakt klass som implementerar interfacet **CardTheme**. Den definierar grundläggande egenskaper och metoder som alla teman måste ha, såsom hantering av olika typer av bilder eller ikoner för korten, *se Figur 4*.

FlagCardTheme: En konkret klass som utökar AbstractCardTheme och representerar ett tematiskt val med flaggor. Den specificerar vilka flaggor som ska användas på korten, vilket ger spelaren ett visuellt tema baserat på olika länders flaggor. Samma gäller **PlaceCardTheme**, **ActivityCardTheme** och **AnimalCardTheme**.



Figur 4: UML diagram som visar klasserna och gränssnittet i theme modulen och hur de hänger ihop

2. Användarmanual

Spelet startas genom att köra MAIN-klassen och trycka på RUN i IntelliJ efter att alla klasser har laddats in. En välkomstskärm visas med alternativen "**NEW GAME**" och "**EXIT**". *Se Figur 1.*

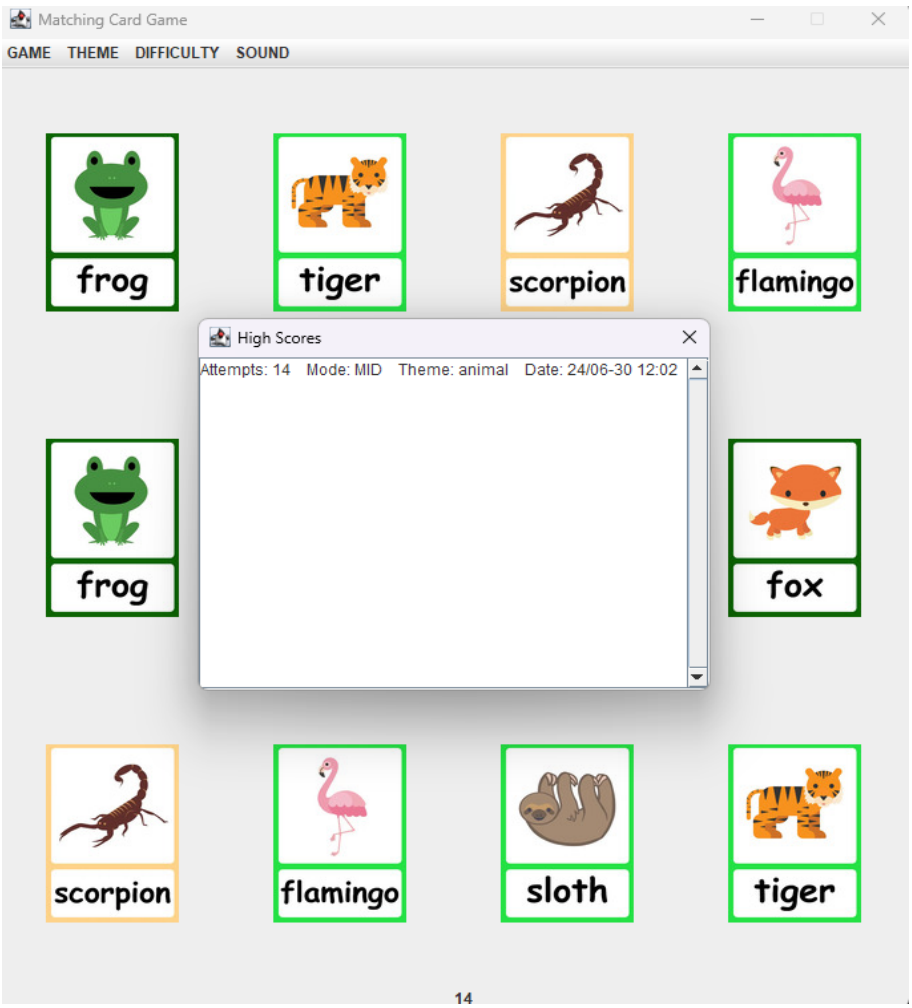
För att starta spelet klickar du på **"NEW GAME**". Därefter visas 12 kort med baksidan uppåt. Klicka på ett kort för att vända det och sedan på ett annat kort för att se om de matchar. Matchande kort förblir uppvända medan icke-matchande vänds tillbaka. Ha koll på kortens positioner!

När alla kort har matchats visas ett fönster med resultatet. Den visar antalet försök, svårighetsgrad, valt tema och speltid. *Se Figur 6.*

För att starta ett nytt spel navigerar du till menyfliken "**GAME**". Där finns alternativen "**NEW**" för att börja ett nytt spel, "**HIGHSCORES**" för att visa tidigare resultat, "**RULES**" för att läsa spelreglerna och "**EXIT**" för att avsluta.

På menyfliken finns även "**THEME**" där du kan välja bland olika teman, *se Figur 5,*"**DIFFICULTY**" för att justera svårighetsgraden och "**SOUND**" för att pausa eller spela bakgrundsmusiken.

Figur 5: Visar flagg-tema där spelaren är ett kort från vinst.



Figur 6: Visar ett avslutet spel med resultatet på skärmen.