Memory

Syfte

Syftet med den här uppgiften är att lära oss att självständigt skapa program i Objektorientead stil. Ni ska också arbeta i grupp om 3-4 personer och lära er använda gitbrancher för att organisera ert arbete sinsemellan.

Kursmål som täcks av uppgift

Kunskaper

9. Använda beprövade metoder för att felsöka kod
0. Använda versionshantering med Git i terminalen och GitHub
1. Självständigt utveckla programvara objektorienterat och enligt god programmeringssed

Vad ni ska leverera

README.md (med instruktioner om kompilering och körning/testning)

Card.java

Player.java

Game.java

Visibility.java

BoardView.java

UML-diagram.png (bild på ert diagram)

Planering.txt (beskrivning av problemnedbrytning)

.git

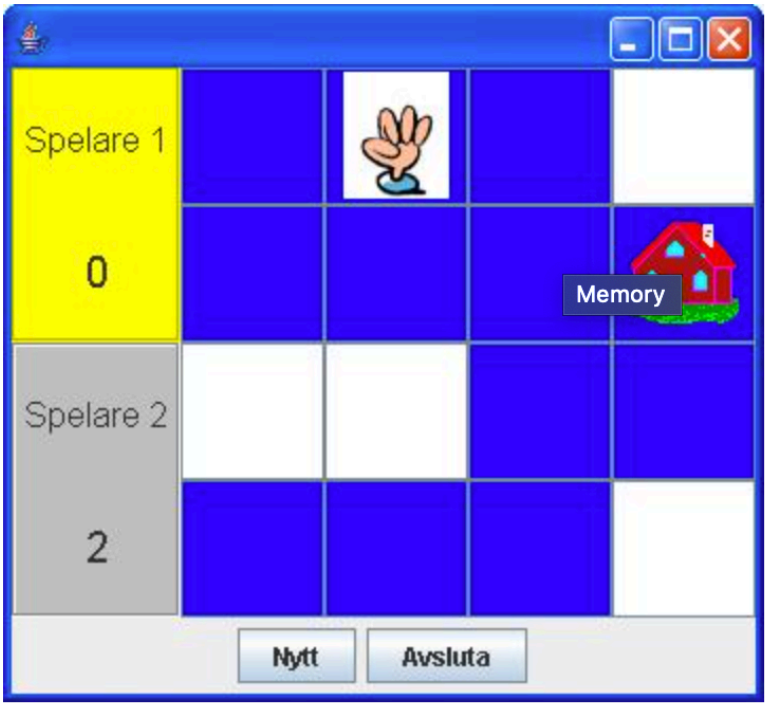
Uppgift

Spelet Memory kan som bekant spelas av två eller flera spelare. Till spelet hör ett antal kort med samma baksida men med olika bilder på framsidan. Varje bild förekommer på exakt två kort. När spelet börjar läggs alla korten ut på bordet med baksidan uppåt. Spelarna får sedan i tur och ordning vända på två kort. Om de två korten har olika bilder vänds de tillbaka så att baksidan kommer uppåt efter 2 sekunder. Nästa spelare får i så fall fortsätta. Om de två korten däremot har samma bild får spelaren ta korten från bordet och lägga dem i sin egen hög. Samma spelare får sedan fortsätta och vända två nya kort. En viss spelare får fortsätta så länge som han eller hon lyckas hitta kortpar med samma bild. Spelet fortsätter tills alla kort på bordet är borta. Den spelare som då har flest par har vunnit.

Kravspecifikation

Uppgiften är att skriva ett datorprogram som låter användarna spela Memory. Istället för att ha kort på bordet skall korten visas på skärmen. Spelarna väljer kort genom att klicka på korten. I programmet skall det finnas ett fönster med ett antal kort. Korten skall placeras ut i ett rutnät. Programmet skall utformas så att antalet rader och kolumner kan väljas när man startar programmet. I fönstret skall det också finnas två knappar som spelarna kan klicka på när de vill starta ett nytt spel respektive avsluta programmet. Programmet skall i grundutförandet ha två spelare. De två spelarnas aktuella poäng, d.v.s. antalet kortpar spelarna har hittat, skall visas. Programmet skall hålla reda på vilken spelares tur det är och detta skall tydligt markeras i fönstret. Man kan t.ex. använda olika färg för att visa spelarnas poäng.

Ett exempel på hur det kan se ut visas i följande figur i vilken Spelare 1 just har valt två kort som visade sig vara olika. I denna figur används gul bakgrundsfärg för att visa vems tur det är. Dom vita rutorna anger att Spelare 2 har hittat 2 par.



Ni ska i slutändan ha följande klasser: Card, Player, Game, BoardView, Visibility (enum)

Visibility (enum)

Representerar synligheten hos ett kort: VISIBLE, INVISIBLE, ELIMINATED (spelare har hittat par)

Card

Representerar ett kort/bild som kan vändas (Visibility) och vara visible, invisible, eliminated

Player

Representerar en spelare som kan vara active eller inactive (boolean), har points (int), namn (String)

Game

Representerar spellogiken och håller koll på vilken spelares tur det är, vilka kort som finns, samt kontrollerar BoardView. Denna klass är också huvudprogrammet som har main metod.

BoardView

Representerar det grafiska gränssnittet och ritar ut korten, markerar den aktiva spelaren, samt vänder på korten när någon trycker på ett kort helst representerat som en subklass till JButton.

För att designa spelet ska ni skapa ett UML-diagram över era klasser och alternativt hur de också interagerar (sekvensdiagram). För att göra detta kan ni exempelvis använda https://app.diagrams.net/

Ni ska också planera ert arbete i denna uppgift (helst innan ni börjar koda). Problemnedbrytning ska dokumenteras i en textfil Planering.txt. Denna kan ha följande format (Bankomat exempel) :

Problemnedbrytning

```
1. Bankomat GUI
    1. Rita ut knappar för pinkod, enter och avbryt
    2. Koppla knappar till lyssnare för att låta programmet få input
    3. Printa ut alla intryckta värden (input) vid varje nytt tryck
2. Autentisering
    1. Gör koppling till databas
    2. Fråga efter användare i databas (hämta lösenord)
    3. Kontrollera att användaren har tryckt rätt lösenord (matchar databasen)
3. Fel pin
    1. Om användaren har tryckt fel pin, låt hen göra 2 nya försök
    2. Annars ge felmeddelande
    3. Ge inte tillbaka kort till användaren (stulet)
4. Rätt pin
    1. Om användaren har tryckt rätt pin, visa saldo
    2. Ge användaren val att ta ut pengar
    3. Ta input från användaren om önskat belopp
    4. Kontrollera att användaren inte överskrider sitt saldo
5. Överskrider saldo
    1. Ge information om att användaren inte har tillräckligt på sitt konto
    2. Fråga efter input om nytt önskat belopp att ta ut
6. Har tillräckligt saldo
    1. Gör koppling till databas
    2. Uppdatera användares saldo
```

```
3. Ge användare pengar
4. Ge kort tillbaka till användare
```

Betygskrav

Betyg G

Programmet ska uppfylla specifikationen

Projektet ska vara versionshanterat med git, med flera commits som beskriver utvecklingen av programmet (OBS: en commit kommer klassas som fusk och ge IG)

Betyg VG

Projektet har en gitbranch per person som visar i flera commits vad var och en har gjort i projektet

Projektet har mer än 5 enhetstester (Junit) för spellogiken i spelet

UML diagram över design

En textfil (Planering.txt) med en beskrivning av hur ni har brytit ned problemet.