Laboration 4: Yatzy

Syfte

Syftet med denna laboration är att utveckla ett något större program som är uppbyggt av ett flertal klasser och som använder grafik och händelsestyrd programmering.

Förberedelser

Läs noga igenom laborationstesen. Börja inte koda innan ni analyserat problemet och verkligen förvissat er om att ni förstått vad de olika klasserna gör och hur de är tänkta att samverka med varandra. Följ de råd som ges angående programmets uppbyggnad.

Uppgift

Uppgiften består i att skriva ett grafiskt program för att spela Yatzy.

Spelet Yatzy spelas med 5 vanliga 6-sidiga tärningar. Spelet består av 15 ronder och spelet går ut på att få högsta möjliga poäng. I varje rond får man kasta tärningarna maximalt tre gånger. Första gången tärningarna kastas i en rond måste samtliga tärningar kastas, medan man i de två resterande kasten i en rond får kasta ett godtyckligt antal av de fem tärningarna, dvs man kan avstå från att kasta vissa tärningar.

Efter varje rond måste man föra in sitt resultat på ett resultatkort. Det finns 15 olika poängkategorier. Den poängkategori som väljs blir inte längre möjlig att välja i en senare rond. Nedan anges vilka poängkategorier som finns och vilken poäng som erhålls:

Ettor	summan av poängen för tärningarna med värdet 1
Tvåor	summan av poängen för tärningarna med värdet 2
Treor	summan av poängen för tärningarna med värdet 3
Fyror	summan av poängen för tärningarna med värdet 4
Femmor	summan av poängen för tärningarna med värdet 5
Sexor	summan av poängen för tärningarna med värdet 6
Par	summan av de två högsta tärningarna med samma värde
Två par	summan av poängen för två par av tärningar med inbördes samma värd
Triss	summan av poängen för tre tärningar med samma värde
Fyrtal	summan av poäng för fyra tärningar med samma värde
Liten Stege	15 poäng erhålles om träningarna är i följd från 1 till 5
Stor stege	20 poäng erhålles om träningarna är i följd från 2 till 6
Kåk	summan av poängen för tre respektive två tärningar med inbördes samma värde
Chans	summan av samtliga fem tärningar
Yatzy	50 poäng om alla tärningar har samma värde.

Dessutom gäller att om man erhåller mer än totalt 63 poäng tillsammans för de sex första poängkategorierna (Ettor till Sexor) fås en bonus på 50 poäng.

Testa hur programmet fungerar

För att kunna testa hur programmet fungerar finns en applet upplagd på kursens hemsida.

Programmets uppbyggnad

I ett Yatzy-spel finns fyra aktörer - spelaren, tärningarna, resultatkortet och spellogiken. Spellogiken utgöra *modellen* för spelets regler och hur spelet framskrider. Dessa aktörer samverkar med varandra på olika sätt under spelets gång. För att få en överblickbar struktur i ert program skall var och en av dessa aktörer utgöra en egen klass. Kalla dessa klasser för Player, FiveDices, ScoreCard och Model.

Ett Yatzy-spel, som implementeras i klassen Yatzy, *har* alltså ett objekt av klassen Player, ett objekt av klassen FiveDices, ett objekt av klassen ScoreCard och ett objekt av klassen Model.

Det grafiska gränssnittet som byggs upp i klassen Yatzy skall (ungefärligen) se ut som i figuren nedan:

I detta grafiska gränssnittet kan vi tydligt urskilja följande aktörer:

- Knappsatsen längst ned i figuren ovan är ett objekt av klassen Player. Här finns tre knappar; en knapp för att kasta tärningarna, en knapp för att starta om spelet med och en knapp för att avsluta programmet. Med dessa knappar kan användaren styra spelet genom att kasta tärningarna, starta om spelet eller avsluta spelet.
- Till höger i figuren finns den del som handhar tärningarna, dvs ett objekt av klassen FiveDices. När användaren gör en tryckning på knappen "GÖR KASTET" skall tärningarna kastas. Mellan kasten skall användaren kunna markera om han/hon vill kasta en specifik tärning eller inte i nästa kastomgång.
- Till vänster i figuren har vi ett objekt av klassen ScoreCard, dvs den del som handhar resultatkortet. Här skall användaren kunna markera den poängkategori som han/hon väljer efter en rond av spelet.

Förutom de grafiska komponenterna Player, FiveDices och ScoreCard finns också klassen Model som är vår modell eller *"spelmotor"* för spelat Yatzy. Klassen Model innehåller sådant som inte har med den grafiska presentationen att göra. För att få en bra uppbyggnad på programmet är det viktigt att gränssnittet är åtskilt från spellogiken. I alla spelprogram kan man förändra användargränssnittet men aldrig spellogiken (då skulle vi ju få ett annat spel).

De olika aktörerna i Yatzy-spelet - klasserna Player, FiveDices, ScoreCard och Model - måste samverka med varandra vid lösandet av den givna uppgiften. Således måste aktörerna kopplas i hop med varandra, dvs vi måste bestämma vilka *relationer* som aktörerna har sinsemellan. Detta kan göras på många olika sätt. För att få en bra och enkel struktur har vi i klassen Yatzy specificerat följande relationer:

Player: *känner till* FiveDices, *känner till* ScoreCard och *känner till* Model.

ScoreCard: *känner till* Player och *känner till* Model.

De olika klasserna

Klassen Model

Klassen Model håller reda på den aktuella ställningen i spelet och handhar de datastrukturer som är nödvändiga för detta.

Dessutom handhar klassen de spelregler som finns; nämligen att de 5 tärningarnas poäng beräknas på ett korrekt sätt för var och en av de olika poängkategorierna som finns på resultatkortet, att varje spelrond består av maximalt 3 kast samt att 50 bonuspoäng erhåller om de sex första poängkategorierna överstiger 63 poäng.

För att beräkna poängkategorierna skall klassen Model använda klassen PlayEngine som ni skrev i laboration 3.

Samspelet med Player

När klassen Player gör ett kast, får klassen Model ett meddelande om detta – tillsammans med vilka valörer de 5 tärningarna har. Klassen Model beräknar poängkategorierna och uppdaterar spelställningen.

I Yatzy är det tillåtet med maximalt 3 kast i varje rond. Kasten sker i klassen Player, men denna regel handhas av klassen Model. Således måste klassen Model hålla reda på antalet gjorda kast i den pågående ronden samt tillhandahålla metoder som klassen Player kan använda sig av för att exempelvis ta reda på om sista kastet i en i en rond har gjorts eller att meddela att en ny rond påbörjas. Ytterligare ett par andra metoder är nödvändiga, vilket ni upptäcker när ni börjar analysera klassen Player.

Samspelet med ScoreCard

Det är klassen ScoreCard som visar den aktuella ställningen i spelet, således måste klassen Model tillhandahålla metoder som möjliggör för ScoreCard att ta reda på den aktuella ställningen i spelet (dvs. samtliga uppgifter som visas på poängkortet).

Klassen ScoreCard

Komponenterna i det grafiska gränssnitt av resultatkortet kan naturligtvis väljas och placeras ut på många olika sätt. I förlagan (se figur sid 2), som ni förhoppningsvis testkör, utgörs den vänstra delen av JButton-objekt (med undantag för Summa, Bonus och Total som är JLabels) och den högra delen av JLabel-objekt. Komponenterna i den vänstra respektive den högra delen av resultatkortet har lagts in i var sin JPanel med användning av GridLayout. Dessa paneler har sedan satts samman till en ny panel, vilken därefter har placerats i en BorderLayout tillsammans med en JLabel med texten "Poängkort".

För att aktivera respektive avaktivera knapparna används metoden setEnabled och för att inte få några ramar på knapparna har metoden setBorderPainted använts (se vidare i API:n).

Tips: Använd ett fält för att handha JButton-objekten i den vänstra delen av resultatkortet och ett parallellt fält för att handha JLabel-objekten i den högra delen av resultatkortet.

När ni har blivit nöjda med layouten och fått knapparna att fungera som ni vill på resultatkortet skall ni lägga till klassen Model och se till att poängen, som en given tärningsuppsättning ger för de olika poängkategorierna, skrivs ut på korrekt sätt. Ett embryo för klassen Model finns på kursens hemsida, där det också finns ett testprogram (GraphicTestScore.java) som kan vara till hjälp.

Så småningom måste ni också börja fungera på hur klassen ScoreCard skall samspela med Player.

Klassen Player

Knapparna i klassen Player har i implementation av förlagan (se figur sid 2) placerats ut med hjälp av GridLayout.

Den grafiska delen i klassen Player är mycket enkelt. Svårigheten är att bestämma hur samspelet med klasserna ScoreCard och Model går till. Samspelet med klassen FiveDices är förutbestämd eftersom denna klass finns färdigskriven (se nedan).

Klassen FiveDices

Denna klass behöver ni inte skriva själva, utan den finns redan implementerad och kan kopieras från kursens hemsida. Studera koden och förvissa dej om att du förstår hur klassen fungerar.

I klassen finns tre instansvariabel:

```
private GraphicDice[] dices = new GraphicDice[5];
private Button[] buttons= new Button[5];
private boolean[] savedDices = new boolean[5];
```

Instansvariabeln dices är ett fält som innehåller de 5 grafiska tärningarna. Således gäller relationen att klassen FiveDices *har* 5 objekt av klassen GraphicDice. Instansvariabeln buttons är ett fält som innehåller knapparna som användaren kan trycka på för att markera om tärningen skall kastas eller inte kastas i nästa kastomgång. Instansvariabeln savedDices håller reda på vilka tärningar som skall kastas eller inte kastas i nästa kastomgång.

Följande metoder finns i klassen FiveDices

public int[] getDiceValues()	som returnerar ett fält som innehåller tärningarnas valörer
public void newRound()	som startar en ny rond med "blanka" tärningar
public void newThrow()	som kastar tärningarna
public void lockButtons()	som sätter "tärningsknapparna" icke-aktiva
public void unLockButtons()	som sätter "tärningsknapparna" aktiva

Klassen Yatzy

Det är i klassen Yatzy som den slutliga grafiska layouten görs och där alla kopplingar mellan aktörerna sker. Således gäller att klassen Yatzy *har* ett objekt av klassen Player, *har* ett objekt av klassen FiveDices, *har* ett objekt av klassen ScoreCard och *har* ett objekt av klassen Model.

Det gäller att

- objektet av klassen Player och objektet av klassen ScoreCard känner till varandra
- objektet av klassen ScoreCard och objektet av klassen Player känner till objektet av klassen Model.

Klassen Yatzy är redan implementerad och kan kopieras från kursens hemsida.

```
import java.awt.*;
import javax.swing.*;
public class Yatzy extends JFrame {
    public Yatzy() {
        FiveDices theDices = new FiveDices();
        Model theModel = new Model();
        ScoreCard theScoreCard = new ScoreCard(theModel);
        Player thePlayer = new Player(theDices, theScoreCard, theModel);
        theScoreCard.setPlayer(thePlayer);
        // skapa panelen i centrum
        JPanel centerPanel = new JPanel(new GridLayout(1,2));
        centerPanel.add(theScoreCard);
        centerPanel.add(theDices);
        // lägg ut komponenterna
        setLayout( new BorderLayout());
        add("South", thePlayer);
        add("Center", centerPanel);
        setBackground(new Color(0,130,0));
        setForeground(Color.blue);
        pack();
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    } //konstruktor
    public static void main(String[] arg) {
        Yatzy p = new Yatzy();
    } //main
} //Yatzy
```

Studera koden och *förvissa er om att ni förstår hur de olika objekten kopplas samman*, då detta har påverkan på de klasser ni själva skall implementera.

Tips på ordningsföljd i implementationsarbetet

1. Utveckla de grafiska delarna i klassen ScoreCard, dvs utseendet på resultatkortet.
2. Testa samspelet mellan ScoreCard och Model, dvs att en given tärningsuppsättning ger rätt poäng för var och en av de olika poängkategorierna på resultatkortet. Använd testklassen GraphicTestScore som finns på kursens hemsida där också ett embryo till klassen Model finns att tillgå för denna test. Förvissa dej om att du förstår hur klassen GraphicTestScore och embryot till klassen Model fungerar.
3. Utveckla de grafiska delarna av klassen Player.
4. Nu finns de grafiska delarna i samtliga klasser som klassen Yatzy använder. Ni har således ett körbart program
5. Sedan återstår att utveckla och testa själva spelförfarandet.

Redovisning

Laborationen skall dels demonstreras och godkännas av en handledare, dels redovisas skriftligt enligt de direktiv som finns på kursens hemsida.

Sista redovisningsdag och sista inlämningsdag för dokumentationen är fredag 2/3.