Laboration 5: Dunka Grodor

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

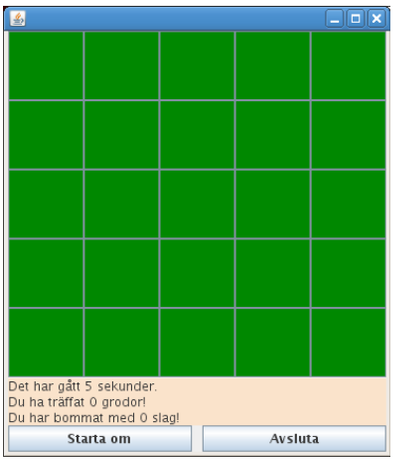
Läs noga igenom laborationstesen. Börja inte koda innan ni analyserat problemet. Följ de råd som ges angående programmets uppbyggnad.

Uppgift

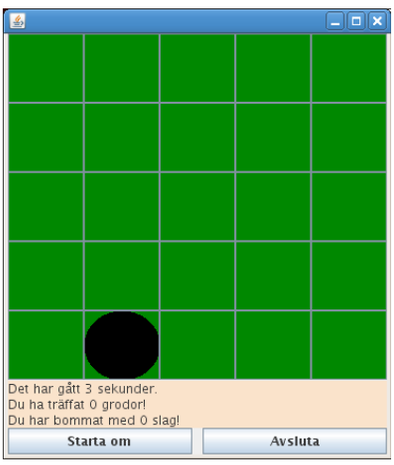
På Liseberg finns en attraktion, som enligt deras hemsida kallas *Huggkubben* och som går ut på att *slå på allt som rör sig*. Ett alternativt, men kanske inte lika politiskt korrekt, namn på detta slagnummer är *Dunka Grodor*.

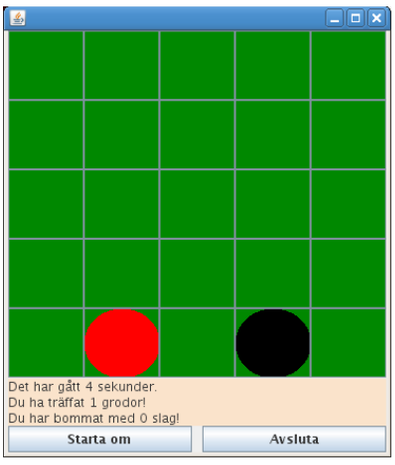
Din uppgift består i att göra ett grafiskt program för att simulera spelet *Dunka Grodor*.

När programmet startas visas ett fönster enligt figuren nedan:



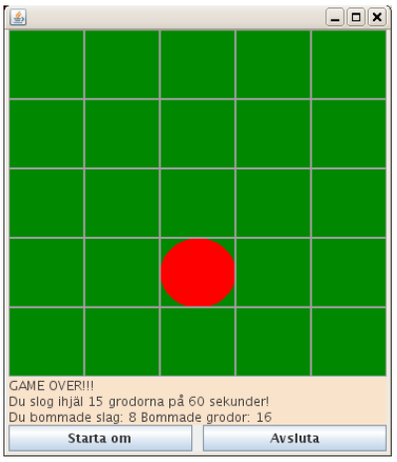
Fönstret består av tre delar. I den översta delen av fönstret finns ett antal knappar, i detta exempel 25 stycken. Det är på dessa knappar grodor slumpmässigt kommer att dyka upp. Grodorna kommer att vara synliga under en begränsad tid. Uppgiften för spelaren är att hinna trycka på knappen där en groda är synlig innan den försvinner, dvs hinna *dunka grodan*. I vårt exempel illustreras en grodda som dyker upp med en fylld svart cirkel, och en groda som blivit *dunkad* illustreras med en fylld röd cirkel. En groda som blivit dunkad visas under 5 sekunder, varefter den försvinner. En ny groda kan sedan på nytt dyka upp på knappen.





I den mittersta delen av fönstret visar hur lång tid som spelet pågått, hur många grodor som blivit träffade, samt hur många slag som bommat.

Flera olika varianter av spelet är tänkbara att implementera. I exemplet som illustreras här pågår spelet under 1 minut, varefter spelarens resultat skrivs ut och spelet stoppar.



I den nedersta delen av fönstret finns två knappar för att avsluta respektive återstarta programmet.

Testa hur programmet fungerar

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

Programmets uppbyggnad

För att få en bra struktur i ett program eftersträvar man att dela in programmet i ett antal lämpliga klasser, vilka sinsemellan samarbetar för att lösa uppgiften. Trots att det program ni skall utveckla är relativt litet kan designen av programmet göras på många olika sätt, därför ges här ett förslag på hur ni bör bygga upp ert program för att få en överblickbar struktur och för att underlätta ert utvecklingsarbete.

Det slutliga programmet, dvs det grafiska fönstret i vilket man kan dunka grodor, implementeras i klassen DunkaGrodor. När vi betraktar fönstret ser vi att det är uppbyggt av tre huvudkomponenter:

- spelplanen på vilken grodor kan dyka upp,
- resultattavlan där det bl.a. skrivs ut hur många grodor som blivit träffade,
- knappsatsen där användaren kan starta om eller avsluta spelet.

Implementera var och en av dessa komponenter i en egen klass. Kalla dessa klasser för InPutPanel, OutPutPanel respektive ButtonPanel.

För att få en bra uppbyggnad på programmet är det viktigt att användargränssnittet är åtskilt från spellogiken. I alla spelprogram kan man förändra användargränssnittet men aldrig spellogiken (då skulle vi ju få ett annat spel). Förutom de grafiska komponenterna InPutPanel, OutPutPanel och ButtonPanel behövs således en klass som innehåller sådant som inte har med den grafiska presentationen att göra, t.ex. att hålla reda på hur många grodor som blivit dunkade. Kalla denna klass för PlayEngine.

Klassen DunkaGrodor, som implementerar spelet Dunka Grodor, *har* alltså ett objekt av klassen InPutPanel, ett objekt av klassen OutPutPanel, ett objekt av klassen ButtonPanel och ett objekt av klassen PlayEngine.

De olika aktörerna i spelet Dunka Grodor - klasserna InPutPanel, OutPutPanel, ButtonPanel och PlayEngine - 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 flera olika sätt. Den elegantaste lösningen skulle vi få genom att använda det så kallade Observer-mönstret, men detta ligger utanför ambitionerna för denna kurs. Därför specificerar vi i klassen DunkaGrodor följande relationer:

PlayEngine: *känner till* InPutPanel och *känner till* OutPutPanel.

InPutPanel: *känner till* PlayEngine.

ButtonPanel: *känner till* PlayEngine.

De olika klasserna

Klassen PlayEngine

Klassen PlayEngine håller reda på spelreglerna och ansvarar för den bokföring som hör till detta. Det är alltså PlayEngine som bestämmer när en groda skall visas, vilken groda som skall visas samt under hur lång tid grodan skall visas. PlayEngine bestämmer vidare hur lång tid spelet skall fortgå, samt bokför t.ex. hur många grodor som blivit träffade och hur många slag som har utdelats.

PlayEngine *har* två objekt av klassen javax.swing.Timer. Ett av dessa objekt används för att bestämma när en groda skall dyka upp och det andra objektet används för att hålla reda på hur länge spelet pågått.

Samspelet med InPutPanel

Med de relationer som givits ovan känner klassen InPutPanel till klassen PlayEngine och klassen PlayEngine känner till klassen InPutPanel.

Den första relationen är nödvändig eftersom det är via knapparna i klassen InPutPanel som användaren gör sina knapptryckningar för att dunka grodorna, medan det är PlayEngine som håller reda på hur många knapptryckningar som gjorts och hur många grodor som träffats. InPutPanel måste alltså kunna rapportera knapptryckningarna till PlayEngine.

Den andra relationen är nödvändig eftersom det är PlayEngine som bestämmer vilken groda som skall visas, medan själva grodan visas i InPutPanel.

Samspelet med OutPutPanel

PlayEngine måste känna till OutPutPanel eftersom det är PlayEngine som t.ex. håller reda på hur många grodor som blivit träffade och hur lång tid spelet pågått, medan det är i OutPutPanel dessa uppgifter skrivs ut. Klassen OutPutPanel tillhandahåller således metoder för att kunna göra önskade utskrifter.

Klassen InPutPanel

Klassen InPutPanel utgör den grafiska spelplanen på vilken grodorna visas. Spelplanen består av ett antal knapp-objekt organiserade i ett två-dimensionellt fält. Knapp-objekten är av typen FrogButton, som beskrivs mer ingående nedan. Klassen InPutPanel *har* alltså ett fält av FrogButton-objekt.

Uppgiften för klassen InPutPanel är att visa de grodor som PlayEngine beordrar att visa upp. Vidare ansvarar InPutPanel för att avläsa de knapptryckningar användaren gör på spelplanen, avgöra om en groda blir träffad eller inte, samt rapportera resultatet till PlayEngine.

Klassen OutPutPanel

Klassen OutPutPanel är spelets resultattavla på vilken klassen PlayEngine anslår t.ex. hur många grodor som dunkats, hur många slag som utdelats och hur lång tid spelet pågått.

Lämpligen implementeras OutPutPanel med hjälp av tre JLabel- objekt.

Klassen ButtonPanel

Klassen ButtonPanel består av två knappar för att starta en ny spelomgång respektive avsluta spelet. När en ny spelomgång skall startas måste detta meddelas PlayEngine, eftersom det är PlayEngine som handhar all bokföring som berör spelet. ButtonPanel måste således känna till PlayEngine.

Klassen FrogButton

Klassen FrogButton är en utvidgning av klassen JButton. Denna klass används i klassen InPutPanel för att bygga upp spelplanen på vilken användaren dunkar grodor. Spelplanen består alltså av ett antal objekt av klassen FrogButton. På en viss plats på spelplanen, dvs på ett objekt av klassen FrogButton, kan det visas en groda, visas en träffad groda eller visas ingen groda alls. Detta innebär alltså att det finns tre olika tillstånd som kan visas på ett objekt av FrogButton. En icke-träffad groda visas som en fylld svart cirkel på grön botten, en träffad groda visas som en fylld röd cirkel på grön botten och ingen groda alls visas som en fylld grön cirkel på grön botten (dvs grodan syns inte).

Som tidigare nämnts skall en groda visas på spelplanen under en begränsad tid, och efter en viss tid försvinner en dunkad groda från spelplanen. Av denna anledning *har* klassen FrogButton ett objekt av klassen javax.swing.Timer.

Klassen DunkaGrodor

Det är i klassen DunkaGrodor som alla kopplingar mellan aktörerna skall göras. Således gäller att klassen DunkaGrodor *har* ett objekt av klassen InPutPanel, *har* ett objekt av klassen OutPutPanel, *har* ett objekt av klassen ButtonPanel och *har* ett objekt av klassen PlayEngine.

Klassen DunkaGrodor är redan implementerad. Koden visas nedan och kan kopieras från kursens hemsida.

```
import javax.swing.*;
import java.awt.*;
public class DunkaGrodor extends JFrame {
    public DunkaGrodor(int row, int col) {
        InPutPanel inPanel = new InPutPanel(row,col);
        OutPutPanel outPanel = new  OutPutPanel ();
        PlayEngine theEngine = new PlayEngine(row,col, inPanel,outPanel);
        ButtonPanel buttPanel = new ButtonPanel(theEngine);
        inPanel.setEngine(theEngine);
        JPanel center = new JPanel();
        center.setLayout(new BorderLayout());
        center.add("Center", inPanel);
        center.add("South", outPanel);
        setLayout(new BorderLayout());
        add("Center", center);
        add("South", buttPanel);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300,430);
        setVisible(true);
    }//konstruktor
    public static void main(String[] args) {
        DunkaGrodor win = new DunkaGrodor(5, 5);
    }//main
}//DunkaGrodor
```

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

Tips på ordningsföljd i implementationsarbetet

1. Börja med klassen FrogButton. Utveckla klassen stegvis, börja med att knappen kan visa de tre olika tillstånden och avsluta med att införa timer-objektet. Testa varje utvidgning av klassen genom att skriva enkla testprogram.
2. Utveckla de grafiska delarna i klassen InPutPanel.
3. Utveckla de grafiska delarna i klassen OutPutPanel.
4. Utveckla de grafiska delarna i klassen ButtonPanel.
5. Nu finns de grafiska delarna i alla klasser som klassen DunkaGrodor använder. Skriv ett skelett till klassen PlayEngine, så blir det möjligt att köra huvudprogrammet i klassen DunkaGrodor.
6. Utveckla och testa de delar av klassen PlayEngine som erfordras i samspelet med InPutPanel.
7. Utveckla och testa de delar av klassen PlayEngine som erfordras i samspelet med OutPutPanel.
8. Utveckla klassen ButtonPanel och de delar av klassen PlayEngine som erfordras.

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 11/3.