Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Magnus Myréen

Tentamen för Objektorienterad programvaruutveckling, TDA545

Fredag, 2014-10-31, 08:30-12:30, byggnad M

Ansvarig lärare:	Magnus Myréen, besöker byggnad M kl 09:00 och 11:30
Vitsords gränser:	3=28p, 4=38p, 5=48p, max 60p.
Hjälpmedel:	på sista sidan av dokument finns ett utdrag ur Javas API
Resultat:	skickas per epost från Ladok
Lösningar:	kommer att finnas på kurshemsidan
Granskning av rättning:	tider för detta kommer att finnas på kurshemsidan

Kom ihåg att inte fastna på en uppgift. Bestäm i förväg din egen tidsgräns per uppgift. ***Lycka till!***

Uppgift 1: [4 poäng totalt] *while-loopar, boolska uttryck, primitiva typer*

```
int x = 0;
int y = 1;
while (0 < y) {
    System.out.println(x);
    System.out.println(y);
    x = x + y;
    y = x + y;
}
System.out.println("slut");
```

a) Skriv ner de första 8 raderna av programmets utskrift. [2 poäng]

b) Skriver programmet någonsin `slut`? *Förklara ditt svar.* [2 poäng]

Uppgift 2: [4 poäng totalt] *referensvärden, parameteröverföring, metodanrop*

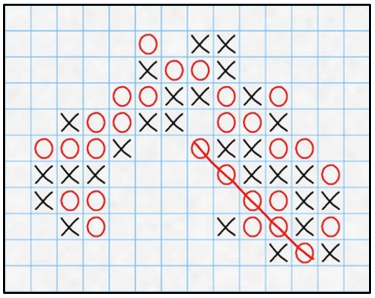
Skriv ner vad programmet skriver ut när man kör `java A`. [4 poäng]

```
public class A {
    public int n = 0;
    public static A fun(A t, A u) {
        u.n = u.n + 1;
        t = new A();
        t.n = 4;
        return u;
    }
    public static void main(String[] args) {
        A x = new A();
        A y = new A();
        x.n = 10;
        A z = fun(x,y);
        z.n = y.n + 2;
        System.out.println("x.n = " + x.n);
        System.out.println("y.n = " + y.n);
        System.out.println("z.n = " + z.n);
    }
}
```

Tips: Var noggrann! Bra att rita "minnesplatser" och "pilar" som vi gjorde på föreläsningarna.

Uppgift 3: [18 poäng totalt] *fält, for-loopar, exceptions*

Här implementerar vi en klass som representerar ett luffarschack spel (ibland kallas det fem-i-rad).



Luffarschack spelas på ett plan av rutor som antingen kan vara tomma, eller ha ett `X` eller ett `O` ritat i sig. Spelet tar slut när det finns fem celler med samma markering (`X` eller `O`) i rad, antingen lodrätt, vågrätt eller diagonalt.

Nedan finns det en skiss av klassen `Luffarschack`. Uppgiften är att fylla i koden som saknas, enligt punkterna (a), (b), (c) nedanför koden.

```
public class Luffarschack {

    public enum Cell { X, O, EMPTY };

    int width;
    int height;
    Cell[][] board;
    boolean xTurnNext;

    public Luffarschack(int width, int height) {
        ... svar till del (a) här ...
    }

    public void choose(int i, int j) throws IllegalArgumentException {
        ... svar till del (b) här ...
    }

    public boolean hasFiveInARow() {
        ... svar till del (c) här ...
    }

    ... eventuella hjälpmetoder här för del (a), (b), (c) ...
}
```

a) Skriv koden för konstruktorn. Den bör skapa ett tomt spel plan. [3 poäng]

b) Skriv koden för `choose`-metoden. Den bör skriva ett `X` i rutan med koordinater `i` och `j` ifall `xTurnNext` är sant, annars skriver den `O` i den rutan. I båda fallen skall den byta värdet på `xTurnNext`. Den bör kasta en exception ifall något är fel, t.ex. rutan inte finns eller inte är tom. [5 poäng]

c) Skriv koden för `hasFiveInARow`-metoden. Skriv kod som returnerar sant ifall spelet har tagit slut, annars skall den returnera falskt. Den får *inte* kasta exception. [10 poäng]

Tips: Del (c) är kanske stiligast löst med en (eller flera) hjälpmetoder, t.ex. en metod som kollar om det finns fem av samma i en given riktning från givna koordinater `i` och `j`.

Uppgift 4: [9 poäng totalt] *deklarationer, arv, gränssnitt*

a) Förklara alla delar av deklarationen (declaration) nedan. [2 poäng]

```
private static int numberOfCars = 0;
```

b) Ge *ett exempel* på överskuggning (overriding). [2 poäng]

c) Förklara *med ett exempel* vad abstrakta (abstract) metoder och klasser är. [3 poäng]

d) Förklara *med ett exempel* vad ett gränssnitt (interface) är. [2 poäng]

Uppgift 5: [12 poäng totalt] *arv, händelsehantering, timer, swing*

a) Beskriv kort den inbyggda `JFrame` klassen (dvs. `javax.swing.JFrame`). [2 poäng]

b) Beskriv kort den inbyggda `Timer` klassen (dvs. `javax.swing.Timer`). Beskriv vad en lyssnare är och hur `Timer` klassen använder sig av lyssnare, i detta fall en `ActionListener`. [3 poäng]

c) Skriv kod för en ny klass med namnet `SnapFrame`. Den nya klassen `SnapFrame` bör ärva `JFrame` och vara på nästan alla sätt precis samma som en `JFrame`. Klassen `SnapFrame` bör avvika från klassen `JFrame` när det gäller synlighet:

- En `SnapFrame` får endast vara synlig *en gång*, och då i *max* 5 sekunder.
- Efter att man anropar `setVisible(true)`, för första gången, skall `SnapFramen` *automatiskt* gömma sig efter 5 sekunder.
- Därefter bör det vara omöjligt att få fönstret synligt igen.

[7 poäng]

Obs: Det står max 5 sekunder. Se till att det är möjligt att gömma fönstret (med att kalla på `setVisible(false)`) innan det gått 5 sekunder.

Tips: Överskugga `setVisible` metoden och använd dig av superklassens `setVisible` metod.

Uppgift 6: [13 poäng totalt] *grafik, rekursion, swing*

Skriv kod som ritar, med Javas `Graphics` klass, rektanglar på följande sätt:

- Varje rektangel innehåller två mindre rektanglar.
- De inre rektanglarna har också två (ännu mindre) rektanglar i sig.
- Dessa små rektanglar har igen rektanglar i sig, dvs samma upprepas.
- Detta mönster upprepas tills de minsta rektanglarna är för små för att ritas.

Rektanglarna ska ritas i en `JPanel` som sitter i ett `JFrame` fönster, se exemplen nedan.

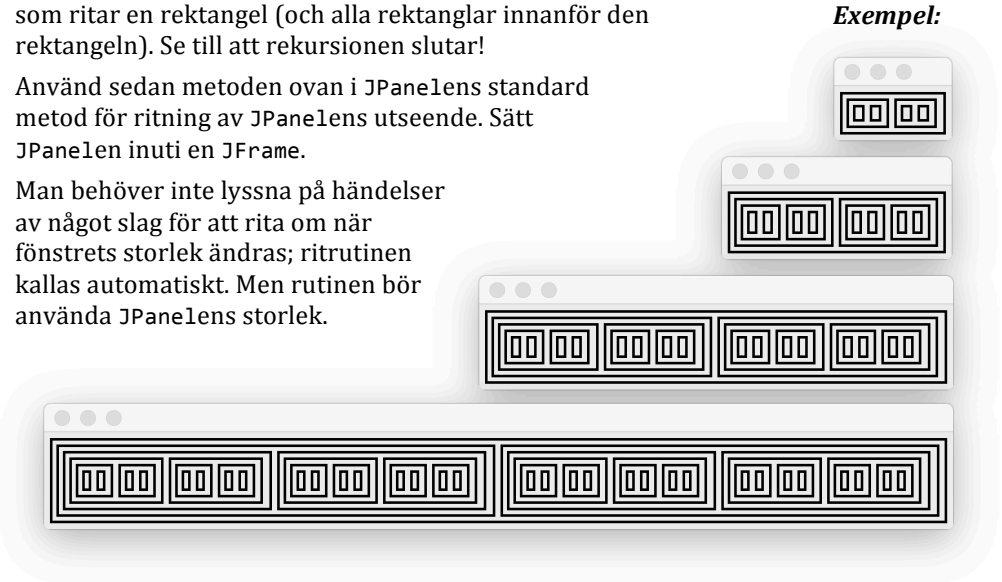
Obs. Rektanglarna bör fylla hela fönstret, även när användaren ändrar på fönstrets storlek.

Obs. Rektanglarnas linjer får inte kollidera eller ligga på varandra. *Förklara* varför de inte kolliderar / ligger på varandra med den koden som du har skrivit.

Tips:

- Det är kanske lättast att börja med att skriva den metoden som ritar en rektangel (och alla rektanglar innanför den rektangeln). Se till att rekursionen slutar!
- Använd sedan metoden ovan i `JPanelens` standard metod för ritning av `JPanelens` utseende. Sätt `JPanelen` inuti en `JFrame`.
- Man behöver inte lyssna på händelser av något slag för att rita om när fönstrets storlek ändras; ritrutinen kallas automatiskt. Men rutinen bör använda `JPanelens` storlek.

Exempel:



Utdrag ur Javas API. ***Obs.*** Man behöver inte använda alla dessa klasser. Man får också använda annat från Javas API.

Interface ActionListener

`void actionPerformed(ActionEvent e)`
Invoked when an action occurs.

Class ActionEvent extends AWTEvent extends EventObject extends Object

`ActionEvent(Object source, int id, String command)`
Constructs an ActionEvent object.
`String getActionCommand()`
Returns the command string associated with this action.

Class Color extends Object

`static Color BLACK`
The color black.

Class Component extends Object

`int getHeight()`
Returns the current height of this component.
`int getWidth()`
Returns the current width of this component.

Class Container extends Component extends Object

`Component add(Component comp)`
Appends the specified component to the end of this container.

Class FlowLayout extends Object, implements LayoutManager

`FlowLayout()`
Constructs a new FlowLayout with a centered alignment and a default 5-unit horizontal and vertical gap.

Class Graphics extends Object

`void drawRect(int x, int y, int width, int height)`
Draws the outline of the specified rectangle.
`abstract void fillRect(int x, int y, int width, int height)`
Fills the specified rectangle.
`abstract void setColor(Color c)`
Sets this graphics context's current color to the specified color.

Class JComponent extends Container extends Component extends Object

`protected void paintComponent(Graphics g)`
Calls the UI delegate's paint method, if the UI delegate is non-null.

Class JFrame extends Frame extends Window extends Container extends Component extends Object

`Container getContentPane()`
Returns the contentPane object for this frame.

Class JPanel extends JComponent extends Container extends Component extends Object

`JPanel()`
Creates a new JPanel with a double buffer and a flow layout.
`JPanel(LayoutManager layout)`
Create a new buffered JPanel with the specified layout manager

Interface LayoutManager

Class Object

`boolean equals(Object obj)`
Indicates whether some other object is "equal to" this one.
`String toString()`
Returns a string representation of the object.

Class Timer extends Object

`Timer(int delay, ActionListener listener)`
Creates a Timer and initializes both the initial delay and between-event delay to delay milliseconds.
`void setActionCommand(String command)`
Sets the string that will be delivered as the action command in ActionEvents fired by this timer.
`void setRepeats(boolean flag)`
If flag is false, instructs the Timer to send only one action event to its listeners.
`void start()`
Starts the Timer, causing it to start sending action events to its listeners.
`void stop()`
Stops the Timer, causing it to stop sending action events to its listeners.

Class Window extends Container extends Component extends Object

`void setVisible(boolean b)`
Shows or hides this Window depending on the value of parameter b.