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

Modelsvar för
Tentamen för Objektorienterad programvaruutveckling, TDA545

Onsdag, 2015-08-26, 08:30-12:30, byggnad M

Ansvarig lärare:	Magnus Myréen (besöker tentan)
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: [9 poäng totalt] *variabler, typer, klasser*

För varje exempel nedan, förklara varför Java kompilatorn (compiler) inte kommer att accepterar koden.

a) Typfel. [1 poäng]

```
int a = "1";
int b = "45";
System.out.println(a + b);
```

b) Det finns ingen konstruktör som tar int, int. [2 poäng]

```
public class Test1 {
  public static Test1 pass(int x, int y) {
    return new Test1(x,y);
  }
}
```

c) Det finns ingen variabel `a` utanför `test` metoden. [2 poäng]

```
public class Test2 {
  public int test(int a) { return this.a; }
}
```

d) *Klass*metoden (`static`) försöker använda *instans*variablen `a`. [2 poäng]

```
public class Test3 {
  private int a;
  public static int getA() { return a; }
}
```

e) Metoden returnerar inte en `int`. [2 poäng]

```
public class Test4 {
  public static int add(int x, int y) {
    System.out.println (x + y);
  }
}
```

Uppgift 2: [20 poäng totalt] *att skriva klass, rekursion, arrays*

Tips. Läs igenom hela uppgiften innan du börjar.

a) Denna uppgift handlar om en klass som representerar lådor. Varje låda har ett namn som ges via konstruktören. Varje låda kan innehålla ett antal lådor. Man anger innehållet med `setContent`-metoden.

```
public class Box {
  ...
  public Box (String name) {
    ...
  }
  public void setContent (Box[] contents) {
    ...
  }
  public void print (String prefix) {
    ...
  }
}
```

Skriv kod som implementerar allt utom `print`-metoden. [7 poäng]

```
public class Box {

    private String name;
    private Box[] content;

    public Box(String name) {
        this.name = name;
    }

    public void setContent(Box[] content) {
        this.content = content;
    }

    public void print(String prefix) { ... }
}
```

b) Implementera `Box` klassens `print`-metod så att koden nedan

```
Box red = new Box ("röd");
Box blue = new Box ("blå");
Box green = new Box ("grön");
Box yellow = new Box ("gul");
Box fuchsia = new Box ("magenta");
Box[] red_content = {blue, green, yellow};
red.setContent(red_content);
Box[] blue_content = {fuchsia};
blue.setContent(blue_content);
red.print("utskrift: ");
```

skriver följande utskrift på terminalen.

```
utskrift:   - en röd låda som innehåller:
utskrift:     - en blå låda som innehåller:
utskrift:       - en magenta låda som är tom
utskrift:     - en grön låda som är tom
utskrift:     - en gul låda som är tom
```

[9 poäng]

Tips. Man skapar lätt indenteringen genom att sätta " " till prefixen när man anropar `print`-metoden rekursivt.

```
public void print(String prefix) {
    System.out.print(prefix);
    System.out.print("  - en " + name + " låda ");
    if (content == null) {
        System.out.println("som är tom");
    } else {
        System.out.println("som innehåller:");
        for (int k=0; k < content.length; k++) {
            content[k].print(prefix + "   ");
        }
    }
}
```

c) Skriv en kodsnutt som skapar en låda som heter `black` (av typ `Box`). Lådan bör orsakar en 'oändlig' utskrift när man anropar `black.print("utskrift: ")`. Förklara varför utskriften upprepar sig. Exempel utskrift: [4 poäng]

```
utskrift:   - en svart låda som innehåller:
utskrift:      - en svart låda som innehåller:
utskrift:         - en svart låda som innehåller:
utskrift:            - en svart låda som innehåller:
utskrift:               - en svart låda som innehåller:
utskrift:                  - en svart låda som innehåller:
...
```

```
Box black = new Box ("svart");
Box[] black_content = {black};
black.setContent(black_content);
black.print("utskrift: ");
```

Förklaring: den svarta lådan skapas. Sedan skapas en array som består av den nya svarta lådan. Jag använder `setContent` metoden så att denna array används som innehåll för den svarta lådan som skapades först. När `print` anropas, kör den in i en oändlig loop för att `print` kallar rekursivt på `print` för varje element som finns i `contents`, men där finns ju samma låda igen. Utskriften fastnar i en loop.

Uppgift 3: [5 poäng] *loopar, utskrift*

Skriv ett program som skriver ut multiplikationstabellerna upp till tio. Exempel utskrift:

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

```
public class Mult {
    public static void main (String[] args) {
        for (int k = 1; k <= 10; k++) {
            for (int l = 1; l <= 10; l++) {
                System.out.print(" ");
                System.out.print(k * l);
            }
            System.out.println();
        }
    }
}
```

Uppgift 4: [9 poäng totalt] *exceptions*

a) Förklara begreppet exceptions. Hur fungerar exceptions i Java? När bör man använda exceptions? När bör man undvika dem? [2 poäng]

Javas exceptions används för att ta hand om sådana situationer som normalt inte skall inträffa och som är krävande att hela tiden kolla med `if`-statser. En exception är ett objekt som man kan *kasta*. När man kastar en exception slutar den normala körningen och programmet hoppar istället ut till den senaste `try-catch` statsen som kan fånga denna typs exception. Exceptions bör användas för ovanliga fall. Det går att skriva program så att exceptions används för normala returvärden, och andra hopp, men det skall man vara försiktig med för att exceptions är långsammare än vanliga metodreturer.

b) Definiera en egen exception typ som kallas `IntIntEx`. Denna exception bör innehålla två `int` värden. [4 poäng]

```
class IntIntEx extends Exception {
    private int x;
    private int y;
    public IntIntEx(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public int getX() { return x; }
    public int getY() { return y; }
}
```

c) Ge exempelkod som visar hur man skapar (create), kastar (throw) och fångar (catch) den nya `IntIntEx` exception typen. [3 poäng]

```
public class Test {
    public static void main(String[] args) {
        try {
            IntIntEx i = new IntIntEx(3,4);
            if (1 + 1 == 2) {
                throw i;
            }
            System.out.println("This is never printed.");
        } catch (IntIntEx j) {
            System.out.println(j.getX());
        }
    }
}
```

Uppgift 5: [17 poäng] *händelsehantering, swing, grafik*

Skriv ett program som visar ett fönster med följande funktionalitet: när man klickar på fönstrets yta bör en liten rektangel ritas vid koordinaterna där man klickade. Om man klickar flera gånger bör endast den senaste rektangeln synas.

Förklara hur ditt program implementerar denna funktionalitet.

Tips. Använd metoden `addMouseListener`, klassen `MouseEvent` och gränssnittet `MouseListener`. Ett utdrag ur Javas API finns på nästa sida.

```
public class Click extends JPanel implements MouseListener {

    private int x = -500 ;
    private int y = -500 ;

    public static void main(String[] args) {
        new Click();
    }

    public Click() {
        addMouseListener(this);
        JFrame f = new JFrame();
        f.add(this);
        f.setVisible(true);
    }

    public void paintComponent(Graphics g) {
        g.fillRect(x-5,y-5,11,11);
    }

    public void mouseClicked(MouseEvent e) {
        x = e.getX();
        y = e.getY();
        repaint();
    }

    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}

}
```

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

`Interface MouseListener`

`void mouseClicked(MouseEvent e)`
Invoked when the mouse button has been clicked (pressed and released) on a component.

`Class MouseEvent extends Object`

`int getX()`
Returns the horizontal x position of the event relative to the source component.
`int getY()`
Returns the vertical y position of the event relative to the source component.

`Class String extends Object`

`int length()`
Returns the length of this string.

`Class Component extends Object`

`void addMouseListener(MouseListener l)`
Adds the specified mouse listener to receive mouse events from this component.
`int getHeight()`
Returns the current height of this component.
`int getWidth()`
Returns the current width of this component. .
`void setVisible(boolean b)`
Shows or hides this component depending on the value of parameter b.

`Class Container extends Component extends Object`

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

`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.

`Class JButton extends AbstractButton extends JComponent extends Container extends Component …`

`JButton(String text)`
Creates a button with text.

`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.

`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 Random extends Object`

`Random()`
Creates a new random number generator.
`int nextInt()`
Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence.
`int nextInt(int n)`
Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

`Class Window extends Container extends Component extends Object`

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