Tentamen för TDA540
Objektorienterad Programmering

Institutionen för Datavetenskap
CTH HT-16, TDA540

Dag: 2017-01-09, *Tid:* 14.00-18.00

Ansvarig:	Alex Gerdes
Examinator:	Carlo A. Furia
Förfrågningar:	Alex Gerdes (alexg@chalmers.se, 0729744966)
Resultat:	Erhålls via Ladok
Betygsgränser:	3:a 24 poäng 4:a 36 poäng 5:a 48 poäng max 60 poäng
Siffror inom parentes:	Anger maximal poäng på uppgiften
Granskning:	Anslås på kurshemsidan. Vid eventuella åsikter om rättningen ange noggrant vad du anser är fel.
Hjälpmedel:	Cay Horstmann: *Java for everyone* eller Jan Skansholm: *Java direkt med Swing* Understrykningar och smärre förtydligande noteringar får finnas.
Var vänlig och:	Skriv tydligt och disponera papperet på lämpligt sätt. Börja varje uppgift på nytt blad. Skriv ej på baksidan av papperet.
Observera:	Uppgifterna är ej ordnade efter svårighetsgrad. Titta därför igenom hela tentamen innan du börjar skriva. Alla program skall vara väl strukturerade, lättöverskådliga och enkla att förstå. Indentera programkoden! Vid rättning av uppgifter där programkod ingår bedöms principiella fel allvarligare än smärre språkfel.

Lycka till!

Uppgift 1

Betrakta klassen `Match` nedan[1]:

```
1   class Match {
2     public static void match(String pattern, String text) {
3       int i, j;
4       for (i = 0; i <= text.length() - pattern.length(); i++) {
5         for (j = 0; j < pattern.length(); j++) {
6           if (pattern.charAt(j) != text.charAt(i + j))
7             break;
8         }
9         if (j == pattern.length())
10          System.out.println(i);
11      }
12    }
13
14    public static void main(String[] args) {
15      match("test", "kein protest, du testest");
16    }
17  }
```

a) Förklara kort varje *keyword* i klassen. (3 poäng)

b) Lista i, korrekt ordning, alla metodanrop som sker då programmet exekveras För varje metodanrop ange:

- i vilken klass metoden är deklarerad,
- metodens signatur,
- metodens returtyp, och
- om metoden är statisk eller ej.

(3 poäng)

c) Beskriv programmets 'kontrollflöde' när det exekveras (genom `java Match`) till och med första anropet av `break`. Räkna upp alla exekverade satser (i korrekt ordning) tillsammans med radnummer. Om satsen är en metodanrop ange också parametrarnas värde. (3 poäng)

d) Vilken utskrift fås då program körs? Förklara utskriften genom en (informell) beskriving av hur `match` metoden fungerar. Hur hänger parametrarna `pattern` och `text` ihop med varandra? (3 poäng)

e) Förbättra programmet:

- Byta ut innersta `for`-loopen mot en `while`-loop. Bytet skall medföra att `break` satsen försvinner.
- Ersätt argumenterna till `match` i anropet i `main` med inmatning av data via `System.in`.
- Gör så att metoden `match`, istället för att skriva ut resultatet, returnerar det. Använd en lämplig returtyp.

(5 poäng)

[1] Radnumren är inte del av programmet.

Uppgift 2

(3 poäng)

Nedanstående kodavsnitt har några (kompilerings)fel. Förklara vad som är fel och rätta till.

```
public static int power(int x, int n)
  int m = 1.0;
  for (int i = 0; i < n; i++)
    m *= x;
}
```

Uppgift 3

```
class Divide {
  static int[] nums   = {2, 3, 6, 8, 4, 8};
  static int[] denoms = {1, 2, 0, 2, 3};

  public static int divide(int a, int b) {
    return (a / b);
  }

  public static void main(String[] args) {
    try {
      for (int i = 0; i < nums.length; i++) {
        int number = divide(nums[i], denoms[i]);
        System.out.println(number);
      }
    } catch (ArithmeticException e) {
      System.out.println("Division by zero!");
    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Out of bounds index!");
    } finally {
      System.out.println("Done!");
    }
  }
}
```

a) Vad skrivs ut då programmet körs? (3 poäng)

b) Ändra programmet så att det också kastar ett undantag (exception) om $a \,\%\, b \neq 0$. (3 poäng)

c) Vad är den gemensamma superklassen till alla undantag (exceptions)? (1 poäng)

Uppgift 4 (12 poäng)

Vi har definierat en `Point` klass så här:

```
class Point {
  private final int x, y;

  public Point (int x, int y) {
    this.x = x;
    this.y = y;
  }

  public int getX () {
    return x;
  }

  public int getY () {
    return y;
  }
}
```

Din uppgift är att skapa en `Rectangle` klass. Man kan konstruera en rektangel med hjälp av två positioner: övre vänster och undre högre hörnen. Klassen skall därför ha följande konstruktor:

```
public Rectangle (Point upperLeft, Point lowerRight)
```

Konstruktorn skall kontrollera om positionerna är korrekta, d.v.s. att positionen `upperLeft` verkligen befinner sig faktiskt till vänster och ovanför positionen `lowerRight`. Om så inte är fallet kastar konstruktorn en `IllegalArgumentException`. Klassen ska ha (åtminstone) följande metoderna:

```
public Point getUpperLeft()
public Point getLowerRight()
public int area()
public boolean contains(Rectangle r)
public boolean overlaps(Rectangle r)
```

Metoden `area` skall returnera rektangelns area. Metoden `contains` avgör om rektangeln `r` är innesluten av en (annan) rektangel, t.ex. om `r1.contains(r2)` returnerar `true` så är `r2` innesluten av `r1`. Metoden `overlaps` kontrollerar om två rektanglar är överlappande. Ni får anta att vi har bara positiva koordinater.

Uppgift 5

Betrakta nedanstående klasser:

```
class A {
  int value = 1;

  public int getVal() {
    return value;
  }
}

class B extends A {
  int value = 2;

  public int getVal() {
    return value;
  }

  public int getSup() {
    return super.getVal();
  }
}

class C extends A {
  int value = 3;
}
```

Och anta att vi har gjort följande deklarationer:

```
A a = new A();
B b = new B();
C c = new C();
```

a) Hur är klasserna `A`, `B` och `C` relaterad till varandra med hänsyn till arv? Förklara ditt svar. (2 poäng)

b) Vad är värdet av följande uttryck:

1. `a.value`
2. `b.value`
3. `((A)b).value`
4. `c.getVal()`
5. `b.getSup()`
6. `((A)b).getVal()`

(3 poäng)

c) Bektrakta nedanstående klass:

```
class D {
  public static void printA(A x) {
    System.out.println(x.getVal());
  }

  public static void printB(B x) {
    System.out.println(x.getVal());
  }
}
```

Bestäm om följande metodanrop är korrekta och om så ange utskrifterna; om inte förklara varför de är inte korrekta och om det handlar om en kompilerings- eller runtime-fel.

1. `D.printA(a);`
2. `D.printA(b);`
3. `D.printA(c);`
4. `D.printB(a);`
5. `D.printB(b);`
6. `D.printB(c);`

(3 poäng)

Uppgift 6

Betrakta klassen `Tree` nedan:

```
class Tree extends Object {
  Tree l; // left subtree
  Tree r; // right subtree
  long data;

  Tree (long data) {
    super();
    this.l    = null;
    this.r    = null;
    this.data = data;
  }

  // Create a tree from two subtrees
  Tree (Tree g, Tree d, long data) {
    this(data);
    this.l = g;
    this.r = d;;
  }
}
```

a) Ta bort så mycket överflödiga delar som möjligt från klassen `Tree` utan att påverka klassens funktionalitet (med hänsyn till andra klasser som använder den här klassen). Du får *inte* lägga till något eller ändra namn! Förklara varför du kan ta bort vissa delar. (1 poäng)

b) Någon påstår:

> "`Match` (från uppgift 1) är ett imperativt program men `Tree` är ett objekt-orienterat program".

Vad menas med detta? Stämmer det (och varför)? (1 poäng)

c) Skapa en klass `ArityTree` som generaliserar `Tree` klassen och har ett godtyckligt antal sub-träd istället av två. (4 poäng)

d) Låt `Tree` ärva från `ArityTree` och uppdatera konstruktorerna. (2 poäng)

e) Utöka klassen `ArityTree` med en metod `flatten()` som returnerar alla `data` element (d.v.s inklusive alla element från alla subträden) i en lista. Fundera på vad en lämpligt returtyp är. Ordningen av elementen i listan spelar ingen roll. (4 poäng)

f) Är det möjligt att använda ett objekt av typen `ArityTree` där ett objekt av typen `Tree` förväntas? Och tvärtom? Förklara varför det är logisk. (1 poäng)