Omtentamen för TDA540
Objektorienterad Programmering

Institutionen för Datavetenskap
CTH HT-17, TDA540

Dag: 2018-08-30, *Tid:* 14.00-18.00

Ansvarig:	Alex Gerdes
Examinator:	Carlo A. Furia
Förfrågningar:	Alex Gerdes (alexg@chalmers.se, 031-772 6154)
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:	Tentamen kan granskas på studieexpeditionen. Vid eventuella åsikter om rättningen eposta och ange noggrant vad du anser är fel så återkommer vi.
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

(15 poäng)

Betrakta klassen `Uppgift1` nedan[1]:

```
1   public class Uppgift1 {
2     public static void main(String[] args) {
3       int[] xs = {3, 3};
4       int[] ys = {2, 3, 3, 3, 1, 4, 5, 2, 3, 3};
5
6       maf(xs, ys);
7     }
8
9     public static void maf(int[] xs, int[] ys) {
10      int i, j;
11
12      for (i = 0; i <= ys.length - xs.length; i++) {
13        for (j = 0; j < xs.length; j++) {
14          if (xs[j] != ys[i + j])
15            break;
16        }
17        if (j == xs.length)
18          System.out.println(i);
19      }
20    }
21  }
```

a) Förklara kort varje *keyword* i klassen. (2 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 Uppgift1`) till och med första anropet av `break`. Räkna upp alla exekverade *satser* (i korrekt ordning) tillsammans med radnummer. Om satsen är ett metodanrop ange också parametrarnas värde. (3 poäng)

d) Vilken utskrift fås då program körs? Förklara utskriften genom en (informell) beskrivning av hur `maf` metoden fungerar. (2 poäng)

e) Ändra programmet:

- Gör så att metoden `maf`, istället för att skriva ut resultatet, returnerar det. Använd en lämplig returtyp.
- Byta ut innersta `for`-loopen mot en `while`-loop. Bytet skall medföra att `break` satsen försvinner.

(5 poäng)

(OBS ett fel svar ger poängavdrag)

[1] Radnumren är inte del av programmet.

Uppgift 2 (4 poäng)

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

```
import java.util.List;

class Error {
  public static List<Character replicate(Integer size, Character c) {
    List<Character> res = new List<Character>();

    for (int i = 0; i < size; ++i)
      res.add(c)
  }
}
```

Uppgift 3 (7 poäng)

```
class InRange {
  public static void inRange(int n) throws Exception {
    if (n < 42)
      throw new NumberFormatException("Way too small!");
    else if (n > 1337)
      throw new Exception("Way too large!");
  }

  public static void main(String[] args) {
    int[] xs = {42, 33, 3137, 1000};

    for (int x : xs) {
      try {
        inRange(x);
        System.out.println(x + " is in range!");
      } catch (NumberFormatException e) {
        System.out.println(e.toString());
      } catch (Exception e) {
        System.out.println("Hmm...");
      }
    }
  }
}
```

a) Vad skrivs ut då programmet körs? (4 poäng)

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

c) Vad händer om vi tar bort sista `catch` blocket? (2 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 `Circle` klass. Man kan konstruera en cirkel med en mittpunkt och radie. Klassen skall därför ha följande konstruktor:

```
public Circle(Point center, double radius)
```

Konstruktorn skall kontrollera om radie är större än noll, om så inte är fallet kastar konstruktorn en `IllegalArgumentException`. Klassen ska ha (åtminstone) följande metoderna:

```
public Point getCenter()
public double getRadius()
public double area()
public double circumference()
public boolean contains(Circle c)
public boolean overlaps(Circle c)
```

Ni får anta att vi har bara positiva koordinater. Metoden `area` skall returnera cirkelns area och metoden `circumference` beräknar cirkelns omkrets. Metoden `contains` avgör om cirkeln `c` är innesluten av en (annan) cirkel, t.ex. om `c1.contains(c2)` returnerar `true` så är `c2` innesluten av `c1`. Metoden `overlaps` kontrollerar om två cirklar är överlappande. OBS, i metoderna `overlaps` och `contains` kommer man troligen att jämföra reella tal med varandra och måste man ta hänsyn till trunkeringsfel. *Hint*: tillåt en viss skillnad när ni jämför reella tal, en så kallad epsilon.

Uppgift 5 (14 poäng)

Betrakta nedanstående klasser och interfaces:

```
interface A {
    int getX();
}

abstract class B implements A {
    protected int x;
}

interface C {
    void printX();
    void printY();
}

public class D extends B {
    public int getX() {
        return x;
    }
}

public class E extends D implements C {
    public void printX() {
        System.out.println(getX());
    }
    public void printY() {
    }
}

public class F extends E {
    protected int x = 4;
    @Override
    public void printY() {
        super.printX();
    }
}
```

Anta att vi har gjort följande deklarationer:

```
D d = new D();
E e = new E();
F f = new F();
```

a) Hur är `A`, `B`, `C`, `D`, `E` och `F` relaterad till varandra med hänsyn till arv? Rita en *klassdiagram* där alla klasser representeras av en nod och rita en pil mellan nod X och nod Y om Y är den direkta superklassen till X. (4 poäng)

b) Vad är värdet av följande uttryck:

1. `d.getX()`
2. `e.x`
3. `((D)e).getX()`
4. `f.getX()`
5. `f.x`
6. `((D)f).getX()`
7. `((E)f).getX()`
8. `((E)f).x`

(4 poäng)

c) Vad blir utskriften när vi evaluerar följande satser?

1. `e.printX();`
2. `e.printY();`
3. `f.printX();`
4. `f.printY();`

(2 poäng)

d) Bestäm om följande satser är korrekta; om inte förklara varför de inte är korrekta och om det handlar om en kompilerings- eller runtime-fel.

1. `A v1 = new A();`
2. `A v2 = new D();`
3. `C v3 = new D();`
4. `D v4 = new E();`
5. `C v5 = new E();`
6. `F v6 = new D();`
7. `List<D> v7 = new ArrayList<D>();`
8. `ArrayList<D> v8 = new ArrayList<E>();`
9. `A v9 = new B();`

(4 poäng)

Uppgift 6

(8 poäng)

För att avgöra om ett n-siffrigt tal N är ett Keithtal skapar man en Fibonacci-liknande talföljd som börjar med de n siffrorna i N med den mest signifikanta siffran först. Man fortsätter sedan talföljden med termer som var och en är summan av de n föregående termerna. N är ett Keithtal om N ingår i den på detta sätt konstruerade talföljden och är större än 9.

Betrakta exempelvis ett tresiffrigt tal $N = 197$. Talföljden blir då:

$$1, 9, 7, 17, 33, 57, 107, 197, 361, \ldots$$

Eftersom 197 ingår i talföljden så är det ett Keithtal. Om $N = 123$ då blir talföljden:

$$1, 2, 3, 6, 11, 20, 37, 68, 125, \ldots$$

och kan vi dra slutsatsen att 123 är inget Keithtal, för 123 är inte med i talföljden.

Skriv en metod som givet ett heltal n returnerar om talet är ett Keithtal eller inte.