Omtentamen för TDA540
Objektorienterad Programmering

Institutionen för Datavetenskap
CTH HT-17, TDA540

Dag: 2018-04-06, *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

(18 poäng)

Betrakta klassen `Votes` nedan[1]:

```
1   class Votes {
2
3      public static void countVotes(int nOptions, int[] votes) {
4         int[] count = new int[nOptions];
5         for (int k = 0; ; k++) {
6            int vote = votes[k];
7            count[vote - 1]++;
8            if (k == votes.length - 1)
9               break;
10        }
11        for (int k = 0; k < count.length; k++) {
12           System.out.println("Option " + (k + 1) + " got " + count[k] + " votes");
13        }
14     }
15
16     public static void main(String[] args) {
17        int[] votes = {1, 2, 1, 1, 3, 2, 1, 4, 3, 2};
18        countVotes(4, votes);
19     }
20  }
```

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 Votes`) till och med första anropet av `if`. 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) beskriving av hur `countVotes` metoden fungerar. (3 poäng)

e) Ändra programmet:

- Ändra första `for`-satsen i `countVote` så att den inte avslutar med en `break`-sats.
- Ändra typen av lokala variabeln `count` från `int[]` till `Integer[]` och modifiera resten av `countVotes` så att metoden har fortsatt samma beteende.

(6 poäng)

[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.

```
public static int max(int[] a) {
  int m = a[0];
  while (k < a.length) {
    if (a[k] > m)
       m = a[k];
    k + 1;
  }
  System.out.println(m);
}
```

Uppgift 3

(8 poäng)

```
class Conversions {

   public static int cc(char c) throws Exception {
      switch(c) {
         case 'A':
            return 1;
         case 'B':
            return 2;
         case 'C':
            return 3;
         default:
            throw new RuntimeException("Out of range!");
      }
   }

   public static void main(String[] args) {
      int n = 0;
      try {
        n += cc('A');
        System.out.println(n);
        n += cc('C');
        System.out.println(n);
        n += cc('D');
      } catch (Exception e) {
        n += 3;
      } finally {
        System.out.println(n);
      }
   }
}
```

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

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

c) Hur ändras programmets beteende om `try`-blocket bara skulle innehålla sista anropet `cc('D')`? Det vill säga `main`-metoden ser ut så här:

```
public static void main(String[] args) {
  int n = 0;
  try {
    n += cc('D');
  }
}
```

Kom ihåg att `Exception` är en *checked exception* klass medan `RuntimeException` är en *unchecked exception* klass. (4 poäng)

Uppgift 4 (16 poäng)

I den här uppgiften ska ni skapa en klass som lagrar data-punkter (med typen `double`) och som kan utföra enkla statistiska beräkningar. Vi har definierat en `BasicData` klass som lagrar data-punkter i en privat array `data` och exponerar följande publika metoder: `put(d)` för att lägga till en data-punkt `d`, `get(k)` för att hämta en data-punkt med index `k`, och `size()` för att läsa av antalet lagrade data-punkter.

```
class BasicData {

   private double[] data;
   private int size;

   public BasicData() {
      data = new double[10000];
      size = 0;
   }

   public void put(double d) {
      data[size] = d;
      size++;
   }

   public double get(int n) {
      return data[n];
   }

   public int size() {
      return size;
   }
}
```

Din uppgift är att skapa en `Data` klass. `Data` ärver från `BasicData` och utökar den med följande publika metoder och konstruktorer:

- `Data()` skapar en tom samling data-punkter
- `int argmin()` returnerar den minsta data-punktens index
- `double min()` returnerar minsta lagrade data-punkt
- `double sum()` returnerar algebraisk summa av alla data-punkter
- `double mean()` returnerar *medelvärdet* av alla data-punkter; kom ihåg att medelvärdet av punkter $d_1, \ldots, d_n$ är $\frac{\sum_k d_k}{n}$
- `Data filter(double x)` returnerar ett nytt `Data` objekt med en kopia av alla data-punkter som är mindre än `x`
- `Data centered()` returnerar ett nytt `Data` objekt med en kopia av alla data-punkter, där alla punkter är delade med medelvärdet

Uppgift 5 (14 poäng)

Betrakta nedanstående klasser och interfaces:

```
interface A {
   int get();
}
interface B extends A {
   void put(int val);
}
public class C implements A, B {
   protected int val = 0;
   public int get() {
      return 1;
   }
   public void put(int val) {
   }
}
public class D extends C {
   public int get() {
      return val;
   }
}
public class E extends D {
   public void put(int val) {
      this.val = val;
   }
}
public class F extends E {
   protected int val;
   public void put(int val) {
      this.val = val;
   }
}
public class G extends F {
   public int get() {
      return val;
   }
}
```

Anta att vi har gjort följande deklarationer:

```
C c = new C();
D d = new D();
E e = new E();
F f = new F();
G g = new G();
```

a) Hur är `A`, `B`, `C`, `D`, `E`, `F` och `G` 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) Betrakta följande sekvens med metodanrop:

```
c.put(3);
System.out.println( c.get() );        // 1.
d.put(3);
System.out.println( d.get() );        // 2.
e.put(3);
System.out.println( e.get() );        // 3.
f.put(3);
System.out.println( f.get() );        // 4.
((E)f).put(3);
System.out.println( ((E)f).get() );   // 5.
System.out.println( f.get() );        // 6.
((D)f).put(3);
System.out.println( ((D)f).get() );   // 7.
((C)f).put(3);
System.out.println( ((C)f).get() );   // 8.
g.put(3);
System.out.println( g.get() );        // 9.
((F)g).put(3);
System.out.println( ((F)g).get() );   // 10.
System.out.println( g.get() );        // 11.
```

Vilket värde skrivs ut i varje `get()` anrop? (6 poäng)

c) 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 a1 = new D();`
2. `D d2 = new A();`
3. `A a3 = new A();`
4. `C c4 = new G();`
5. `G g5 = c4;`
6. `A a6 = c4;`
7. `List<A> v7 = new ArrayList<A>();`
8. `List<A> v8 = new ArrayList<C>();`

(4 poäng)