Tentamen Programmeringsteknik I 2015-01-10

Skrivtid: 8.00 – 13.00
Hjälpmedel: Java-bok

Detta är ett lösningsförslag till tentamen. Lösningarna är inklistrade i rutor vid respektive fråga i detta dokument.

Tänk på följande:

- Det finns en referensbok (Java) hos tentavakten som du får gå fram och läsa men inte ta tillbaka till bänken.
- Skriv läsligt! Använd inte rödpenna!
- Skriv bara på framsidan av varje papper.
- Lägg uppgifterna i ordning. Skriv uppgiftsnummer och tentamens-kod (eller namn om du saknar sådan) på alla papper. Skriv inte längst upp i vänstra hörnet - det går inte att läsa där efter sammanhäftning.
- Fyll i försättssidan fullständigt.
- Det är principer och ideér som är viktiga. Skriv så att du övertygar examinator om att du har förstått dessa även om detaljer kan vara felaktiga.
- Alla uppgifter gäller programmeringsspråket Java och programkod skall skrivas i Java. Koden skall vara läslig dvs den skall vara vettigt strukturerad och indenterad.
- Namn på variabler, metoder, klasser etc skall vara beskrivande men kan ändå hållas ganska korta.
- Observera att poängavdrag bland annat kan göras för
  - ✓ icke-privata instansvariabler,
  - ✓ dålig läslighet,
  - ✓ upprepning av identisk kod och
  - ✓ underlåtenhet att utnyttja given eller egen tidigare skriven metod
- Det är totalt 30 poäng på skrivningen. Betygsgränser: 16 räcker säkert till 3, 22 säkert till 4, 27 säkert till 5.

Lycka till!

Torsten Andersson och Tom Smedsaas

Uppgift 1

Nedanstående, inte helt korrekta, klass skall representera en cirkel.

```
 1  public class Circle {
 2    private double r,x,y;
 3
 4    public Circle (double x, double y, double r) {
 5      this.x = x; this.y = y; this.r = r;
 6    }
 7
 8    public int area () {
 9      return Math.PI * this.r * this.r;
10    }
11
12    public void scale (double sf) {
13      this.r = sf * this.r ;
14    }
15
16    public static void main ( String [ ] arg ) {
17      Circle c = new Circle(0.0,0.0,0.5);
18      System.out.println(c);
19      double sf = c.scale(2.0);
20      System.out.println(c);
21      double a = c.area();
22      System.out.println(a);
23    }
24  }
```

När main-metoden körs är det meningen att utskriften ska se ut på följande sätt:

```
x=0.0, y=0.0, r=0.5
x=0.0, y=0.0, r=1.0
3.141592653589793
```

a) När man kompilerar klassen fås följande meddelande:

```
2 errors found:
File: Circle.java  [line: 9]
Error: Circle.java:9: possible loss of precision
found   : double
required: int

File: Circle.java  [line: 19]
Error: Circle.java:19: incompatible types
found   : void
required: double
```

Vad är det som är fel? Vad skall det stå? (1p)

Lösning:

Vad som är fel:

Satsen på rad 9 returnerar ett double-värde, men enligt sats 8 är metodens returtyp int.

På rad 19 anropas en void-metod som den vore en metod som returnerar något (double).

Vad som kan stå:

Rad 9 ändras till: public double area ()

Rad 19 ändras till: c.scale(2.0);

b) När dessa kompileringsfel rättats ger en körning följande utskrifter:

```
Circle@11fb24d3
Circle@11fb24d3
3.141592653589793
```

Förklara resultatet. Ändra klassen så att utskriften blir som förväntat (mainmetoden får dock inte ändras). (2p)

Lösning:

Vid en utskrift av Circle-objekt kommer en toString-metod att anropas. Eftersom det saknas en toString metod i klassen Circle kommer en toString-metod i klassen Object att användas och den orsakar den utskrift som visas. Med följande toString-metoden i klassen Circle kommer utskriften bli som önskas:

```
public String toString() {
      return "x="+this.x + ", y="+this.y + ", r="+this.r;
 }
```

c) Skriv metoden `collide` som returnerar `true` om två cirklar kolliderar med varandra, annars `false`. Två cirklar anses kollidera om avståndet mellan deras centrumpunkter är mindre än summan av deras radier.

Exempel på anrop (givet att c2 och c är Circle-objekt):

```
    if (c2.collide(c)) {
       System.out.println("Kollision" );
    }
```

(3p)

Lösning:

```
public boolean collide(Circle c) {
    double dx = this.x - c.x;
    double dy = this.y- c.y;
    double d = Math.sqrt(dx*dx + dy*dy);
    return d <= this.r + c.r;
 }
```

Uppgift 2

Givet är följande klass:

```
public class Book {
  private String isbn;   // Bokens identitet
  private int number;    // Antal exemplar av boken
  private double price;  // Bokens pris

  public Book(String isbn, int number, double price) {
    this.isbn = isbn;
    this.number = number;
    this.price = price;     // b-uppgiften
  }

  public String toString() {
    return "["+this.isbn+","+this.number+","+this.price+"]";
  }

  public static void main (String[] arg) {
    // Boktitel med identiteten 999-91-7000-150-5, antal ex 10, priset 500 kr
    Book b = new Book("999-91-7000-150-5",10,500.00);
    System.out.println(b);
    for (int i=1; i<=3; i++) {
      if (b.buy(4)) {    // Önskar köpa fyra exemplar av boken
        System.out.println("Köpet genomfört: " + b);
```

```
      }
      else {
        System.out.println("Köpet gick ej att genomföra: " + b);
      }
    }
  }  // main
} // Slut klassen Book
```

Metoden buy i klassen Book saknas dock för att mainmetoden skall fungera. Metoden skall hantera ett köp av ett antal exemplar av den aktuella boken, om köpet går att genomföra.

a) Skriv den saknade metoden buy så att när man kör mainmetoden får följande resultatutskrift:

```
[999-91-7000-150-5,10,500.0]
Köpet genomfört: [999-91-7000-150-5,6,500.0]
Köpet genomfört: [999-91-7000-150-5,2,500.0]
Köpet gick ej att genomföra: [999-91-7000-150-5,2,500.0]
```

(3p)

Lösning:

```
public boolean buy(int n) {
   if (this.number<n) {
     return false;
   }
   this.number=this.number-n;
   return true;
}
```

b) Om satsen `this.price = price;` ändras till `price = price;`
Vad blir resultatutskriften av mainmetoden i det fallet? Motivera svaret. (1p)

Lösning:

Instansvariabeln price kommer därmed inte att tilldelas något värde i konstruktorn, dvs den kommer att ha sitt initiala värde som är 0.0. Detta betyder att Book-objekt kommer få ett pris på noll kronor. Resultatutskriften blir identisk som den visade, förutom att värdet 500.0 ersätts med 0.0.

Uppgift 3

Givet är följande:

```
public class BookStore {
  private String name;
  private Book[] books;
  private int numberOfBookTitles;

  public BookStore(String name, int n) {
    this.name = name;
    this.books = new Book[n];
    this.numberOfBookTitles=0;
  }

  public String toString() {
    String s=this.name + ":";
    for (int i=0; i<this.numberOfBookTitles; i++) {
      s = s + this.books[i];
    }
```

```
    return s;
  }

  public static void main(String[] arg) {
    BookStore bs = new BookStore("Bokhandeln",20);
    // Bok med identiteten 999-91-7000-150-5, antal ex 14, priset 500 kr
    Book b = new Book("999-91-7000-150-5",14,500.00);
    bs.addBook(b);
    System.out.println(bs);
    // Bok med identiteten 888-88-7000-150-7, antal ex 5, priset 400 kr
    b = new Book("888-88-7000-150-7",5,400.00);
    bs.addBook(b);
    System.out.println(bs);

    // Prisändring -10% (dvs prissänkning)
    double priceChange = -10.0;
    bs.changePrices(priceChange)
    System.out.println("Prisändring " + priceChange + "%");
    System.out.println(bs);

  }
} // Slut klassen BookStore
```

Skriv de båda metoderna **addBook** och **changePrices** som saknas i klassen **BookStore.** Metoden addBook skall lägga till en ny bok i affären. Metoden changePrices skall göra en prisändring av alla böcker. Prisändringen anges med ett procenttal. Om du tycker att det saknas en eller flera metoder i klassen klassen Book för att kunna lösa denna uppgift, så skriv i så fall även dessa metoder.

Resultatet skall bli följande när man kör mainmetoden ovan:

```
Bokhandeln:[999-91-7000-150-5,14,500.0]
Bokhandeln:[999-91-7000-150-5,14,500.0][888-88-7000-150-7,5,400.0]
Prisändring -10.0%
Bokhandeln:[999-91-7000-150-5,14,450.0][888-88-7000-150-7,5,360.0]
```

(9p)

Lösning:

```
public void addBook(Book b) {
   if (this.numberOfBookTitles<this.books.length)  {
      this.books[this.numberOfBookTitles]=b;
      this.numberOfBookTitles++;
   }
}

public void changePrices(double priceChange)  {
   for (int i=0; i<this.numberOfBookTitles; i++) {
      double price = this.books[i].getPrice();
      this.books[i].setPrice( price*(1+priceChange/100.0)),
   }
}
```

Metoden changePrices behöver kunna ändra priset på en bok. Det kan lösas med två metoder I klassen Book, nämligen getPrice och setPrice:

```
public double getPrice() { return this.price; }
public void setPrice(double newPrice) { this.price=newPrice;}
```

Alternativt med bara EN metod changePrice:

```
public void changePrice(double priceChange) {
   this.price=(1+priceChange)*this.price;
}
```

Uppgift 4

Antag att en individ har en identitet och ett antal egenskapsvärden, där varje sådant egenskapsvärde är ett heltal 0-9. Om vi har två individer kan vi beräkna hur bra dessa båda individer matchar med varandra med avseende på egenskapsvärdena. Ett exempel:
En individ A har följande egenskapsvärden: 9, 0, 0, 2, 3.
En individ B har följande egenskapsvärden: 9, 0, 0, 1, 6.
Matchningen beräknas genom att man summerar alla kvadrater på skillnaderna mellan värdena för individerna. En matchning mellan individ A och B ger *matchningsvärdet* 10 eftersom:
$(9-9)^2 + (0-0)^2 + (0-0)^2 + (2-1)^2 + (3-6)^2 = 1+9=10$

Givet är följande skal till en klass `Individ` som representerar en individ som har en identitet och ett antal egenskapsvärden.

```
public class Individ {


   public Individ(String ident, int[] values) {
       this.id=ident;
       this.featureValues = new int[values.length];
       for (int i=0; i<values.length; i++) {
          this.featureValues[i]=values[i];
       }
    }

   public String toString() {
      String s = "";
      for (int i=0; i<this.featureValues.length; i++) {
         s = s + this.featureValues[i] + " ";
      }
      return "<" + this.id + " " + s + ">";
   }

   public static void main (String[] arg) {
      int [] featureValuesA = {9,0,0,2,3};
      int [] featureValuesB = {9,0,0,1,6};
      Individ indA = new Individ("A",featureValuesA);
      Individ indB = new Individ("B",featureValuesB);
      System.out.println(indA);
      System.out.println(indB);
      int match = indA.matchingValue(indB);
      System.out.println("Matchningsvärdet är " + match);
   }
}
```

a) Skriv de instansvariabler som saknas (1p)

Lösning:
```
private String id;
private int[] featureValues;
```

b) Metoden **`matchingValue`** saknas. Skriv den metoden så att den passar mainmetoden och när den körs skall resultera i följande utskrifter:

```
<A 9 0 0 2 3 >
<B 9 0 0 1 6 >
Matchningvärdet är 10
```
(4p)

Lösning:

```
public int matchingValue(Individ annan) {
   int sum=0;
   for (int i=0; i<this.featureValues.length; i++) {
      sum = sum + (int) (Math.pow( this.featureValues[i]-annan.featureValues[i], 2));
   }
   return sum;
}
```

c) Är det *grundkopiering* eller *djupkopiering* som utförs i konstruktorn i klassen **`Individ`**? Svaret måste motiveras. (1p)

Lösning:

Det är djupkopiering. Motivering: Satsen

```
this.featureValues = new int[values.length];
```

skapar en array och elementen i arrayen `values` kopieras därefter till arrayen `this.featureValues`.

Om det hade varit frågan om grundkopiering hade konstruktorn bara bestått av en sats så här:

```
this.featureValues = values;
```

Därmed refererar de båda arrayerna till samma array.

Uppgift 5

Föregående uppgift handlade om att matcha invidividers egenskaper. Antag att individernas data finns i textfilen `individdata.txt`. Filen kan se ut så här:

```
kim.nilsson@pollax.se     5  1  3  1  8
fia.jonsson@pollax.se     4  6  4  5  3
bill.ek@pollax.se         2  9  8  9  9
ida.lonn@pollax.se        5  7  3  8  2
my.ehn@pollax.se          3  9  1  1  4
bo.gran@pollax.se         9  1  3  5  6
eva.ronn@pollax.se        3  3  0  5  3
dan.lind@pollax.se        8  8  7  9  1
```

Varje rad i filen består av data för en individ, först på raden finns identiteten, därefter följer fem egenskapsvärden. Filen kan bestå av ett godtyckligt antal individer.

Följande program utför en matchning av individerna, alla mot alla, som finns i filen `individdata.txt`:

```
import java.io.*;
public class Matching {

  public static void main (String[] arg ) throws IOException {
    Group g = new Group("individdata.txt");
    Individ[] best = g.bestMatch();
    System.out.println("Bästa match:" + best[0] + " och " + best[1]);
    System.out.println("Antal individer: " + g.getAntal() );
    System.out.println(g);
  }  // main

} // Matching
```

Om man kör mainmetoden fås följande resultat:

```
Antal individer: 8
<kim.nilsson@pollax.se 5 1 3 1 8 >
<fia.jonsson@pollax.se 4 6 4 5 3 >
<bill.ek@pollax.se 2 9 8 9 9 >
<ida.lonn@pollax.se 5 7 3 8 2 >
<my.ehn@pollax.se 3 9 1 1 4 >
<bo.gran@pollax.se 9 1 3 5 6 >
<eva.ronn@pollax.se 3 3 0 5 3 >
<dan.lind@pollax.se 8 8 7 9 1 >
Bästa match:<fia.jonsson@pollax.se 4 6 4 5 3 > och <ida.lonn@pollax.se 5 7 3 8 2 >
```

Programmet jämför alla par av individer och beräknar det par som matchar bäst, dvs har det lägsta matchningsvärdet (se föregående uppgift). Mainmetoden använder sig av klassen `Group` som finns på nästa sida, där uppgiften fortsätter.

Givet är följande skal till klassen **`Group`**:

```
import java.util.*;
import java.io.*;

public class Group {
  private ArrayList<Individ> individList;

  public Group (String filnamn) throws IOException {
    this.individList = new ArrayList<Individ>();
    File input = new File(filnamn);
    if (!input.exists()) {
      System.out.println("Filen '" + filnamn + "' existerar ej");
      return;
    }
    Scanner fsc = new Scanner(input);
    String id;
    int [] eg = new int[5];
    while (fsc.hasNextLine() ) {
      id = fsc.next();
      for (int i=0; i<5; i++) {
        eg[i]=fsc.nextInt();
      }
      Individ ind = new Individ(id,eg);
      this.individList.add(ind);
    }
    fsc.close();
  }

  public int getAntal() {
    return this.individList.size();
  }

  public  String toString() {
    String s ="";
    for (int i=0;i<this.individList.size();i++) {
      s = s + this.individList.get(i) + "\n";
    }
    return s;
  }

  // Metoden bestMatch saknas
  // ...
}  // Group
```

Skriv metoden `bestMatch` som saknas i klassen `Group`. Metoden skall beräkna det par av individer som matchar bäst. Antag att det ej finns fler än ett sådant par. Metoden skall passa mainmetoden i klassen **`Matching`** och så att resultatutskrifterna blir som i körexemplet på föregående sida . (5p)

Lösning:

Metoden skall returnera en Individ-array

```
public Individ[] bestMatch() {
    // Skapa en array för resultatet
    Individ [] best = new Individ[2];

    int index1=-1, index2=-1, m, min=10000000;

    // En dubbelloop för att beräkna alla kombinationer av individer
    // i = 0.. (antal individer minus ett)
    for (int i=0; i<this.individList.size()-1; i++) {
       // j = i+1 .. (antal individer)
       for (int j=i+1; j<this.individList.size(); j++) {
          // Beräkna matchvärde mellan i och j
          m = this.individList.get(i).matchingValue( this.individList.get(j));
          // Om detta värde är lägre än det lägsta
          if (m<min) {
             index1=i;  // "Bästa" paret av individer
             index2=j;
             min=m;
          }
       } // j-loopen
    } // i-loopen

    best[0]=this.individList.get(index1);
    best[1]=this.individList.get(index2);

    return best;
}
```