Tentamen Programmeringsteknik I 2014-10-17

Skrivtid: 0800-1300
Hjälpmedel: Java-bok

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 pin-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 idéer 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 tillåtet att lägga till egna metoder om så behövs. Sådana metoder skall dokumenteras och motiveras. De skall också ha en bra design och t ex inte förstöra objektens integritet.
- Det är totalt 30 poäng på skrivningen. Betygsgränser: 16 räcker säkert till 3, 22 säkert till 4, 27 säkert till 5.

Lycka till!

Torsten och Tom

Uppgifter

1. Nedanstående, inte helt korrekta, klass skall representera en tärning:

```
1  public class Dice {
2     private int numberOfSides;
3     private int value;
4
5     Dice(int nos) {
6        int numberOfSides = nos;
7        this.roll();            // random initial value
8     }
9
10    public void roll() {
11       this.value = Math.random()*this.numberOfSides + 1;
12    }
13
14    public int getValue() {
15       return this.value;
16    }
17
18    public static void main(String[] args) {
19       Dice d = new Dice(7);
20       System.out.println("Dice with " + 7 +" sides:");
21       for (int i=1; i<=20; i++) {
22          d.roll();
23          System.out.print(d.getValue() + ", ");
24          if (i%10 = 0) {    // line break after 10 values
25             System.out.println();
26          }
27       }
28    }
29 }
```

   En tärning har alltså två attribut: *antal sidor* som ges när tärningen skapas och ett *värde* tilldelas varje gång tärningen slås (metoden `roll`)

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

```
Dice with 7 sides:
1, 4, 3, 4, 4, 4, 3, 6, 1, 6,
1, 1, 7, 5, 7, 4, 1, 1, 4, 3,
```

   a) När programmet kompileras erhålles följande felutskrifter:

```
2 errors found:
File: Dice.java  [line: 11]
Error: Dice.java:11: possible loss of precision
found   : double
required: int

File: Dice.java  [line: 24]
Error: Dice.java:24: unexpected type
required: variable
found   : value
```

Vad är det som är fel och vad skall det stå?

> På rad 11 så är `this.vale` av typen `int` medan det som står till höger är av typen `double`. En sådan tilldelning är illegal. Högersidan måste explicit omvandlas till heltalstyp vilket görs med typomvandlingsoperatorn `(int)`:
>
> ```
> this.value = (int)(Math.random()*this.numberOfSides + 1);
> ```
>
> Observera att hela det gamla högerledet måste sättas inom parentes — annars verkar bara `(int)` på `Math.random` vilket alltid kommer bli 0.

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

```
Dice with 7 sides:
1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

Eftersom detta resultat erhålles varje gång programmet körs är slutsatsen att det måste vara något fel i koden. Vad är felet? Förklara resultatet! Rätta felet! (2p)

> Felet ligger på rad 6 (i konstruktorn). Eftersom det står `(int)` framför `numberOfSides` så blir detta en *lokal* variabel i konstruktorn som inte ändrar på *instansvariabeln* `numberOfSides`. Lösning är att ta bort `(int)`.
>
> Anmärkning: Om man missade att sätta parenteser kring det gamla högerledet i föregående uppgift så får man också svaret 1 *men* det räcker alltså inte som förklaring.

2. Nedanstående (del av) klass skall representera en mängd med 6-sidiga tärningar.

```
1  public class SetOfDice {
2     private Dice[] theDice;
3
4     public SetOfDice(int numberOfDice) {
5        this.theDice = new Dice[numberOfDice];
6     }
7
8     public String toString() {
9        String theString = "";
10       for (int i=0; i<numberOfDice; i++) {
11          theString = theString + this.theDice[i].getValue();
12          if (i<this.theDice.length-1) {
13             theString +=", ";
14          }
15       }
16       return "(" + theString + ")";
17    }
18
19    public static void main(String[] args) {
20       SetOfDice sod = new SetOfDice(10);
21       System.out.println(sod);
22    }
23 }
```

Antalet tärningar ges alltså när objekt skapas.

a) När klassen kompileras kommer följande felutskrift:

```
1 error found:
File: SetOfDice.java  [line: 10]
Error: SetOfDice.java:10: cannot find symbol
symbol  : variable numberOfDice
location: class Dice[]
```

Vad är felet och vad skall det stå?

> På rad 10 finns ingen `numberOfDice`. Den `numberOfDice` som finns i konstruktorn är ett namn på den formella parametern och den finns bara i konstruktorn. Problemet kan lösas på två sätt: Antingen deklarerar man en instansvariabel med namnet `numberOfDice` och ger den värde i konstruktorn:
>
> ```
> this.numberOfDice = numberOfDice
> ```
>
> eller så använder man arraylängden på rad 10:
>
> ```
> for (int i=0; i<theDice.length; i++)
> ```

b) När felet är rättat går programmet att kompilera men körning ger följande resultat

```
java.lang.NullPointerException
at SetOfDice.toString(SetOfDice.java:11)
at java.lang.String.valueOf(String.java:2826)
at java.io.PrintStream.println(PrintStream.java:771)
at SetOfDice.main(SetOfDice.java:21)
```

Vad är fel? Rätta felet så att programmet producerar en utskrift i stil med:

```
(2, 6, 1, 4, 5, 2, 1, 6, 2, 5)
```

Instansvariabeln `theDice` är en array som skapas i konstruktorn. Arrayens komponenter har dock inte skapas utan alla dessa blir satta till `null`. I konstruktorn måste man alltså gå igenom alla komponenter och tilldela dem ett `Dice`-objekt:

```
for (int i=0; i<numberOfDice; i++) {
   this.theDice[i] = new Dice(6);
}
```

3. Du skall skriva en klass `Article` som representerar en artikel i ett varulager. En vara skall ha ett artikel-id (heltal), ett antal (heltal) som anger hur många exemplar av denna vara som finns och ett pris (flyttal).

   Skriv klassen `Article` med följande

   a) Tre instansvariabler för artikel-id, antal och pris.

   b) En konstruktor som tar emot artikel-id, antal och pris

   c) En konstruktor som tar emot artikel-id och pris och som sätter antal till 0.

   d) *get*-metoder för antal och pris.

   e) En `toString`-metod. Se körexemplet nedan!

   f) En metod `setPrice(double newPrice)` som ger en vara ett nytt pris.

   g) En metod `addNumber(int number)` som ökar antalet exemplar med `number`

   h) En `public boolean equals(Article a)` som returnerar `true` om denna vara har samma idummer som varan `a`, annars `false`.

Testprogram: *Utskrifter:*

```
public static void main(String[] args) {
   Article a = new Article(2526, 10, 5.52);
   Article b = new Article(1357, 100, 565.25);
   Article c = new Article(1357, 44, 489);
   System.out.println("a: " + a);                      a: <2526, 10, 5.52>
   System.out.println("b: " + b);                      b: <1357, 100, 565.25>
   System.out.println("c: " + c);                      c: <1357, 44, 489.0>

   a.addNumber(5);
   System.out.println("a: " + a);                      a: <2526, 15, 5.52>

   System.out.println("a.equals(b): " + a.equals(b));  a.equals(b): false
   System.out.println("a.equals(c): " + a.equals(c));  a.equals(c): false
   System.out.println("b.equals(c): " + b.equals(c));  b.equals(c): true

   a.setPrice(a.getPrice()*2); // Double the price
   System.out.println("a: " + a);                      a: <2526, 15, 11.04>
}
```

(8p)

```
public class Article {
   private int id;
   private int number;
   private double price;

   public Article(int id, int number, double price) {
      this.id = id;
      this.number = number;
      this.price = price;
   }

   public Article(int id, double price) {
      this.id = id;
      this.number = 0;
      this.price = price;
   }

   public int getId() { // Efterfrågades inte här men behövs i uppgift 4
      return id;
   }

   public int getNumber() {
      return number;
   }

   public double getPrice() {
      return price;
   }

   public void setPrice(double newPrice) {
      price = newPrice;
   }

   public void addNumber(int n) {
      number += n;
   }

   public String toString() {
      return "<" + id + ", " + number + ", " + price + ">";
   }

   public boolean equals(Article a) {
      return this.id == a.id;
   }
```

4. Skriv klassen `Store` som lagrar ett antal `Article`-objekt. Ett `Store`-objekt skall ha ett namn (`String`) och ett varulager (en `ArrayList` med `Article`-objekt).

   Klassen skall ha följande:

a) En konstruktor som tar emot butikens namn och skapar ett (tomt) varulager.

```
public class Store {
   private String name;
   private ArrayList<Article> storage;

   public Store(String name) {
      this.name = name;
      this.storage = new ArrayList<Article>();
   }
```

b) En `print`-metod som fungerar enligt nedanstående körexemplet.

```
  public void print() {
     System.out.println("\nContents of " + this.name);
     for (Article a: this.storage) {
        System.out.println(a);
     }
  }
```

eller, med en annan variant av `for`-satsen:

```
     for (int i=0; i<this.storage.size(); i++) {
        System.out.println(a.get(i));
     }
```

c) En metod `int searchIndex(int id)` som returnerar index för vara med angivet artikel-id eller -1 om varan inte finns.

```
  private int searchIndex(int id) {
     for (int i=0; i<this.storage.size(); i++) {
        if (this.storage.get(i).getId()==id) {
           return i;
        }
     }
     return -1;
  }
```

d) En metod `int Article search(int id)` som returnerar varan med angivet artikel-

id eller `null` om varan inte finns.

```
public Article search(int id) {
   int index = this.searchIndex(id);   // Use the method above
   if (index == -1) {
      return null;
   } else {
      return this.storage.get(index);
   }
}
```

e) En metod `boolean addNewArticle(int id, int number, double price)` som lägger in en ny vara med angivet id, antal och pris. Varan skall läggas sist i varulagret. Om varan redan finns skall en utskrift göras och inget ändras. Om det gick bra att lägga in varan skall metoden returnera `true`, annars `false`.

```
public boolean addNewArticle(int id, int number, double price) {
   Article article = this.search(id);
   if (article!=null) {
      System.out.println("*** Already in storage: " + id);
      return false;
   } else {
      this.storage.add(new Article(id, number, price));
      return true;
   }
}
```

f) En metod `double totalValue())` som beräknar och returnerar det totala värdet av varulagret dvs summan av produkterna av varje artikels antal och pris.

```
public double totalValue() {
   double sum = 0;
   for (Article a: this.storage) {
      sum += a.getPrice()*a.getNumber();
   }
   return sum;
}
```

Se testprogrammet på nästa sida!

Testprogram: *Utskrifter:*

```
Store store1 = new Store("Lisas livs");
store1.addNewArticle(256, 3, 54.);
store1.addNewArticle(213, 7, 126.);
store1.addNewArticle(156, 18, 17.70);
store1.print();                                  Contents at Lisas livs
                                                 <256, 3, 54.0>
                                                 <213, 7, 126.0>
                                                 <156, 18, 17.7>

store1.addNewArticle(256, 30, 25);               Already in storage: 256

store1.print();                                  Contents at Lisas livs
                                                 <256, 3, 54.0>
                                                 <213, 7, 126.0>
                                                 <156, 18, 17.7>

System.out.println("Article 156: " +
                   store1.search(156));          Article 156: <156, 18, 17.7>
System.out.println("213 is at index " +
                   store1.searchIndex(213));     213 is at index 1
System.out.println("Total value: " +
                   store1.totalValue());         Total value: 1362.6
```

5. I föregående uppgift lades nya artiklar in sist i arraylistan. Vanligt är dock att man vill hålla listan sorterad.

   a) Skriv en metod `boolean insertNewArticle(int id, int antal, double price)` som fungerar som metoden `addNewArticle` ovan, men ser till att listan blir sorterad i ordning efter artikel-id. En förutsättning för att det skall fungera är alltså att man hela tiden använder den metoden för att lägga till nya varor.

   Så här kan en körning alltså se ut:

Testprogram: *Utskrifter:*

```
Store store2 = new Store("Arnes livs");
store2.insertNewArticle(5, 3, 25.50);
store2.insertNewArticle(2, 7, 126.);
store2.insertNewArticle(6, 18, 17.70);
store2.insertNewArticle(1, 30, 25);
store2.insertNewArticle(3, 9, 9);
store2.print();                                  Contents of Arnes livs
                                                 <1, 30, 25.0>
                                                 <2, 7, 126.0>
                                                 <3, 9, 9.0>
                                                 <5, 3, 25.5>
                                                 <6, 18, 17.7>

store2.insertNewArticle(3, 9, 9);                *** Already in storage: 3

store2.print();                                  Contents of Arnes livs
                                                 <1, 30, 25.0>
                                                 <2, 7, 126.0>
                                                 <3, 9, 9.0>
                                                 <5, 3, 25.5>
                                                 <6, 18, 17.7>
```

```
public void insertNewArticle(int id, int number, double price) {
   // Find the position
   int i = 0;
   while (i<storage.size() && id>storage.get(i).getId()) {
      i++;
   }
   // Check if it is a new article
   if (i==storage.size() || id<storage.get(i).getId()) {
      storage.add(i, new Article(id, number, price));
   } else {
      System.out.println("*** Already in storage: " + id);
   }
}
```

Observera att ordningen på testerna både i `while`- och `if`-satserna är väsentlig!

b) Skriv en metod `Store order()` som skapar ett nytt `Store`-objekt där varorna är sorterade. Koden

```
Store store3 = store1.order();
store3.print();
```

skall alltså skapa ett nytt `Store`-objekt med samma namn och samma varor som `store1` *men* där varorna är lagrade sorterade på artikel-id.

```
public Store order() {
  Store orderedStore = new Store(this.name);
  for (Article a: storage) {
     orderedStore.insertNewArticle(a.getId(), a.getNumber(), a.getPrice());
  }
  return orderedStore;
}
```