Tentamen Programmeringsteknik I 2013-06-14 med lösningar

1. Svara *kortfattat* på följande:

   a) Vad är en instansvariabel?

   > *Variabler som hör till enskilda objekt. Deklareras på översta nivån i klassen och i regel som* `private`*.*

   b) Vad är det för skillnad på *formella* och *aktuella* parametrar?

   > *Formella parametrar är de som deklareras i metodhuvudet. Dessa får värden från de aktuela parametrarna som anges vid metod- eller konstruktoranropet. En formell parameter är alltid en variabel medan en aktuell parameter är ett uttryck som evalueras till ett visst värde.*

   c) Nämn två andra typer av variabler (utöver instansvariabler och parametrar) och ge en *kort* beskrivning av dem.

   > *i. Lokala variabler som deklareras inuti en metod. Dessa kan bara användas i metoden och när metoden är klar "glöms" värdet bort.*
   >
   > *ii. Klassvariabler som deklareras med ordet* **static** *på översta nivån i klassen. Dessa hör inte till enskilda objekt utan är gemensamma för alla objekt i klassen. Refereras oftas via klassnamnet (inte via en objektreferens). Exempel:* `Math.PI`

   d) Hur gör man för att anropa en konstruktor?

   > *Med det reserverade* **new** *följt av klassnamnet och en lista med aktuella parametrar.*

   e) Vad menas med en *iteration*? Nämn två sätt att uttrycka iterationer i Java.

   > *En iteration (eller "repetition" eller "loop") är en upprepning av kod till något visst villkor är (eller inte är) uppfyllt. Javakonstruktioner för detta är satserna* `for`*,* `while` *och* `do`

   f) Vad anger ordet `void` i Java?

   > *Ordet* `void` *kan sättas som "typ" i en metoddeklaration och anger att metoden i fråga inte returnerar något värde.*

(6p)

2. Nedanstående klass representerar ett elektriskt motstånd:

```
public class Resistor {
  private double resistance; // resistans i Ohm
  private double maxPower;   // maximal tillåten effekt

  public Resistor(double resistance, double maxPower) {
    // Uppgift d)
  }

  public String toString() {
     // Uppgift a)
  }

  public double current(double u) {
    return u/resistance;
  }

  public double power(double u) {
    // Uppgift c)
  }

  public Resistor seriesConnection(Resistor m) {
    // Uppgift e)
   }

  public double maxVoltage() {
    // Uppgift b)
  }
```

Testprogram:

```
public static void main(String[] args) {
   Resistor r = new Resistor(10, 40);
   System.out.println("r: " + r);
   System.out.println("Ström vid 5V: " + r.current(5));
   System.out.println("Max spänning: " + r.maxVoltage());
   Resistor s = new Resistor(5, 30);
   System.out.println("s: " + s);
   Resistor t = r.seriesConnection(s);
   System.out.println("r och s i serie: " + t);
   r = new Resistor(-3, 5);
   System.out.println("r: " + r);
}
```

Utskrift:

```
r: <10.0, 40.0>
Ström vid 5V  : 0.5
Max spänning: 20.0

s: <5.0, 30.0>

r och s i serie: <15.0, 30.0>
Negativ parameter till Resistor
r:    Vad skrivs här? Uppgift f)
```

a) Skriv klart `toString`-metoden. Se körexemplen för specifikation!

```
public String toString() {
   return "<" + resistance + ", " + maxPower + ">";
}
```

b) Skriv klart metoden `maxVoltage()` som beräknar och returnerar den största tillåtna spänningen.

Formeln $w = u^2/r$ uttrycker sambandet mellan effekt, spänning och resistans. I formeln är w effekten i watt, u spänningen i volt och r motståndet i ohm.

```
public double maxVoltage() {
  return Math.sqrt(resistance*maxPower);
}
```

c) Skriv klart metoden `double power(double u)` som skall beräkna och returnera den effekt som utvecklas i motståndet vid spänningen `u`.

Även här kommer ovanstående formel väl till pass.

```
public double power(double u) {
   return u*u/resistance;
}
```

d) Skriv klart konstruktorn `Resistor(double resistance, double maxPower)`. Om båda parametrarna är icke-negativa skall dessa värden ges till instansvariablerna. Om ena eller båda parametrarna är negativ skall konstruktorn ge en felutskrift och inte göra någon tilldelning till instansvariablerna.

```
public Resistor(double resistance, double maxPower) {
  if (resistance<0 || maxPower<0) {
    System.out.println("Negativ parameter till Resistor");
  } else {
    this.resistance = resistance;
    this.maxPower = maxPower;
  }
}
```

e) Skriv klart metoden `Resistor seriesConnection(Resistor m)` som skapar ett nytt resistorobjekt med de egenskaper man får om man seriekopplar det egna objektet med objektet `m`. (Seriekoppling medför att de ingående resistanserna adderas. Den maximala effekten blir lika med den minsta av de ingående maxeffekterna.)

```
public Resistor seriesConnection(Resistor m) {
  double r = m.resistance + resistance;
  double maxPower = Math.min(this.maxPower, m.maxPower);
  return new Resistor(r, maxPower);
 }
```

f) Vad skrivs ut av den sista raden i `main`-metoden (dvs efter `r:`)? Motivera ditt svar!

När ett objekt skapas får instansvariablerna först sina default-värden (0. i detta fall). Om parametrarna till konstruktorn är felaktiga kommer inte dessa default-värden förändras så utskriften blir `<0., 0.>`

(12p)

3. En *prioritetskö* är en kö där uttagen styrs av en prioritet i stället för ankomstordning. Denna typ av köer är vanlig t ex vid akutmottagningar på sjukhus.

a) Skriv en klass `Patient` som skall representera en patient med namn (typ `String`) och prioritet (typ `int`). Klassen skall ha en konstruktor som tar emot namn och prioritet, en `toString`-metod och en `int getPriority()` som returnerar patientens prioritet. Se `main`-metoden och dess resulterande utskrift nedan!

```
public class Patient {
  private String name;
  private int priority;

  public Patient(String name, int priority) {
    this.name = name;
    this.priority = priority;
  }

  public String toString() {
    return "<" + name + ", " + priority + ">";
  }

  public int getPriority() {
    return priority;
  }
}
```

b) Skriv en klass `PriorityQueue` som representerar en prioritetskö. Klassen skall ha

- en konstruktor,
- en `toString`-metod,
- en metod `add(Patient p)` som lägger in en ny patient i kön samt
- en metod `Patient get()`. Metoden skall ta bort patienten med högst prioritet ur kön och returnera den som värde. Om flera patienter har den högsta prioriteten skall den som stått längst i kön väljas.

Klasserna du skriver i denna uppgift skall fungera ihop med nedanstående `main`-metod och ge angivet resultat:

```
public static void main(String[] args) {
  PriorityQueue pq = new PriorityQueue();
  pq.add(new Patient("Kalle", 5));
  pq.add(new Patient("Lisa", 10));
  pq.add(new Patient("Olle", 7));
  System.out.println(pq);
  System.out.println(pq.get());
  System.out.println(pq);
  pq.add(new Patient("Anna", 7));
  System.out.println(pq);
  System.out.println(pq.get());
  System.out.println(pq);
}
```

Utskrift:

```
[<Kalle, 5>, <Lisa, 10>, <Olle, 7>]
<Lisa, 10>
[<Kalle, 5>, <Olle, 7>]

[<Kalle, 5>, <Olle, 7>, <Anna, 7>]
<Olle, 7>
[<Kalle, 5>, <Anna, 7>]
```

(12p)

```
import java.util.ArrayList;

public class PriorityQueue {
  private ArrayList<Patient> theQ;

  public PriorityQueue() {
    theQ = new ArrayList<Patient>();
  }

  public void add(Patient p) {
    theQ.add(p);
  }

  public Patient get() {
    if (theQ.size() == 0) {
      return null;
    }
    Patient p = theQ.get(0);
    for (int i=1; i<theQ.size(); i++) {
      if (theQ.get(i).getPriority() > p.getPriority()) {
        p = theQ.get(i);
      }
    }
    theQ.remove(p);
    return p;
  }

  public String toString() {
    return theQ.toString();
  }
}
```