Tentamen Programmeringsteknik I 2013-06-14

Skrivtid: 1400-1700
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.

  Obs: *Dålig läslighet kan ge poängavdrag!*
- Det är totalt 30 poäng på skrivningen. Betygsgränser: 16 räcker säkert till 3, 22 säkert till 4, 27 säkert till 5.

Lycka till!

Tom

Uppgifter

1. Svara *kortfattat* på följande:
   a) Vad är en instansvariabel?
   b) Vad är det för skillnad på *formella* och *aktuella* parametrar?
   c) Nämn två andra typer av variabler (utöver instansvariabler och parametrar) och ge en *kort* beskrivning av dem.
   d) Hur gör man för att anropa en konstruktor?
   e) Vad menas med en *iteration*? Nämn två sätt att uttrycka iterationer i Java.
   f) Vad anger ordet `void` i Java?

   (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!

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.

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.

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.

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.)

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

(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!

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)