Tentamen Programmeringsteknik I 2014-03-21

Skrivtid: 1400-1900
Hjälpmedel: Java-bok

Tänk på följande

- Det finns en referensbok (Java) hos vakten 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: 15 räcker säkert till 3, 21 säkert till 4, 26 säkert till 5.

Lycka till!

Anna och *Tom*

Uppgifter

1. Betrakta följande klass:

```
public class Person {
   private String address;
   private String name;

   public Person(String n, String a) {
      String address = a;
      n = name;
   }

   public String toString() {
      return "Name: " + this.name + ", address: " + this.address;
   }

   public static void main(String[] arg) {
      Person p = new Person("Putte", "Storagatan 2");
      System.out.println(p.toString());
   }
}
```

   När programmet körs blir utskriften

```
Name: null, address: null
```

   a) Varför kommer inte det angivna namnet (”Putte”) och den angivna adressen (”Storgatam 2”) ut? (1p)

   b) Hur bör koden skrivas för att angivet namn och angiven adress skall skrivas ut? (1p)

Resten av skrivningen handlar om tre klasser som kan användas för att samla väderobservationer och producera statistik. Klasserna finns i bilagan men delar av koden är utelämnade.

Klassen `Observation` representerar en enskild observation bestående av uppgift om vindstyrka och temperatur (reella tal). (I praktiken skulle en observation naturligtvis bestå av många fler komponenter som klockslag, vindriktning, luftfuktighet, ...)

Klassen `Station` representerar en specifik väderstation med namn och en lista av observationer.

Klassen `StationList` innehåller en lista av stationer (`Station`-objekt).

2. I klassen `Observation`:

   a) Deklarera de instansvariabler som behövs i klassen för att representera en observation bestående av en uppgift om vindstyrka och en om temperatur.

   Skriv en konstruktor som tar emot dessa två värden.

   Skriv klart metoderna `getWind` och `toString`.

   Körexemplet visar vad `toString`-metoden skall producera. (2p)

   b) Skriv metoden `boolean belowTemp(double limit)` som returnerar `true` om observationens temperaturuppgift är under `limit`, annars `false`. (1p)

3. Klassen `Station` skall representera en väderstation med namn (en sträng) och ett antal `Observation`-objekt som skall lagras i en arraylist.

   a) Skriv de instansvariabler som behövs och skriv klart konstruktorn. (2p)

   b) Skriv klart metoden `int numberOfObservations()` som returnerar antal lagrade observationer. (1p)

   c) Skriv klart metoden `addObservation(Observation obs)` som lägger till en observation till listan med observationer. (1p)

   d) Skriv klart metoden `double meanWind()` som beräknar och returnerar medelvärdet av de lagrade vindobservationerna. (2p)

   e) För att ett vindkraftverk skall fungera får det varken blåsa för lite eller för mycket. Skriv en metod `double meanUsable(double low, double high)` som returnerar medelvinden för de vindobservationer som ligger i intervallet [`low`, `high`]. (3p)

   f) Skriv klart metoden `double[] windArray()` som skapar och returnerar en *array* med alla vindvärden från en station. (3p)

4. Klassen `StationList` representerar ett antal `Station`-objekt med hjälp av en `ArrayList`. Objekten är sorterade på namnen i bokstavsordning.

   Skriv klart metoden `Station addNameSorted(String name)` som letar i listan efter en station med angivet namn. Om stationen inte hittas skall den skapas och läggas in i listan så att den förblir sorterad på namn. Metoden skall returnera en referens till (det funna eller skapade) objektet. (4p)

5. I klassen `Observation` finns en metod `read` som används för inläsning av en observationspost dvs vind och temperatur. Metoden har en referens till ett `Scanner`-objekt som parameter. Om det som står på tur att läsas med med `Scanner`-objektet *inte* är ett tal returneras `null`. I annat fall läses vind- och temperaturuppgift och ett objekt med dessa skapas och returneras.

   Om det inte går att läsa en temperatur efter vinduppgiften så avbryts programmet med ett *RuntimeException* dvs med koden
   `throw new RuntimeException("Observation could not be read");`

   Se senare delen av `main`-metoden i klassen `Observation`

   a) Skriv klart metoden! (2p)

   b) Metoden `read` är deklarerad som `static`. Vad innebär det? (1p)

   c) Varför skriver man den inte som en ”vanlig” metod dvs utan `static`? (1p)

   d) Skriv en konstruktor som uppfyller samma specifikation eller förklara varför det är olämplig eller inte går. (1p)

6. Metoden `void load(String filename)` i klassen `StationList` skall läsa rader från en fil med uppgifter om stationer och observationsdata. Varje rad börjar med ett stationsnamn följt av ett antal observationsposter. Metoden skall använda `read` i `Observation` för att läsa enskilda observationer och metoden `addNameSorted` för att få in namn och observationer på rätt plats i listan. Om samma stationsnamn kommer igen skall dess observationer läggas till den redan lagrade stationen. I listan skall alltså ingen station förekomma flera gånger. Se körexempelet!

   Skriv klart metoden! (4p)