Tentamen Programmeringsteknik I 2014-06-12

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. Skriv en klass `Vehicle` som representerar fordonsobjekt i en trafiksimulering. Varje fordonsobjekt skall innehålla uppgift om *destination* (ett heltal), en *födelsetid* som anger när fordonet kommer in i systemet och en *sluttid* som anger när fordonet passerat trafiksystemet.

   Destination och födelsetid (båda av heltalstyp) skall ges som parametrar till konstruktorn.

   Eftersom sluttiden inte är känd när objektet skapas används metoden `setExitTime` för att ange den. Metoden skall kontrollera att sluttiden inte sätts till ett lägre värde än födelsetiden. Innan denna metod anropats skall sluttiden vara 0.

   Klassen skall vidare innehålla en metod `getDestination` som returnerar fordonets destination, en metod `getBornTime` som returnerar dess födelsetid och en metod `getSpentTime` som returnerar hur länge fordonet existerat dvs sluttid minus födelsetid.

   Slutligen skall klassen innehålla en `toString`-metod.

   Nedan följer ett litet testprogram med utskrifter som visar hur metoderna kan anropas. Din lösning skall fungera ihop med testprogrammet!

```
public static void main(String[] args) {
    int destination = 2;
    int time = 8;
    Vehicle v = new Vehicle(destination, time);
    System.out.println("Created vehicle v        : " + v);
    System.out.println("Destination              : " + v.getDestination());
    System.out.println("Born time                : " + v.getBornTime());
    v.setExitTime(15);
    System.out.println("v after setting exit time: " + v);
    System.out.println("Spent time               : " + v.getSpentTime());
    v.setExitTime(1);    // Illegal setting - should cause error message
 }
```

   Utskrifter:

```
Created vehicle v        : (2, 8, 0)
Destination              : 2
Born time                : 8
v after setting exit time: (2, 8, 15)
Spent time               : 7
java.lang.RuntimeException: exitTime must be > borntime
```

2. Klassen `VehicleCollection` innehåller en `ArrayList`. Klassen används för att samla fordon som passerat trafiksystemet och beräkna statistik på för dessa. Klassen har följande utseende:

```
public class VehicleCollection {

  private ArrayList<Vehicle>theCollection;
  ...

  public VehicleCollection() {
    ...
  }

  public void add(Vehicle v) {
    ...
  }


  public String toString() {
    return theCollection.toString();
  }


  /**
   * Computes the statistics for the stored vehicles i. e.
   * compute the mean and max value of time spent
   * and the number of vehicles using the max time
   **/
  public void computeStatistics() {
    ...
  }

  /**
   * Prints the statistics for the stored vehicles
   */
  public void print() {
     System.out.println("Number of vehicles           : " + theCollection.size());
     System.out.println("Mean time                    : " + meanTime);
     System.out.println("Max time                     : " + maxTime);
     System.out.println("Number vehicles with max time: " + numberMaxTime);
     System.out.println("The collection               : " + theCollection);
  }
```

   a) Deklarera de instansvariabler som behövs utöver arraylistan för att metoden `print` skall fungera.

   b) Skriv konstruktorn.

   c) Skriv metoden `add(Vehicle v)` som lägger till ett fordonsobjekt till samlingen.

   d) Skriv metoden `computeStatistics` som beräknar statistikvärden (medeltid, maxtid samt antal fordon som behövs maxtiden).

   På nästa sida ett testprogram och dess utskrifter

```
  public static void main(String[] args) {
     VehicleCollection vc = new VehicleCollection();
     Vehicle v = new Vehicle(1,1);
     v.setExitTime(3);
     vc.add(v);
     v = new Vehicle(0, 2);
     v.setExitTime(3);
     vc.add(v);
     v = new Vehicle(0, 4);
     v.setExitTime(6);
     vc.add(v);
     System.out.println("\nPrint before statistics is computed:");
     vc.print();
     vc.computeStatistics();
     System.out.println("\nPrint after statistics is computed:");
     vc.print();
  }
```

```
> run VehicleCollection

Print before statistics is computed:
Number of vehicles           : 3
Mean time                    : 0.0
Max time                     : 0
Number vehicles with max time: 0
The collection               : [(1, 1, 3), (0, 2, 3), (0, 4, 6)]

Print after statistics is computed:
Number of vehicles           : 3
Mean time                    : 1.6666666
Max time                     : 2
Number vehicles with max time: 2
The collection               : [(1, 1, 3), (0, 2, 3), (0, 4, 6)]
```

3. Klassen `TrafficSystem` definierar själva trafiksystemet (filer, rondeller, trafikljus, . . .). Det enda man behöver veta om denna klass är att när man skapar den så anger man hur många destinationer som förekommer och när man anropar metoden `step` avanceras systemet ett tidssteg.

   Metoden `step` returnerar ett fordon som passerat systemet (endast ett fordon per tidssteg kan komma ut) eller `null` om inget fordon kom ut detta tidssteg.

   Klassen `Simulation` har bara en `main`-metod som driver simuleringen genom att skapa systemet och tidsstega:

```
public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   System.out.print("Number of destinations: ");
   int numberOfDests = sc.nextInt();
   System.out.print("Number of time steps   : ");
   int numberOfTimeSteps =sc.nextInt();

   TrafficSystem ts = new TrafficSystem(numberOfDests);

   /*
    * Create and initialize an array of VehicleCollections with one vehicleCollection
    * for each destination
    */
    ...


   /*
    * Perform the specified number of time steps and store the vehicles coming out
    * in the designated vehicle collection (i.e. a vehicle with destination 0 should be
    * stored at index 0 in the array, a vehicle with destination 1 at index 1 etc
    */
   for (int i= 0; i<numberOfTimeSteps; i++) {
      Vehicle v = ts.step();

      ...

   }

   System.out.println("\nStatistics:");
   for (int i=0; i<numberOfDests; i++) {
      System.out.println("\nDestination '" + (char) (i + 'a') +"'");
      System.out.println("===============");

      out[i].computeStatistics();
      out[i].print();
      System.out.println("Lane: \n" + out[i].toString());
   }

   System.out.println("\nLeft in the system:");
   System.out.println(ts.toString());
}
```

Du skall komplettera `main`-metoden med följande:

a) Deklarera och initiera en array `out` med `VehicleCollection`-objekt. Det skall finnas ett `VehicleCollection`-objekt per möjlig destination.

b) Lägg till kod i tidsloopen som tar hand om de fordon som kommer ut ur systemet och lägger in dem i respektive `VehicleCollection`.

Din kod skall fungera i den angivna metoden!

Exempel på utskrifter från en körning finns på nästa sida.

```
Statistics:

Destination 'a'
===============
Number of vehicles           : 0
Mean time                    : NaN
Max time                     : 0
Number vehicles with max time: 0
The collection               : []
Lane:
[]

Destination 'b'
===============
Number of vehicles           : 3
Mean time                    : 1.3333334
Max time                     : 3
Number vehicles with max time: 1
The collection               : [(1, 3, 3), (1, 4, 5), (1, 5, 8)]
Lane:
[(1, 3, 3), (1, 4, 5), (1, 5, 8)]

Destination 'c'
===============
Number of vehicles           : 2
Mean time                    : 2.0
Max time                     : 2
Number vehicles with max time: 2
The collection               : [(2, 2, 4), (2, 7, 9)]
Lane:
[(2, 2, 4), (2, 7, 9)]

Left in the system:
<(1, 10, 0) >
```