Tentamen Programmeringsteknik I 2013-03-20

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 40 poäng på skrivningen. Betygsgränser: 18 ger säkert 3, 26 ger säkert 4, 34 ger säkert 5.

Lycka till!

Tom et al.

Uppgifter

1. Du skall skriva en klass som representerar ett enkelt *medelvärdesfilter*. Filtret skall lagra (upp till) ett visst givet antal tal. Varje gång man matar in ett tal får man tillbaka medelvärdet av de lagrade talen.

   Klassen skall heta `MeanFilter` skall ha följande:

   > En *konstruktor* `MeanFilter(int max)` som skapar ett filter som kan rymma upp till `max` tal.
   >
   > En metod `double add(double x)` som lagrar `x`. Om denna lagring medför att antalet lagrade tal överstiger `max` så skall det tal som legat längst tas bort. Metoden skall beräkna och returnera medelvärdet av de lagrade talen.
   >
   > En `toString`-metod. Se utskrifterna från testprogrammet nedan!

   Testprogram:

```
public static void main(String[] args) {
  MeanFilter mf = new MeanFilter(3);
  System.out.print  (mf.add(1.));
  System.out.println("  " + mf.toString());
  System.out.print  (mf.add(2.));
  System.out.println("  " + mf.toString());
  System.out.print  (mf.add(3.));
  System.out.println("  " + mf.toString());
  System.out.print  (mf.add(4.));
  System.out.println("  " + mf.toString());
  System.out.print  (mf.add(8.));
  System.out.println("  " + mf.toString());
}
```

   Utskrift från testprogrammet:

```
1.0  [1.0]
1.5  [1.0, 2.0]
2.0  [1.0, 2.0, 3.0]
3.0  [2.0, 3.0, 4.0]
5.0  [3.0, 4.0, 8.0]
```

   Skriv klassen!

   Tips: arraylist! (10p)

2. I bilagan finns ett program för att simulera trafikflödet i en korsning. På en del ställen är kod ersatt med tre punkter (. . .). Programmet är begränsat till att hantera en fil mot väster (”westbound”) och en fil mot norr (”northbound”).

   Korsningen kontrolleras av en signal som har tre möjliga lägen: släppa fram fordon mot väster (och ej mot norr), släppa fram fordon mot norr (och ej mot väster) och inte släppa fram fordon i någon riktning.

   Programmet har tre klasser: `Vehicle`, `Signal` och `TrafficSystem`.

   Klassen `Vehicle` lagrar ett fordon med identitet (`String`) och en födelsetid (`int`).

   Klassen `Signal` kan alltså tre lägen:

   a) Grön för fordon på väg mot väster (i `timeWest` tidsteg)

   b) Röd i alla riktningar (i `timeStop` tidsteg)

   c) Grön för fordon på väg mot norr (i `timeNorth` tidsteg)

   Signalen cirkulerar genom dessa tre tillstånd i denna ordning.

   Klassen `TrafficSystem` sätter ihop ett systemet av en signal och två köer. Klassen håller också reda på det fordon som just passerat korsningen (attributet `passed`).

Exempel på utskrifter från det färdiga programmet (metoderna `print` och `printStatistics`):

```
--------------------
            []
<w 0> (West)
            []
--------------------
            []
      (West)
            []
--------------------
            []
<w 2> (West)
            []
--------------------
            [<n 3>]
      (West)
            []
--------------------
            [<n 3>]
      (Stop)
            []
--------------------
            [<n 3>]
      (Stop)
            []
```

```
--------------------
            [<n 6>]
<n 3> (North)
            [<w 6>]
--------------------
            []
<n 6> (North)
            [<w 6>]
--------------------
            []
      (North)
            [<w 6>, <w 8>]
--------------------
            [<n 9>]
<w 6> (West)
            [<w 8>, <w 9>]

Statistik:

Antal bilar mot norr: 2
Genomsnittlig kötid : 2.0

Antal bilar mot väst: 3
Genomsnittlig kötid : 1.0
```

a) När programmet kompileras (när ... ersatts av korrekt kod) får vi följande kompilerinsgfel:

```
2 errors found:
File: /Users/tom/Dropbox/Prog1/2013-03-20/code/TrafficSystem.java  [line: 11]
Error: /Users/tom/Dropbox/Prog1/2013-03-20/code/TrafficSystem.java:11: cannot find symbol
symbol  : class ArrayList
location: class TrafficSystem
File: /Users/tom/Dropbox/Prog1/2013-03-20/code/TrafficSystem.java  [line: 12]
Error: /Users/tom/Dropbox/Prog1/2013-03-20/code/TrafficSystem.java:12: cannot find symbol
symbol  : class ArrayList
location: class TrafficSystem
```

Förklara vad felet beror på och hur det skall rättas! (2p)

b) När ovanstående fel är rättat går kompileringen bra men vid körning får man följande:

```
java.lang.NullPointerException
     at TrafficSystem.step(TrafficSystem.java:58)
     at TrafficSystem.main(TrafficSystem.java:109)
```

Förklara vad felet beror på och hur det skall rättas!
När detta är rätta så fungerar programmet. (2p)

c) Skriv klar `toString`-metoden i klassen `Vehicle` så att utskrifterna blir enligt körexemplet (2p)

d) Skriv metoderna `greenWest` och `greenNorth` i klassen `Signal` som returnerar `true` om fordon får köra mot väster respektive norr, annars `false`. (4p)

e) Skriv `toString`-metoden i `Signal` så att den fungerar enligt körexemplet. (2p)

f) Skriv metoderna `addNorthbound` and `addWestbound` (2p)

g) Deklarera de instansvariabler som behövs för att samla statistiken och skriv klart metoden `printStatistics` som skriver ut statistiken i enlighet med körexemplet. Du behöver inte skriva om metoden — det räcker med att ange vad ... skall ersättas med. Statistiken räknas enbart på fordon som har passerat korsningen. (6p)

h) Skriv klar metoden `step` i klassen `TrafficSystem`. Om signalen är grön för någon fil hämtas första fordonet från den kön (om det finns något i kön). Metoden skall också samla information för statistiken. Som framgår av körexemplet så räknas ingen kötid för fordon som anländer när signalen är grön. (6p)

i) Rita en bild över vilka objekt som finns vid sista tidssteget i körexemplet. Utgå från variabeln `ts` i `main`. Varje skapat objekt skall finnas med i figuren som en ”låda” och det skall framgå vilken referens som håller reda på den. (4p)