Bilaga B

Redovisning av inlämningsuppgifter

1 Rapportens innehåll

Varje inlämningsuppgift ska redovisas med en skriftlig rapport. Rapporten ska vara häftad och försedd med ett försättsblad med uppgiftens namn, inlämningsdatum, gruppmedlemmarnas namn, inskrivningsår och `student.lu.se`-e-postadresser.

Rapportens målgrupp är personer som kan programmera i Java men som inte känner till den aktuella uppgiften.

Språket i rapporten ska vara korrekt. De tankegångar som redovisas ska vara tydliga och lätta att begripa. Det finns bra regler för detta: en mening ska uttrycka en tanke, nästa mening ska följa av den första, när man börjar på en ny tankegång så börjar man ett nytt stycke, och så vidare.

Låt gärna någon annan granska rapporten och komma med synpunkter. Läs åtminstone själva igenom rapporten och försöka tänka er in i hur en läsare som inte känner till uppgiften uppfattar rapporten.

När ni lämnar in en ny version av en rapport som inte är godkänd så måste ni naturligtvis ha åtgärdat alla brister som rättaren har påpekat.

Följande avsnitt ska finnas i rapporten:

1 Bakgrund

Här ska ni ge tillräcklig bakgrund så att läsaren (tänk på målgruppen) kan förstå vad uppgiften rör sig om. Gör det med egna ord; skriv inte av uppgiftstexten. Det gäller att beskriva de mest väsentliga delarna av uppgiften — det är naturligtvis inte avsikten att allt som står i uppgiftstexten ska behandlas här.

Komplettera det som står i uppgiften med era egna synpunkter och insikter. Om ni har använt andra källor än uppgiftstexten så ska ni ge hänvisningar till dessa.

När man läst detta avsnitt ska man vara så väl insatt i problemet att man kan förstå er lösning.

2 Modell

Beskriv de klasser som finns i er modell. Beskriv sambanden mellan klasserna, vilka datastrukturer som används och vilken (viktig) information som skickas mellan klasserna. Använd gärna figurer för att förklara. Figurer ska förses med text som förklarar vad figuren beskriver.

Informationen i detta avsnitt ska underlätta studiet av programkoden. Det är inte meningsfullt att beskriva sådant som en läsare (som ju behärskar Java) direkt kan utläsa ur koden.

3 Brister och kommentarer

Kommentera sådant som kanske inte blev så bra och sådant som ni skulle gjort annorlunda om ni skulle gjort om uppgiften. Beskriv också sådant som kan verka som avsteg från uppgiften. Om ni har gjort sådana så måste ni motivera dem utförligt.

Om ni har konstruktiva synpunkter på uppgiften eller på uppgiftens formulering så bör ni ta upp dem i detta avsnitt.

4 Programlistor

Utskrift av alla klasser som ni skrivit. Varje klass och varje metod ska ha en dokumentationskommentar. Parametrarna till metoderna ska beskrivas, liksom metodernas resultat. Ni ska inte skriva kommentarer i programkoden. Om ni tycker att det är nödvändigt att kommentera programkoden så bör ni först undersöka om det inte är möjligt att förenkla koden så att den blir begriplig utan kommentarer.

Man kan naturligtvis tänka sig andra sätt att strukturera; detta är bara en rekommendation. Avsnittet ”Brister och kommentarer” behöver till exempel inte vara med om ni inte har några kommentarer.

2 Rapportens form

Rapporten ska vara snygg och välformaterad och framställd med lämpligt ordbehandlingsprogram. Vi rekommenderar starkt att ni använder LaTeX, men ni får använda valfritt program. Eftersom det blir ganska små rapporter så behövs ingen innehållsförteckning. Eventuella figurer ska vara ritade på dator. Samtliga sidor, utom försättsbladet, ska vara numrerade.

Använd breda marginaler som i LaTeX standardmall. Detta gör texten mera lättläst och underlättar för rättaren som skriver kommentarer i rapporten. Använd rak högermarginal.

Det är viktigt för läsbarheten att man väljer bra typsnitt. Programkod och specifikationer ska skrivas med tecken som alla är lika breda. I LaTeX bör du använda `VerbatimInput` (paketet `fancyvrb`) för att inkludera programkoden i rapporten (se datorlaboration 3 i Datorer och datoranvändning).

All programkod ska vara korrekt formaterad och indragningarna ska vara korrekta. Programrader ska få plats på en rad (cirka 70 tecken). Blir raderna längre måste ni bryta dem på lämpliga ställen och sedan göra en korrekt indragning av det brutna resultatet.

3 Exempel på rapport

I de följande avsnitten följer ett exempel på en rapport. Avsnittet Programlistor har förkortats för att spara plats. Försättsbladet ska vara en egen sida, men för att spara plats har vi här skrivit den som ett inledande avsnitt.

Försättsblad

Simulering av biltvätt

Inlämningsuppgift 0, Programmeringsteknik för D/C

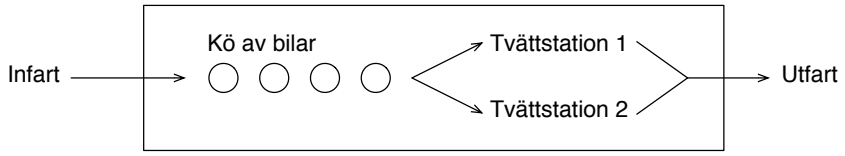
Författare: Xerxes Yngvesson, D14 (dat14xyn@student.lu.se)
Yasmin Kafai, C14 (dic14yka@student.lu.se)

Inlämnad: 2014–09–21

1 Bakgrund

Verksamheten vid en biltvättfirma som har två tvättstationer med något olika kapacitet ska simuleras. I den första tvättstationen tar en biltvätt 8 minuter, i den andra tar den 10 minuter. Bilarna anländer till anläggningen enligt en slumpmässig fördelning och ställer sig i en kö i väntan på att bli tvättade. Från kön körs bilarna in i en ledig tvättstation (om båda stationerna är lediga väljs den snabbaste). När en bil är färdigtvättad körs den ut ur tvättstationen och lämnar systemet.

Skiss över anläggningen:



Bilar anländer under 60 minuter. Simuleringen ska pågå så länge det finns någon bil i ankomstkön. Resultatet av simuleringen ska vara en tabell med bilnummer (1 för den första bilen som anländer, 2 för nästa, osv), ankomsttid, avgångstid och tvättstation för varje bil.

Bilarnas ankomster ska beskrivas av en Poisson-process (se till exempel Rydén och Lindgren, Markovprocesser, Lund 2000, eller annan lärobok i matematisk statistik).

2 Modell

Modellen av anläggningen innehåller följande klasser:

`SimWash`	Innehåller `main`-metoden som skapar de permanenta objekten och utför simuleringen. Simuleringstiden och medelvärdet för antalet bilar som ankommer per minut läses in från tangentbordet när programmet startas.
`CarWash`	Beskriver biltvätten. Dirigerar bilarna till och från tvättstationerna.
`WashingStation`	Beskriver en tvättstation. Håller reda på en bil som tvättas.
`Car`	Beskriver en bil. Samlar också statistik om bilen, till exempel ankomst- och avgångstid.
`Entry`	Infarten till biltvätten, där smutsiga bilar anländer (under simuleringen anländer inte bilarna utifrån, utan de skapas av denna klass).
`Exit`	Utfarten från biltvätten, där rena bilar lämnar systemet.
`Queue`	Kön i vilken bilarna väntar på tvättning. Kön är implementerad med hjälp av en `ArrayList`.
`Clock`	Håller reda på tiden i systemet, i minuter.

Simuleringen implementeras genom att programmet under hela simuleringstiden varje minut undersöker vad som ska ske. Följande händelser kan inträffa:

- En bil anländer till systemet. Detta hanteras av klassen `Entry`, som varje minut drar ett Poisson-fördelat slumptal som anger antalet bilar som anländer.
- En bil börjar tvättas (flyttas från kön till en tvättstation). Detta hanteras av klassen `CarWash`.
- En bil är färdigtvättad (flyttas från en tvättstation till utfarten). Också detta hanteras av klassen `CarWash`.
- En bil lämnar systemet. Detta hanteras av klassen `Exit`.

De nämnda "aktiva" klasserna har alla en metod `tick()` som anropas varje minut. Även klassen `Clock`, som håller reda på tiden i systemet, har en sådan metod.

Bilarna (`Car`-objekten) är helt passiva och skickas bara runt i systemet av de aktiva klasserna. Klasserna `Queue` och `WashingStation` är också passiva. Deras uppgift är att hålla reda på `Car`-objekt under tiden bilarna står i kö eller tvättas.

3 Brister och kommentarer

Eftersom utskriften för varje bil sker när bilarna lämnar systemet kan det inträffa att bilarna inte skrivs ut i nummerordning. Det kan till exempel se ut på följande vis (bilnummer, ankomstminut, avgångsminut, tvättstation):

```
4   5  24   2
5   7  25   1
7  21  33   1
6  17  34   2
```

Uppgiften har lösts enligt specifikationen med två tvättstationer i biltvätten. Det innebär dock inga större svårigheter att ändra lösningen så att den klarar av ett godtyckligt antal tvättstationer. Tiden som simuleringen ska pågå (i uppgiften var det 60 minuter) läses in när programmet startas, liksom medelvärdet för antalet bilar som ankommer per minut.

4 Programlistor

Klasserna finns i filer med samma namn som klasserna, till exempel finns klassen `SimWash` i filen *SimWash.java*. Samtliga filer finns i samma katalog som huvudprogrammet.

4.1 SimWash

```
import java.util.Scanner;

/** SimWash innehåller main-metoden i biltvättsprogrammet */
public class SimWash {
    /** Läser in simuleringstiden och antalet bilar som kommer per
        minut, utför simuleringen */
    public static void main(String[] args) {
        System.out.println("Hur länge ska simuleringen pågå?");
        Scanner scan = new Scanner(System.in);
        int stopTime = scan.nextInt();
        System.out.println("Hur många bilar kommer per minut?");
        double carsPerMinute = scan.nextDouble();
        Clock clock = new Clock();
        Exit exit = new Exit();
        CarWash carWash = new CarWash(clock, exit);
        Entry entry = new Entry(carsPerMinute, carWash);
        while (clock.getTime() < stopTime) {
            clock.tick();
```

```
            entry.tick();
            carWash.tick();
            exit.tick();
        }
        while (carWash.moreToDo()) {
            clock.tick();
            carWash.tick();
            exit.tick();
        }
    }
}
```

4.2 CarWash

```
/** Klassen CarWash beskriver tvättanläggningen */
public class CarWash {
    private Queue queue;               // kön av väntande bilar
    private WashingStation[] stations; // tvättstationerna
    private Clock clock;               // klockan
    private Exit exit;                 // utfarten

    /** Skapar en tvättanläggning med klockan clock och
        utfarten exit */
    public CarWash(Clock clock, Exit exit) {
        this.clock = clock;
        this.exit = exit;
        queue = new Queue();
        stations = new WashingStation[2];
        stations[0] = new WashingStation(8, clock);
        stations[1] = new WashingStation(10, clock);
    }

    /** Registrerar att bilen car ankommer, placerar den
        sist i kön */
    public void carArrives(Car car) {
        car.registerArrivalTime(clock.getTime());
        queue.insert(car);
    }

    /** Utför det som ska göras under aktuell minut: flyttar
        färdigtvättade bilar till utfarten, kör in väntande
        bilar i tvättstationerna */
    public void tick() {
        // Bilarna måste flyttas till utfarten innan man
        // kan köra in väntande bilar i tvättstationerna
        for (int i = 0; i < stations.length; i++) {
            if (stations[k].carReady()) {
                Car c = stations[i].getCar();
                c.registerExitTime(clock.getTime());
                exit.carLeaves(c);
            }
        }
        for (int i = 0; i < stations.length; i++) {
            if (!queue.empty() && stations[i].free()) {
                Car c = queue.first();
                queue.removeFirst();
                c.registerWashingStation(i);
                stations[i].wash(c);
            }
        }
    }
```

```
    /** Undersöker om det finns bilar i kön eller i någon av
        tvättstationerna */
    public boolean moreToDo() {
        return !(queue.empty() &&
                 stations[0].free() && stations[1].free());
    }
}
```

... osv, liknande programlistor för samtliga klasser.

4 Checklista

Kontrollera nedanstående punkter innan ni lämnar in en rapport.

	Uppgift 1	Uppgift 2
Texten är begriplig för personer i målgruppen		
Språket är korrekt och på rätt nivå för målgruppen		
Stavningen är korrekt		
Någon har läst rapporten och kommit med synpunkter		
Alla avsnitt finns med i rapporten		
Det finns korrekta hänvisningar till alla källor		
Avsnitten i rapporten är numrerade		
Uppgiftens namn, inlämningsdatum, våra namn, inskrivningsår och e-postadresser finns på försättsbladet		
Alla sidorna utom försättsbladet är numrerade		
Alla papper är ihophäftade		
Programmet löser den givna uppgiften enligt anvisningarna i uppgiften		
Varje klass och varje metod har en dokumentationskommentar		
Dokumentationskommentarerna ger all information som behövs		
Programlistorna och specifikationerna har rätt typsnitt		
Indragningarna i samtliga programlistor och specifikationer är korrekta		
Raderna i programlistorna är lagom långa		
Om det är en ny version av en inte godkänd rapport: alla brister som rättaren har påpekat har åtgärdats och den ej godkända rapporten bifogas		