

# Omkontrollskrivning, Programmeringsteknik för D/C

2014–01–17, 8.00–13.00

*Anvisningar:* Preliminärt ger uppgifterna  $12 + 18 + 6 + 4 = 40$  poäng. För godkänt betyg krävs ungefär 25 poäng. Tillåtet hjälpmedel: Java-snabbreferens.

Betyget på denna skrivning utgör betyget på kontrollskrivningarna, oavsett resultatet på tidigare skrivningar (dock sänks inte ett tidigare godkänt betyg). Resultatlistan anslås på institutionens anslagstavla när rättningen är klar.

1. En polygon är en geometrisk figur som består av rätta linjer mellan ett antal hörnpunkter. Ordningen mellan punkterna är väsentlig. En polygon kan beskrivas av följande klass:

Polygon

```
/** Skapar en polygon utan några punkter */
Polygon();

/** Läger till en punkt med koordinaterna (x,y) efter de punkter som
    redan finns */
void add(int x, int y);

/** Ritar polygonen i fönstret w. Bilden ska visa en sluten polygon, dvs
    det ska finnas en linje från den sista punkten till den första */
void draw(Window w);

/** Beräknar polygonens area */
double getArea();
```

Implementera klassen enligt följande anvisningar:

- Använd den färdigskrivna klassen Point för att hålla ordning på punkterna. Klassen har en konstruktor Point(int x, int y) och metoderna int getX() och int getY().
- Klassen Window har två operationer: moveTo(x,y) flyttar "pennan" till punkten x,y utan att rita, lineTo(x,y) flyttar pennan och ritar en linje.
- Vid beräkningen av arean förutsätts det att inga av polygonens linjer skär varandra. Då kan arean beräknas med följande formel (punkterna numreras  $0 \dots n-1$ ):

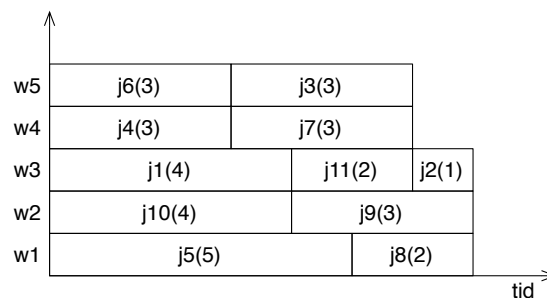
$$area = \frac{1}{2} \sum_{i=0}^{n-1} (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i)$$

Låt här  $x_n$  vara  $= x_0$  och  $y_n = y_0$ . Exempel för en triangel:

$$area = \frac{1}{2} (x_0 \cdot y_1 - x_1 \cdot y_0 + x_1 \cdot y_2 - x_2 \cdot y_1 + x_2 \cdot y_0 - x_0 \cdot y_2)$$

Arean kan bli positiv eller negativ beroende på punkternas orientering (motsols eller medsols), men getArea ska alltid ge ett positivt resultat (absolutvärdet av arean).

2. Ett antal arbetsuppgifter, "jobb", som tar olika lång tid ska fördelas på ett antal personer, "arbetare". Man vill göra ett schema för vem som ska göra vilket jobb på så sätt att alla jobb är klara så tidigt som möjligt. Exempel på ett schema med 5 arbetare (w1–w5) och 11 jobb (j1–j11, jobbets längd står inom parentes):



Ett enkelt sätt att lösa uppgifter av denna typ är att fördela jobben i avtagande ordning efter längd och att ge varje jobb till den arbetare som är först klar med sina jobb. Denna metod har använts för att skapa schemat i figuren. (Metoden ger inte alltid ett optimalt resultat, men är som sagt enkel.)

Jobben, arbetarna och schemat beskrivs av följande klasser (bara de operationer som behövs i denna uppgift finns med):

Job

```
/** Tar reda på jobbets längd */
int getLength();
```

Worker

```
/** Tar bort alla jobb från arbetaren */
void clearJobs();

/** Tar reda på den totala tiden för arbetarens alla jobb */
int getTotalTime();

/** Tilldelar jobbet j till arbetaren */
void assignJob(Job j);

/** Tar bort och returnerar nästa jobb som arbetaren ska utföra. Ger
    null om arbetaren är klar med alla sina jobb */
Job getNextJob();
```

Schedule

```
/** Skapar ett schema (utan några jobb) för arbetarna i vektorn workers */
Schedule(Worker[] workers);

/** Fördelar jobben i listan jobs på arbetarna. Listan är inte sorterad.
    Listan får ändras i metoden */
void makeSchedule(ArrayList<Job> jobs);

/** Tar bort och returnerar nästa jobb som arbetaren med nummer wNbr ska
    utföra. Arbetarna numreras med början på 1. Ger null om wNbr är
    felaktigt eller om arbetaren är klar med alla sina jobb */
Job getNextJob(int wNbr);
```

Implementera klassen Schedule. Klasserna Job och Worker är färdigskrivna.

3. Implementera klassen Worker från uppgift 2. Klassen ska ha en konstruktor utan några parametrar.

4. Studera följande felaktiga klass:

```
public class ListHandler {  
    /**  
     * Läger till talet nbr sist i listan nbrList om det inte redan finns  
     * där. Returnerar true om talet har lagts till, annars false.  
     */  
    public static boolean add(ArrayList<Integer> nbrList, int nbr) {  
        for (int i = 0; i < nbrList.size(); i++) {  
            if (nbrList.get(i) != nbr) {  
                nbrList.add(nbr);  
                return true;  
            }  
        }  
        return false;  
    }  
}
```

a) Antag att följande rader exekveras:

```
ArrayList<Integer> list = new ArrayList<Integer>();  
ListHandler.add(list, 1);  
ListHandler.add(list, 2);  
ListHandler.add(list, 3);  
ListHandler.add(list, 3);
```

Vad innehåller list när dessa rader exekverats? Motivera varför det blir så.

b) Antag att följande rader exekveras:

```
ArrayList<Integer> list = new ArrayList<Integer>();  
list.add(1);  
list.add(2);  
list.add(3);  
ListHandler.add(list, 3);
```

Vad innehåller list när dessa rader exekverats? Motivera varför det blir så och rätta till felet i metoden add.