

Lösningsförslag, omkontrollskrivning PTDC

2014-01-17

```
1. public class Polygon {
    private ArrayList<Point> points;

    public Polygon() {
        points = new ArrayList<Point>();
    }

    public void add(int x, int y) {
        points.add(new Point(x, y));
    }

    public void draw(Window w) {
        if (points.isEmpty()) {
            return;
        }
        w.moveTo(points.get(0).getX(), points.get(0).getY());
        for (int i = points.size() - 1; i >= 0; i--) {
            w.lineTo(points.get(i).getX(), points.get(i).getY());
        }
    }

    public double getArea() {
        double area = 0;
        for (int i = 0; i < points.size(); i++) {
            int next = (i < points.size() - 1) ? i + 1 : 0;
            area += points.get(i).getX() * points.get(next).getY()
                - points.get(next).getX() * points.get(i).getY();
        }
        return Math.abs(area) / 2;
    }
}

2. public class Schedule {
    private Worker[] workers;

    public Schedule(Worker[] workers) {
        this.workers = workers;
    }

    public void makeSchedule(ArrayList<Job> jobs) {
        for (int i = 0; i < workers.length; i++) {
            workers[i].clearJobs();
        }
        while (!jobs.isEmpty()) {
            Job longest = getLongest(jobs);
            Worker firstReady = findLeastOccupiedWorker();
            firstReady.assignJob(longest);
        }
    }
}
```

```

private Job getLongest(ArrayList<Job> jobs) {
    int maxLength = 0;
    int maxIndex = -1;
    for (int i = 0; i < jobs.size(); i++) {
        if (jobs.get(i).getLength() > maxLength) {
            maxLength = jobs.get(i).getLength();
            maxIndex = i;
        }
    }
    return jobs.remove(maxIndex);
}

private Worker findLeastOccupiedWorker() {
    int minTotalTime = Integer.MAX_VALUE;
    int minIndex = -1;
    for (int i = 0; i < workers.length; i++) {
        if (workers[i].getTotalTime() < minTotalTime) {
            minTotalTime = workers[i].getTotalTime();
            minIndex = i;
        }
    }
    return workers[minIndex];
}

public Job getNextJob(int wNbr) {
    if (wNbr < 1 || wNbr > workers.length) {
        return null;
    }
    return workers[wNbr - 1].getNextJob();
}
}

```

```

3. public class Worker {
    private ArrayList<Job> jobs;

    public Worker() {
        jobs = new ArrayList<Job>();
    }

    public void clearJobs() {
        jobs.clear();
    }

    public int getTotalTime() {
        int totalTime = 0;
        for (int i = 0; i < jobs.size(); i++) {
            totalTime += jobs.get(i).getLength();
        }
        return totalTime;
    }

    public void assignJob(Job j) {
        jobs.add(j);
    }

    public Job getNextJob() {
        return (!jobs.isEmpty()) ? jobs.remove(0) : null;
    }
}

```

4. a) Listan kommer att vara tom. `nbrList.size()` är = 0 och man går aldrig in i for-satsen.
- b) Listan kommer att innehålla {1,2,3,3}. De tre första add-satserna medför att talen 1,2,3 läggs in i listan. När `ListHandler.add(list, 3)` anropas kommer dubbletten 3 att läggas in i listan eftersom villkoret för inläggning är att det första elementet är skilt från `nbr`. Man måste gå igenom *hela* listan innan man kan avgöra att talet inte redan fanns. Så här ska den korrekta metoden se ut:

```
public static boolean add(ArrayList<Integer> nbrList, int nbr) {
    for (int i = 0; i < nbrList.size(); i++) {
        if (nbrList.get(i) == nbr) {
            return false;
        }
    }
    nbrList.add(nbr);
    return true;
}
```