Lösningsförslag, tentamen i Programmeringsteknik

2016–08–24

1. `// Totalt 8p`

```
public class WorkPeriod {
    private int start;
    private int finish;
    private String task;

    public WorkPeriod(String task, int hour, int length) { //Attribut och konstruktor 1p
        this.task = task;
        this.start = hour;
        this.finish = hour + length;
    }

    public boolean collidesWith(WorkPeriod wp) { //2,5p
        return (wp.start < finish && wp.finish > start);
    }

    public int compareTo(WorkPeriod wp) { //3,5p
        int timeResult = start - wp.start;
        if (timeResult == 0) {
            return task.compareTo(wp.task);
        } else {
            return timeResult;
        }
    }

    public String toString() { //1p
        return task + " " + start + "-" + finish;
    }
}
```

2.
```
public class WorkPeriodList {                                          // Totalt 10p
    private ArrayList<WorkPeriod> times;

    public WorkPeriodList() { //Attribut och konstruktor 1p
        times = new ArrayList<WorkPeriod>();
    }

    public void add(String task, int hour, int length) { //2p + sortering 5p
        WorkPeriod t = new WorkPeriod(task, hour, length);

        for (int i = 0; i < times.size(); i++) {
            if (times.get(i).compareTo(t) > 0) {
                times.add(i, t);
                return;
            }
        }
        times.add(t);
    }

    public WorkPeriod[] toSortedArray() { //2p (sortering kan ske här)
        WorkPeriod[] temp = new WorkPeriod[times.size()];
        for (int i = 0; i < times.size(); i++) {
            temp[i] = times.get(i);
        }
        return temp;
    }
}
```

3.
```
public class Worker {                                                  // Totalt 10p
    private String name; // arbetarens namn
    private WorkPeriod[] times; // de möjliga arbetstiderna
    private boolean[] scheduled; // håller reda på de tidsintervall personen ska arbeta

    public Worker(String name, WorkPeriod[] times) { //Attribut och konstruktor 3p
        this.name = name;
        this.times = times;
        scheduled = new boolean[times.length];
    }

    public String getName() { //0,5p
        return name;
    }

    public void schedule(int nbr) { //1p
        scheduled[nbr] = true;
    }

    public boolean isScheduled(int nbr) { //1,5p
        return scheduled[nbr];
    }

    public boolean canWork(int nbr) { //4p
        for (int i = 0; i < times.length; i++) {
            WorkPeriod p = times[i];
            if(scheduled[i] && p.collidesWith(times[nbr]) && i != nbr){
                return false;
            }
        }

        return true;
    }
}
```

4.
```
public class TimePlanner {                                          // Totalt 17p
    private WorkPeriod[] times;
    private ArrayList<Worker> persons;

    public TimePlanner(WorkPeriod[] times) { //Attribut och konstruktor 2p
        this.times = times;
        persons = new ArrayList<Worker>();
    }

    public boolean addWorker(String name) { //1,5p (se även findPerson nedan)
        if (findPerson(name) == null) {
            persons.add(new Worker(name, times));
            return true;
        } else {
            return false;
        }
    }

    public void scheduleWorker(String name, int nbr) { //4,5p (se även findPerson nedan)
        //Kolla om arbetspasset är ledigt
        for(Worker w : persons){
            if(w.isScheduled(nbr)){
                return;
            }
        }

        Worker p = findPerson(name);

        //Kolla att personen finns och att
        //passet inte kolliderar med annat pass för denna personen
        if (p != null && p.canWork(nbr)) {
            p.schedule(nbr);
        }
    }

    /** Frivillig hjälpmetod */
    private Worker findPerson(String name) { //3p (kan göras i addWorker och scheduleWorker)
        for (int i = 0; i < persons.size(); i++) {
            if (persons.get(i).getName().equals(name)) {
                return persons.get(i);
            }
        }
        return null;
    }

    public ArrayList<WorkPeriod> availableTimes() { //6p
        ArrayList<WorkPeriod> list = new ArrayList<WorkPeriod>();
        for (int i = 0; i < times.length; i++)  {
            boolean none = true;
            for (int k = 0; k < persons.size(); k++) {
                if (persons.get(k).isScheduled(i)) {
                    none = false;
                }
            }
            if (none) {
                list.add(times[i]);
            }
        }
        return list;
    }
}
```