

# Lösningsförslag, omkontrollskrivning PTDC

2012-08-23

```
1. public class Exam {
    private String course;
    private String date;
    private ArrayList<Problem> problems;
    private ArrayList<Student> students;

    public Exam(String course, String date, ArrayList<Problem> problems) {
        this.problems = problems;
        this.course = course;
        this.date = date;
        students = new ArrayList<Student>();
    }

    public void addStudent(Student student) {
        students.add(student);
    }

    public int maxPoints() {
        int points = 0;
        for (int i = 0; i < problems.size(); i++) {
            points += problems.get(i).maxPoints();
        }
        return points;
    }

    public void printProtocol(Student student) {
        System.out.println("Tentamen i " + course + " " + date);
        System.out.println("Skrivande: " + student.identity());
        MarkReport marks = student.marks();
        for (int i = 0; i < problems.size(); i++) {
            marks.printProtocol(problems.get(i));
        }
    }

    public void printAllProtocols() {
        for (int i = 0; i < students.size(); i++) {
            this.printProtocol(students.get(i));
        }
    }
}
```

```
2. public class Problem {
    private int number;
    private ArrayList<Item> items;

    public Problem(int number, ArrayList<Item> items) {
        this.number = number;
        this.items = items;
    }

    public int number() {
        return number;
    }

    public boolean equals(Problem other) {
        return number == other.number;
    }

    public int maxPoints() {
        int points = 0;
        for (int i = 0; i < items.size(); i++) {
            if (items.get(i).points() > 0) {
                points += items.get(i).points();
            }
        }
        return points;
    }

    public ArrayList<Item> itemsFor() {
        ArrayList<Item> temp = new ArrayList<Item>();
        for (int i = 0; i < items.size(); i++) {
            temp.add(items.get(i));
        }
        return temp;
    }
}
```

---

---

```

3. public class MarkReport {
    private ArrayList<Item>[] marks;

    public MarkReport(int nbrOfProblems) {
        marks = (ArrayList<Item>[]) new ArrayList[nbrOfProblems];
        for (int i = 0; i < nbrOfProblems; i++) {
            marks[i] = new ArrayList<Item>();
        }
    }

    public void add(Problem problem, Item item) {
        marks[problem.number() - 1].add(item);
    }

    public void printProtocol(Problem problem) {
        int nbr = problem.number();
        System.out.println("Uppgift " + nbr + ".");
        int totalPoints = 0;
        for (int i = 0; i < marks[nbr - 1].size(); i++) {
            Item current = marks[nbr - 1].get(i);
            int points = current.points();
            if (points > 0) {
                System.out.println(points + "/" + points + " p: "
                    + current.description());
            } else {
                System.out.println(points + " p: "
                    + current.description());
            }
            totalPoints += points;
        }
        ArrayList<Item> possibleMarks = problem.itemsFor();
        for (int i = 0; i < possibleMarks.size(); i++) {
            Item current = possibleMarks.get(i);
            if (current.points() > 0) {
                if (!this.contains(marks[nbr - 1], current)) {
                    System.out.println("0/" + current.points()
                        + " p: " + current.description());
                }
            }
        }
        if (totalPoints < 0) {
            totalPoints = 0;
        }
        System.out.println("Sammanlagt " + totalPoints + " poäng av "
            + problem.maxPoints() + " möjliga.");
    }

    private boolean contains(ArrayList<Item> items, Item item) {
        for (int i = 0; i < items.size(); i++) {
            if (items.get(i).equals(item)) {
                return true;
            }
        }
        return false;
    }
}

```

---

4. a) 

```
public class AnonymousStudent extends Student {
    private int seq;
    private String code;

    public AnonymousStudent(MarkReport marks, int seq, String code) {
        super(marks);
        this.seq = seq;
        this.code = code;
    }

    public String identity() {
        return seq + " - " + code;
    }
}
```

- b) Det behövs inga ändringar i klassen. I `printProtocol` anropas metoden `marks`, som finns i `Student`, och metoden `identity`, som är abstrakt och har olika implementeringar i `KnownStudent` och `AnonymousStudent`. Men i och med att metoden är abstrakt så väljs under exekveringen automatiskt rätt implementering beroende på objektets typ.