

Lösningsförslag, omkontrollskrivning PTDC

2014-04-29

```
1. public class GradeTable {
    private static final int aIndex = 2, cIndex = 1, eIndex = 0;
    private Pupil pupil;
    private boolean[][] table;

    public GradeTable(Pupil pupil, int nbrAspects) {
        this.pupil = pupil;
        table = new boolean[nbrAspects][3];
    }

    public Pupil getPupil() {
        return pupil;
    }

    public void mark(int a, char grade) {
        switch (grade) {
            case 'A': table[a][aIndex] = table[a][cIndex] = table[a][eIndex] = true; break;
            case 'C': table[a][cIndex] = table[a][eIndex] = true; break;
            case 'E': table[a][eIndex] = true; break;
        }
    }

    public boolean isMarked(int a, char grade) {
        int index = -1;
        switch (grade) {
            case 'A': index = aIndex; break;
            case 'C': index = cIndex; break;
            case 'E': index = eIndex; break;
        }
        return table[a][index];
    }

    public char getTestGrade() {
        int nbrAspects = table.length;
        int[] count = new int[table[0].length];
        for (int gradeIndex = 0; gradeIndex < count.length; gradeIndex++) {
            int nbr = 0;
            for (int i = 0; i < nbrAspects; i++) {
                if (table[i][gradeIndex]) {
                    nbr++;
                }
            }
            count[gradeIndex] = nbr;
        }
        char grade;
        if (count[aIndex] == nbrAspects) {
            grade = 'A';
        } else if (count[cIndex] == nbrAspects) {
            if (count[aIndex] >= nbrAspects / 2.0) {
                grade = 'B';
            } else {
                grade = 'C';
            }
        }
    }
}
```

```

    } else if (count[eIndex] == nbrAspects) {
        if (count[cIndex] >= nbrAspects / 2.0) {
            grade = 'D';
        } else {
            grade = 'E';
        }
    } else {
        grade = 'F';
    }
    return grade;
}
}

```

```

2. public class Test {
    private String name;
    private String date;
    private ArrayList<Aspect> aspects;
    private ArrayList<GradeTable> gradeTables;

    public Test(String name, String date, ArrayList<Aspect> aspects) {
        this.name = name;
        this.date = date;
        this.aspects = aspects;
        gradeTables = new ArrayList<GradeTable>();
    }

    public void addPupil(Pupil p) {
        gradeTables.add(new GradeTable(p, aspects.size()));
    }

    public void mark(String pNbr, int a, char grade) {
        GradeTable gt = findGradeTableFor(pNbr);
        gt.mark(a, grade);
    }

    public void printResults() {
        System.out.println(name + " " + date);
        for (int i = 0; i < gradeTables.size(); i++) {
            GradeTable gt = gradeTables.get(i);
            System.out.println(gt.getPupil().getName() + "\t"
                + gt.getPupil().getPNbr() + "\t" + gt.getTestGrade());
        }
    }

    private GradeTable findGradeTableFor(String pNbr) {
        for (int i = 0; i < gradeTables.size(); i++) {
            if (gradeTables.get(i).getPupil().getPNbr().equals(pNbr)) {
                return gradeTables.get(i);
            }
        }
        return null;
    }
}

```

```

3.    public Aspect getEasiestAspect() {
        int maxPoints = 0;
        int maxIndex = 0;
        for (int i = 0; i < aspects.size(); i++) {
            int sumPoints = 0;
            for (int j = 0; j < gradeTables.size(); j++) {
                GradeTable gt = gradeTables.get(i);
                if (gt.isMarked(j, 'A')) {
                    sumPoints += 3;
                } else if (gt.isMarked(j, 'C')) {
                    sumPoints += 2;
                } else if (gt.isMarked(j, 'E')) {
                    sumPoints += 1;
                }
            }
            if (sumPoints > maxPoints) {
                maxPoints = sumPoints;
                maxIndex = i;
            }
        }
        return aspects.get(maxIndex);
    }

4.    public class TestMain {
        public static void main(String[] args) {
            TestRegister tReg = new TestRegister();

            Scanner scan = null;
            try {
                scan = new Scanner(new File("testresult.txt"));
            } catch (FileNotFoundException e) {
                System.err.println("Cannot open: testresult.txt");
                System.exit(1);
            }

            String testName = scan.next();
            String testDate = scan.next();
            scan.nextLine();
            ArrayList<Aspect> aspects = new ArrayList<Aspect>();
            String line = scan.nextLine();
            Scanner aspectScanner = new Scanner(line);
            while (aspectScanner.hasNextInt()) {
                int aspectId = aspectScanner.nextInt();
                Aspect a = tReg.getAspect(aspectId);
                aspects.add(a);
            }

            Test test = new Test(testName, testDate, aspects);
            while (scan.hasNext()) {
                String pNbr = scan.next();
                Pupil p = tReg.getPupil(pNbr);
                test.addPupil(p);
                for (int i = 0; i < aspects.size(); i++) {
                    char grade = scan.next().charAt(0);
                    if (grade == 'A' || grade == 'C' || grade == 'E') {
                        test.mark(pNbr, i, grade);
                    }
                }
            }
            test.printResults();
            tReg.register(test);
        }
    }

```
