Faculty of Exact Sciences examination Programming
Vrije Universiteit December 14th, 2015 time: 2:45 hours

UITWERKINGEN

Problem 1.

a)
```
1 - 3 - 1
2 - 3 - 2
2 - 3 - 2
2 - 6 - 2
4 - 6 - 4
4 - 3 - 4
4 - 3 - 4
4 - 2 - 4
```

b)
```
double calculate (double x, int numberOfTerms) {
    double result = 0.0,
           power = x * x;
    int sign = 1,
        factor = 5,
        factorial = 1;

    for (int i = 0; i < numberOfTerms; i++) {
        double term = sign * factor * power / factorial;
        result += term;

        sign = -sign;
        factor += 1;
        power *= x;
        factorial *= (i+2);
    }

    return result
}
```

c)
```
static final int NUMBER_OF_ROWS = 9,
                 NUMBER_OF_COLUMNS = 7;
char[][] matrix = new char [NUMBER_OF_ROWS][ NUMBER_OF_COLUMNS];

void shift (int[][] m) {
    for (int i = 0; i < m.length; i++) {
        int first = m[i][0];
        for (int j = 1; j < m[0].length; j++) {
            m[i][j-1] = m[i][j];
        }
        m[i][m[0].length-1] = first;
    }
}
```

d)
```
-5 6
12 7
-5 -4
-4 -4
4 0
-3 -4
-2 -1
```

Problem 2.

a)
```
class University {
    static final int MAX_NUMBER_OF_STUDENTS = 4000;

    Student[] studentArray;
    int numberOfStudents;

    University () {
        studentArray = new Student[MAX_NUMBER_OF_STUDENTS];
        numberOfStudents = 0;
    }

    void add (Student student) {
        studentArray[numberOfStudents] = student;
        numberOfStudents += 1;
    }
}
```

b) Add to the class Student.

```
static final double TOP_AVERAGE = 9.0;

double average () {
    double sum = 0.0;
    for (int i=0; i<results.length; i++) {
        sum += results[i].grade;
    }
    return sum / results.length;
}

boolean topStudent () {
    return average() >= TOP_AVERAGE;
}
```

Add to the class University.

```
University topStudents () {
    University result = new University();

    for (int i=0; i<numberOfStudents; i++) {
        if (studentArray[i].topStudent()) {
            result.add(studentArray[i]);
        }
    }

    return result;
}
```

c) Add to the class Course.

```
static final double PASSED_GRADE = 5.5;

boolean passed () {
    return grade >= PASSED_GRADE;
}
```

Add to the class Student.

```
static final int DOING_OK_PERCENTAGE = 95;

int percentagePassedCourses () {
    int numberOfPassedCourses = 0;
    for (int i=0; i<results.length; i++) {
        if (results[i].passed()) {
            numberOfPassedCourses += 1
        }
    }
    return 100 * double(numberOfPassedCourses)/results.length;
}

boolean doingOK () {
    return percentagePassedCourses() >= DOING_OK_PERCENTAGE;
}
```

Add to the class University.

```
int doingOK () {
    int result = 0;

    for (int i=0; i<numberOfStudents; i++) {
        if (studentArray[i].doingOK()) {
            result += 1;
        }
    }

    return result;
}
```

d) Add to the class University.

```
int doingOkAndRegistered (int years) {
    return registered(years).doingOK();
}
```

Problem 3.

a)
```
String reverse (String s) {
    if (s.length() <= 1) {
        return s;
    }

    char firstChar = s.charAt(0);
    String rest = s.substring(1);
    return reverse(rest) + firstChar;
}
```

b)
```
String clean(String s) {
    if (s.length() < 2) {
        return s;
    }

    char firstChar  = s.charAt(0),
         secondChar = s.charAt(1);
    String rest = s.substring(1);

    if (firstChar == secondChar) {
        return clean(rest);
    } else {
        return firstChar + clean(rest);
    }
}
```