TECHNISCHE UNIVERSITEIT EINDHOVEN

Examination 2IP90
3 November 2015, 9:00–12:00 (12:30)

This exam consists of 3 questions on 4 pages.

- Put your name, number, and the date of today at the top of every Java file you submit. You only have to submit Java files. Do not use (named) packages for your code.
- Submit your solutions as compilable `.java` files in the folder `2IP90-submission` on your desktop (non-Windows users may choose a name themselves).
- **Before you leave the exam room, report to a supervisor to verify that your work has been submitted.**
- You are allowed to consult on your laptop the course material (lecture slides, reader, programs you have made during the course) and the Java API. Use of the internet is not allowed.

---

1 Miscellaneous *(28 pt)*

Submit your answers to these questions in a file `Miscellaneous.java`.

For answers that are not Java code but text, such as question 1.2 below, insert the text as comment (between `/*` and `*/`. Submit a compilable Java file.

Precede every answer with the number of the question in comment, e.g.,

```
/*
 * 1.1
 */ answer...
```

(7) 1. Describe in words what the following method returns

```
boolean m(int[] numbers) {
    boolean result = false;
    for (int i = 0; i < numbers.length; i++) {
        if (numbers[i] == 0) {
            result = true;
        }
    }
    return result;
}
```

(7) 2. There are two things wrong with the following code. Give the line number and explain what is wrong. (That there is no main method is not considered wrong.)

```
class Square {
    double size;

    void setSize( double s ) {
        s = size;
    }

    double getArea() {
        return size * size;
```

```
        }
    }

    class SquareDemo() {
        void demo() {
            Square mySquare;
            mySquare.setSize( 5 );
            System.out.println( mySquare.getArea() );
        }
    }
```

(7) 3. Give for both mistakes a line of code that should be replaced or added to repair the mistake.

(7) 4. Consider the following program.

```
class Halloween {
    String title;
    String[] pranks;

    void printPranks( int n ) {
        int x;

        for (int i=0; i<pranks.length; i++) {
            String p = grades[i];
            x = p.length();
            if ( p.length() >= n ) {
                System.out.print( p + " " );
            }
            System.out.println(i);
        }
    }
}
```

What are the local variables (not including parameters) of this program?

2 Make a difference *(36 pt)*

Extend the provided class Difference with the methods described below. In all these methods, you may assume that the parameter `nums` is not `null`.

Difference already contains a demo method for your, and the grader's, convenience. Leave this method as it is, possibly commenting out functions that you have not implemented. You may add other demo or test methods yourself. Submit the file `Difference.java`.

(5) 1. Write a method `boolean allZero( int[] nums )` that returns `true` if all elements of `nums` are zero; if `nums` has no elements, it should return `false`.

(5) 2. Write a method `int[] difference( int[] nums )` that returns the *difference array* of `nums`. This is an array that is one element shorter than `nums` and contains the differences between successive elements of `nums`. You may assume that `nums` contains at least two elements.

   For example, if `numbers` is the array {`1, 4, 9, 16`}, `difference(numbers)` should return the array {`3, 5, 7`}.

(5) 3. *Using the methods above*, write a method `boolean isConstant( int[] nums )` that returns `true` if the argument is a constant array, i.e., if all elements in `nums` have the same value. Both an empty array and an array with one element are considered constant.

(5) 4. *Using the methods above*, write a method `boolean isLinear( int[] nums )` that returns `true` if the elements are a linear function of the indices. Note that an array is linear if the difference array is constant.

(5) 5. Write a method `void printFunction( int[] nums )` that prints, assuming that `nums` is a linear or a constant array, the linear function that defines the elements.

For example, `printFunction( new int[]{1, 4, 7, 10} )` should print `3x + 1`.

(11) 6. The *degree* of an integer array is the number of times one can apply the difference operator until the array is constant. Hence, considering the definitions above, a constant array has degree 0 and a linear array has degree 1. A quadratic array, where the elements are a quadratic function of the indices, has degree 2, etc.

Write a *recursive* method `int degree( int[] nums )` that returns the degree of `nums` as defined above. You may assume that `nums` has at least one element.

Use only local variables and the parameter in this method. Do not use instance variables.

3 University *(36 pt)*

We are going to model in Java a university building with building users, lecture rooms, etc. Study the given files with the classes Room, Employee, BuildingTest, and the interface User. You have to submit these files and the files `Professor.java`, `Student.java`, `Building.java`.

(5) 1. Create a class Professor that is both an Employee and a building User. Professors are lecturers.

Override in Professor the method `public String toString()` of Employee. This method should give the data of an object in text form. The method `toString` is called when an object is printed. For a Professor, we want the method to give her name preceded by "`Prof. `". The name should be produced by the method `toString()` of Employee.

Note that you are not allowed to change the class Employee.

(5) 2. Create a class Student that is a building User. A Student is not an Employee nor a lecturer. A Student has a `String name` and an `int id` that represents the identification number of the student. Add a constructor that initializes these variables.

(5) 3. Add a Building class with

- instance variables `String name`, `Room hallway`, and `Room[] rooms` that is an array of all the rooms in the building, including the hallway,
- a constructor with two parameters: the name of the building and a non-empty array of rooms. The first element of this array is considered as the hallway.

(4) 4. Add a method `void moveUser(User user, Room room, boolean isEntering)` to the class `Building` that allows a user to move in or out of a room (whether it is in or out depends on the value of `isEntering`).

When `room` is the hallway, the building is left or entered (depending on the value of `isEntering`).

All rooms are connected to the hallway and not to each other.

You may assume that the method is sensibly applied, i.e., when it is called to move a user into a room, the user is not in that room and when it is called to move a user out of a room, the user is in that a room, etc.

(4) 5. Add a method `boolean hasLecturer()` to the class Room that returns `true` when there is a lecturer among the visitors of the room and `false` otherwise.

A lecture room (any room other than the hallway) may only be entered by a non-lecturer if there is already a lecturer in the room.

Adapt the method `moveUser` of Building to take this rule into account. If a move is not allowed because of this rule, the move should not be executed and the call to `moveUser` should have no effect.

(4) 6. Override the `toString()` method for the class Building. Use the `toString()` method of the Rooms, which in turn should use the toString() methods of the users.

Multiple choice questions

(3) 7. Suppose we want to have rooms with light switches that only allow users to enter when the light is on. Printing the building should also print for each room with a switch whether the light is on or not. Do we need to change the class Building?

(a) Yes, otherwise it can only have rooms without light switches.
(b) Yes, but only its toString() method.
(c) No, we can use a new class extending Room without changing Building.
(d) No, but we do need to make a new subclass of Building.

(3) 8. Suppose professor 'Kees' starts in the hallway and then moves to room 1. How will his name show up when room 1 is printed:

(a) "User @ 12345" (the number may vary) will be printed, because `moveUser` converts Kees into type User which is printed like this.
(b) "User (tu/e)" will be printed as Kees is a User but also still an Employee.
(c) "Kees" will be printed as Kees is a User and his name is known.
(d) "Prof. Kees (tu/e)"""" will be printed as Kees is still a Professor and an Employee.

(3) 9. "Lecture rooms that contain any user also contain a lecturer." In the answers below we mean with the initial state of an object the state of the object immediately after execution of the constructor.

(a) This sentence is always true.
(b) This sentence remains true if it is true for the initial state of the building
(c) This sentence is true for the initial state of the building but may become false.
(d) This sentence may become false even if it is true for the intial state of the building.

Good luck!

Grading: The total number of points achieved is the grade g for this examination. The final grade is the result of the following formula rounded to the nearest integer number: $0.6 \cdot g + 0.4 \cdot h$. Here g is the grade for this exam and h is the average of the 6 highest grades for the homework assignments (an assignment that was not submitted is graded with a 0). Furthermore, the grade g has to be at least 5.0 to pass.
