TECHNISCHE UNIVERSITEIT EINDHOVEN

Examination 2IP91 and 2IBP90
24 January 2017, 18:00–21:00/21:30

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.
- Add comments to your code only where clarification is needed.
- Don't make the lines in your code longer than 80 characters. Longer lines will mess up the layout of the printed program and make it harder to read for the graders.
- Before you submit your solutions, check that you have included all the files you want to submit and that you have saved them most recently.
- Submit your solutions as compilable `.java` files in the folder `2IP91submission` on your desktop (non-Windows users choose a name themselves).
- Do not use named packages for your code. Use only letters, digits, and underscores in your file names.
- **When 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. You are allowed to bring a printed copy of the reader to the exam.
- Use of the internet or other means of communication is *not* allowed during the examination.
- Note that the timer of the exam system tells you the time until *the extension time* will elapse. If you don't have a time extension, you have to submit before 21:00h.

---

1 Miscellaneous *(36 pt)*

Submit your answers in the provided file `Miscellaneous.java`. Make sure your code is compilable.

(6) 1. Is the following true or false (explain your answer briefly)? There can only one method named `donald` in a class.

(6) 2. When a variable `a` is declared with `A a` and a method `m` with `void m(B b)`, under what conditions will the call `m(a)` compile?

(8) 3. Write a method `double segsum(double[] a, int from, int to)` that returns the sum of segment of `a`, i.e., the sum of the elements of `a` between position `from` (including) and `to` (not including). E.g., when `a` is {`9, 8, 7, 6, 5, 4`}, `segsum(a, 2, 4)` will return 13.

   You may assume that `from` and `to` are not exceeding the bounds of the array (not smaller than 0 and not larger than its length) and that `from` is not greater than `to`.

(8) 4. Write a method `double segsumRobust(double[] a, int from, int to)` that does the same calculation, but now checks whether `from` and `to` are within their bounds as described above. If one of them or both are not, the exception `IllegalSegment` should be thrown. The Exception class is provided in the template code. The caller of the method will have to handle the exception.

(8) 5. Consider the following class.

```
class Inauguration {
    String president;
    String[] speech;
    long numberOfPeople;

    void printSpeech( int n ) {
        int x;

        for (int i=0; i<speech.length; i++) {
            String p = speech[i];
            x = p.length();
            numberOfPeople += numberOfPeople * x * 1000;
            System.out.println(p);
        }
    }
}
```

What are the local variables (not including parameters) in this class?

2 Upside down *(28 pt)*

Submit your code in the provided file `UpsideDown.java`. Make sure your code is compilable.

You may assume in all questions that the arguments to the methods are non-null.

(12) 1. Write a *recursive* method `String reverse(String s)` that returns the reversal of the `String` `s`. No loops are allowed. Do not use instance variables. A non-recursive version will earn you at most half the points.

   Examples:

   `reverse("papetrog")` returns `"gortepap"`

   `reverse("")` returns `""` (the empty String)

   `reverse("parterretrap")` returns `"parterretrap"`

   Some of the following operations on strings may be helpful (you probably don't need them all).

   `s.substring(start, end)` returns the substring of String `s` that starts at position `start` (an `int`) and ends just before `end` (another `int`)

   `s.charAt(n)` returns the character at position `n` (an `int`) in String `s`

   `s.length()` returns the number of characters in String `s`

(6) 2. A palindrome is a word or piece of text that is the same when read backwards. E.g., *parterretrap* is a palindrome, *papetrog* is not. The empty string is a palindrome.

   Write a method `boolean isPalindrome(String s)` that returns `true` if and only if `s` is a palindrome. Use the method of the previous question.

(10) 3. Add a method to the class UpsideDown `void upsideDown(String[] text)` that prints the elements of `text` in reverse order and each String reversed. If you have don't have answered question 2.1, you may leave the Strings as is, i.e., not reversed. A non-recursive version will earn you maximally 7 points.

   Example:
   when the array `{"madam", "I'm", "adam"}` is provided to `upSideDown`, it will print

```
mada
m'I
madam
```

3 Monsters *(36 pt)*

See the enclosed file `Monsters.java`. Submit your answers in this file. You may use separate files for additional classes if you like. Make sure your code is compilable.

When you run the program, you will see a window with a "monster" in it with a mouth that opens and closes.

(8) 1. Change the method `void addMonsters()` to add another monster, in the top right corner of the window. For each new class you define in the questions below, you will be asked to change `addMonsters` such as to also add a monster of the new class. You have only to submit the final version of this method.

(12) 2. Add a subclass `SeeingMonster` of `Monster` that represents monsters with two eyes (colored circles). Include a constructor that sets the eye color and the position of the monster. Avoid copying code from `Monster` by using `super`.

   Change `addMonsters` to add a `SeeingMonster` in the center of the window.

(8) 3. Add a subclass `MovingMonster` of `Monster` that represents monsters that move horizontally from left to right across the screen by themselves (use the Timer). Include a constructor to set the postion of the monster. Avoid copying code by using `super`.

   Change `addMonsters` to add a moving monster, starting in the bottom left corner of the window.

   Note that a MovingMonster will move off the screen when it reaches the edge. This is the intended behavior.

(8) 4. Add a subclass `BouncingMonster` that represents monsters that have instance variables `int speedX` and `speedY` that contain the speed of the monster in the horizontal and vertical direction respectively. When the monster reaches the edge of the panel, it should bounce back like a ball. I.e., the angle of entry should be equal to the angle of exit (you do not have to calculate angles, simple changes of `speedX` and/or `speedY` suffice).

   Include a constructor that sets the initial position and the horizontal and vertical speed of the monster. Avoid copying code by using `super`.

   Change `addMonsters` to add a bouncing monster starting at the bottom in the middle of the window, moving, initially, diagonally up and to the right.

Good luck!

Grading: The total number of points achieved divided by 10 is the grade g for this examination, maximized to 10. 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 grade for the homework assignments. The grade g has to be at least 5.0 to pass.