EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics & Computer Science

Final Exam 2IP90 and 2IBP90 (Programming), Tuesday, 22 January 2019, 18:00–21:00 (21:30)

This exam consists of 4 questions on 5 pages.

- Put your name, number, and the date of today at the top of *every 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.
- Submit your solutions as `.java` files in the provided folder `2IP90-submission` on your desktop (Windows users). Non-Windows users: make a folder named `2IP90-YourName` (and fill in your own name).
- Do not use named packages for your code.
- **When you leave the exam, 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.

Grading: The grade g for this examination is the total number of points achieved plus 2 divided by 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.

1 Miscellaneous (20 pt)

Submit your answers to these questions in the enclosed file answersMiscellaneous.java.

1. Mention all the local variables (not including parameters) in the following program. (6)

```
1 class Valentine {
2   String title;
3   String[] cards;
4   void printRoses( int n ) {
5     int x;
6     for (int i=0; i < cards.length; i++) {
7       String c = cards[i];
8       x = c.length();
```

```
            if ( c.length() >= n ) {
                System.out.print( c + "  " );
            }
            System.out.println(i);
        }
    }
}
```

.

2. Consider the following code. (6)

```
class ExceptionFragment {
    void fragment() {
        int x = 0;
        try {
            x += 1;
            m();
            x += 2;
        } catch (NullPointerException e) {
            x += 4;
        }
        System.out.println(x);
    }

    // this method throws a NullPointerException
    void m() {
      //...
    }
}
```

Suppose the method `fragment` is called and the method `m` will produce a `NullPointerException`. What will be printed?

a) 0 b) 1 c) 2 d) 3
e) 4 f) 5 g) 6 h) 7 i) nothing .

3. There are two things wrong with the following code (that there is no main method is not considered wrong). Give the line number and explain what is wrong. (8)

```
 1 class Square {
 2   double size;
 3   void setSize( double s ) {
 4     s = size;
 5   }
 6
 7   double getArea() {
 8     return size * size;
 9   }
10 }
11
12 class SquareDemo() {
13   void demo() {
14     Square mySquare;
15       mySquare.setSize( 5 );
16       System.out.println( mySquare.getArea() );
17    }
18 }
```

2 Array Fun (30 pt)

Put the methods of this question in a class `Statistics` and submit this code in a file named Statistics.java. In all questions below, you may assume that the argument `a` is not `null`.

1. Write a method `double sum(double[] a)` that returns the sum of the elements of array `a`. When the array is empty (has length 0), it should return 0. (10)
2. Write a method `double[] colsum(double[][] m)` that returns the sums of the columns of matrix `m` as an array. (10)

   of array `a`. You may assume that `a` is rectangular, i.e., all rows have the same length and all columns have the same length.

   For example, when we have

```
    double[][] matrix = {
      {1, 2, 3},
      {4, 5, 6},
      {-1, -2, -3},
      {2, 4, 6}
    };
```

   then `colsum(matrix)` should return an array containing the values 6.0, 9.0, 12.0.

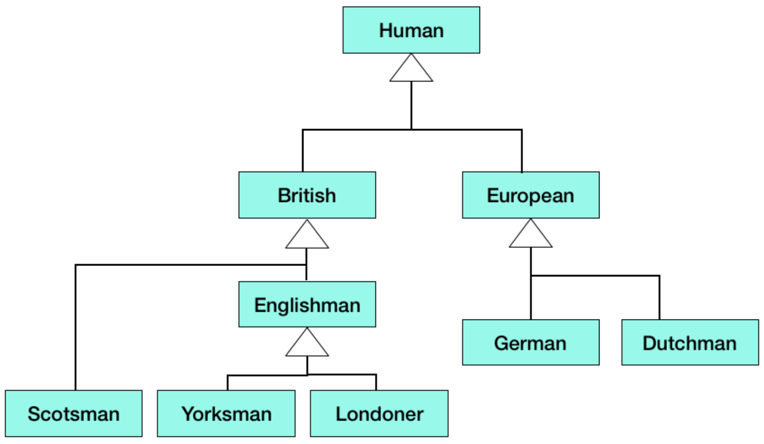
3. Write a *recursive* method `segsum(int[] a, int from, int upto)` that computes the sum of the elements of array `a` from (and including) `a[from]` upto (and not including) `a[upto]`. In other words, `segsum` sums the elements of the segment $[$`from`, `upto`$\rangle$. (10)

   You may assume that $0 \leq$ `from` $\leq$ `upto` $\leq$ `a.length`. The sum of an empty segment is 0.

   Examples:
   When `a` is {`1,2,3,4`},
   `segsum(a,0,4)` is 10
   `segsum(a,1,3)` is 5
   `segsum(a,0,0)` is 0

3 People (20 pt)

Consider the following class diagram.



1. Which of the following class declarations correspond to this class structure? (8)
   (a) `class Scotsman`
   (b) `class Londoner extends Englishman`
   (c) `class Human`
   (d) `class German extends Human`
2. Suppose all classes are declared according to the diagram above. Which of the following statements are correct Java? (12)
   (a) `Londoner lo = new Londoner();`
   (b) `Scotsman s = new Human();`
   (c) `European eur = new Dutchman();`
   (d) `Englishman eng = new Yorksman();`
   (e) `European eu = new British();`
   (f) `Human h = new Scotsman();`

4 Soccer (28 pt)

Submit your solution in the file `Soccer.java`. First copy the file to the submission folder, then start adding your answers to the copy. The file `Soccer.java` contains:

- a class Soccer that is a subclass of JPanel where balls and other shapes can be drawn and interacted with;
- an abstract class Ball that represents shapes on the screen that react when kicked (i.e., when the button Kick is pressed);
- a class BlueBall, subclass of Ball, that is represented by blue circles that move to the right when kicked.

The class contains a method `buildIt` that creates the GUI components necessary for the Soccer application and a method `createBalls` that creates the Ball objects. Initially, it creates one BlueBall in the middle of the screen.

There is a button Kick that, when pressed, should cause every ball on the screen to be kicked.

1. Add another BlueBall object in the bottom left of the window. Make sure that it reacts to kicking, just as the first BlueBall. (7)

2. Add a class PinkBall, subclass of Ball, where a ball is represented by a solid pink rounded rectangle that moves diagonally to the right and down when kicked. See the excerpt from the API on the class `Graphics` below on how to draw rounded rectangles. (7)

   Create a PinkBall object somewhere near the top left corner of the window.

3. Add a class DottedBall, *subclass of BlueBall*, where a ball is represented by a solid blue circle with a black dot in the center that, as a BlueBall, moves to the right when kicked and furthermore increases a little bit in size each time it is kicked. Do not unnecessarily copy code from BlueBall. You may want to use `super`. (7)

   Create a DottedBall in the middle of the top half of the window.

4. Add a class ChameleonBall, subclass of Ball. An object of this class does not move, but changes to a random color each time a kick occurs. (7)

   Create a ChameleonBall somewhere near the bottom left corner of the screen.

   Excerpt from the API description of the class `java.awt.Graphics`:

```
public abstract void fillRoundRect(int x,
                                   int y,
                                   int width,
                                   int height,
                                   int arcWidth,
                                   int arcHeight)
```

Fills the specified rounded corner rectangle with the current color. The left and right edges of the rectangle are at `x` and `x + width - 1`, respectively. The top and bottom edges of the rectangle are at `y` and `y + height - 1`.

Parameters:

`x` - the *x* coordinate of the rectangle to be filled.

`y` - the *y* coordinate of the rectangle to be filled.

`width` - the width of the rectangle to be filled.

`height` - the height of the rectangle to be filled.

`arcWidth` - the horizontal diameter of the arc at the four corners.

`arcHeight` - the vertical diameter of the arc at the four corners.

Good luck!