**EINDHOVEN UNIVERSITY OF TECHNOLOGY**
**Department of Mathematics & Computer Science**

**TU/e**

Final Exam 2IP90 and 2IBP90 (Programming), Tuesday, 29 October 2019, 9:00–12:00 (12:30)

This exam consists of 4 questions on 4 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 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 Miscellaneous.java.

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

```
1 class Jabberwock {
2   double[] toves;
3   long gimble;
4
5   void burble(int wabe) {
6     double slithy;
7     for (int gyre = 0; gyre < toves.length; gyre++) {
8       double jubjubbird = toves[gyre];
```

```
 9          slithy = Math.sqrt(jubjubbird);
10          if ( slithy >= gyre ) {
11            System.out.print( slithy + "  " );
12          }
13          System.out.println(gyre);
14        }
15    }
16  }
```

2. Assume the claas Gnome is defined. What is the number of objects (instances) of the class Gnome that is created when the method `create` is executed? (6)

```
1  void create() {
2    Gnome g = new Gnome();
3    Gnome h = new Gnome();
4    Gnome[] gnomes = new Gnome[5];
5    for (int i=0; i<gnomes.length; i++) {
6          gnomes[i] = g;
7    }
8  }
```

3. Describe what the following method returns. You may assume that `numbers` is not `null`. (4)

```
1  boolean m(int[] numbers) {
2    boolean result = false;
3    for (int i = 0; i < numbers.length; i++) {
4      if (numbers[i] == 0) {
5        result = true;
6      }
7    }
8    return result;
9  }
```

4. Suppose we want this method return `true` if all elements of `numbers` are 0 (including the case that `numbers` is empty) and `false` otherwise, what should change? (4)

## 2 Arrays and Strings (30 pt)

Use for this question the provided file ArrayQuest.java. In all questions below, you may assume that the array and String arguments are not `null`. A demo method is provided for your convenience.

1. Write a method `int[] clamp(int[] a, int max)` that clamps array `a`: each element in `a` that is greater than parameter `max` is replaced by `max`. If `a` is empty, an empty array should be returned. (10)

   Example: when `a` is {`1, 5, 2, 3, 6, 7`}, the call `clamp(a, 5)` will return {`1, 5, 2, 3, 5, 5`}.

2. Write a method `int maxpos(String s)` that returns a position of the greatest character in String `s` (according to character ordering, see note below). You may assume that `s` contains at least one character. Notes: (10)

- `s.charAt(n)` gives the character (type `char`) at postion `n`, where the first character in a String has position 0;
- characters can be compared with `>`, `>=` (further in the alphabet is greater);
- `s.length()` gives the number of characters of String `s`.

Examples: `maxpos("kazaa")` returns 2, `maxpos("lal")` returns 0 or 2 (both are correct).

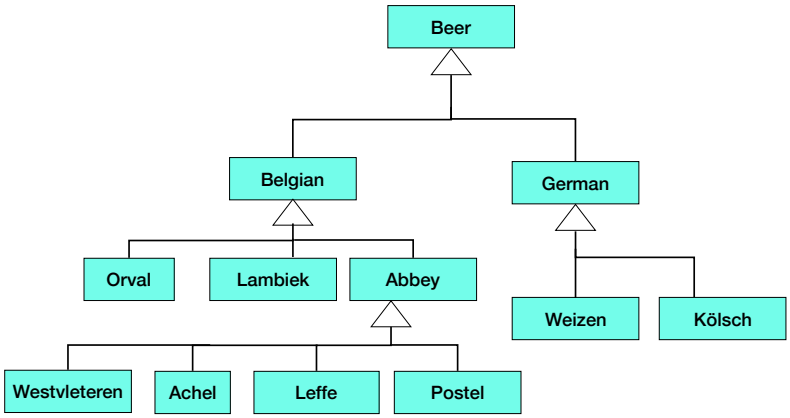
3. Write a *recursive* method `String sort(String s)` that returns a String with the characters of `s` in alphabetic order. Use the method `maxpos` above. Make a string from `s` without the largest character found with `maxpos` and sort this string recursively. (10)

   Do not use loops, do not use instance variables.

   Examples: `sort("yebab")` returns `"abbey"`, `sort("")` returns `""`.

## 3 Cellar (22 pt)

Give your answers to this question in the provided file Cellar.java. Gnome Rumpelstiltskin wants to keep track of his beer cellar using Java. For each type of beer in his collection he wants to have a class. He has come up with the following class diagram.




1. Which of the following class declarations correspond to the pictured class structure? More than one is possible as an answer. (8)

   (a) `class Lambiek`

   (b) `class Leffe extends Belgian`

   (c) `class Beer`

   (d) `class Weizen extends German`

2. Suppose all classes are declared according to the diagram above. Which of the following statements are correct Java? More than one is possible as an answer. (10)

   (a) `Achel ac = new Achel();`

(b) `Belgian s = new Achel();`

(c) `Weizen w = new German();`

(d) `s = new Lambiek();` (see declaration of `s` above)

(e) `Abbey ab = new Orval();`

3. Some beers satisfy the set of special requirements that give them the qualification of Trappist beer. Examples are Achel, Westvleteren, and Orval. Rumpelstiltskin likes to add this qualification to his class structure, but he can not add a class Trappist that is a subclass of Abbey, since not all trappist beers are of the abbey type. What can Rumpelstiltskin do to his class structure to give Achel, Westvleteren, and Orval the type Trappist and leave their relation to the other classes as it is? (4)

# 4 Farming (28 pt)

Consider the provided file Farming.java. When run, it will show a window with a button *Move* and the rear wheel of a tractor.

1. Change the `draw` method of Tractor and have it draw a complete tractor (add two rectangles and a front wheel). (5)
2. Add another Tractor to the scene. (5)
3. Add an instance variable `tractors` (array or ArrayList) that contains 100 Tractor objects at random positions. (5)
4. Add a method `void move()` to the class `Tractor` that changes the position of the tractor picture on the screen `moveDistance` pixels to the right. If the tractor is near the edge of the window, it may move out of sight. This is intended. (5)
5. Have clicking the button *Move* call the method `move` on all Tractor objects in `tractors`. The two Tractors that are not in the ArrayList will stay where they are. (5)

   Hint: implement the ActionListener in FarmPanel, not in Farming.
6. Add a button *Solve* in the bottom of the window that removes about half of the tractors from the ArrayList and from the screen. (3)

---

*Good luck!*