**Graphical object-oriented and internet programming in Java**



**Graduate Trainee assignment 2021-2022**

**Introduction**

This assignment asks that you demonstrate an understanding of simple graphics, inner classes, events and the *ActionListener* interface.

**IMPORTANT NOTE:** Please use at least Java 8 to compile and test your programs. Later versions of Java are also fine.

**Electronic files you should have:**

 *RandomRectangleGUI.java*

**What you should hand in: very important**

At the section there is a list of files to be handed in. Class files are **not** needed, and any candidate giving in only a class file will not receive any marks for that part of the

assessment, **so please be careful about what you upload as you could fail if**

**you submit incorrectly**.

 Please only hand in the files asked for, and not any additional files.

 Please put your name as a comment at the top of each Java file that you hand in.

**Marks for submitting files as requested**

There is one mark for naming any Java classes that you have to amend or write (in this assignment *RandomRectangleGUI* and *Test.java*) ***exactly*** as you have been asked to name them.

There is one mark for making sure that there is no clash between your Java file names and class names, such that the assessor can compile and run your classes. Sometimes candidates add identifying information to their Java files, meaning that the class name and the file name differ and the file will not compile. For example:

 *JSmith\_ RandomRectangleGUI.java*

 *cwk1- RandomRectangleGUI.java*

 *JSmith-assignment- RandomRectangleGUI.java*

In other cases candidates are careless about file names, forgetting that Java expects the file name and the class name to match when the class is public. For example:

 *RandomRectangleGUI .java* (note space before *.java*)

 *test.java* (Java is case sensitive; if the class is called *Test*, as it should be, there will be a compilation error if you use a lower case ‘t’ in the file name).

**No marks for programs that will not compile**

The assessors intend to compile and run your Java programs; for this reason programs that do not compile will not receive any marks.

If your file does not compile ***for any reason*** (except file name/class name mismatch) you will receive no marks for that part of the assignment. In particular, files that contain Java classes that cannot be compiled because they are the wrong type (e.g. PDFs), will not be given any marks.

**Assignment Description**

**Readable code**

This assignment is following the advice given by Robert C Martin in his book *Clean Code: A Handbook of Agile Software Craftsmanship* (published in 2008 by Prentice Hall, ISBN 978-0132350884). Martin describes a system for writing readable code, there are others, but this is the one that this assignment will be focusing on.

Martin writes:

One difference between a smart programmer and a professional programmer is that the professional understands that *clarity is king*. [...] We want to use the popular paperback model whereby the author is responsible for making himself clear and not the academic model where it is the scholar’s job to dig the meaning out of the paper.

Martin writes that ‘making your code readable is as important as making it executable’. He believes that names of variables, methods and classes are a major part of what makes code readable:

The name of a variable, function or class should answer all the big questions. It should tell you why it exists, what it does, and how it is used. If a name requires a comment, then the name does not reveal its intent.

Martin dislikes comments, noting that as code is updated comments are rarely updated at the same time, so however helpful a comment may be at the start, once a class has been in use for a while any comments are likely to be outdated and confusing. He believes that code should be written with names that make the intent clear, such that comments are redundant.

See the appendix for an example of renaming a simple class to make it more readable. When answering questions in this coursework assignment, please remember the following

rules from Martin:

**Formatting**

“You should take care that your code is nicely formatted. You should choose a set of simple rules that govern the layout of your code, and then you should consistently apply those rules. […] It helps to have an automated tool that can apply those formatting rules for you.”

**Methods**

 **Method should do one thing only**. If your method does more than one thing, break it into separate methods.

 **Do not repeat yourself** – if you find yourself writing the same code more than once, put it into a method.

 **Too many arguments (parameters).** “No argument is best, followed by one, two and three. More than three is very questionable and should be avoided with prejudice.”

**Comments**

If you do write a comment, make sure it is grammatical, short, does not state the obvious, and is really needed.

**Names**

 **Choose descriptive names** "Names in software are 90% of what makes software

readable”

 **Unambiguous names** “choose names that make the workings of a function or variable unambiguous”.

 **Names should describe side effects**, e.g. a method *getOos()* that will make an ObjectOuputStreamif one does not already exist should be called *createOrReturnOos()*

**General**

 **Obscured intent** – make the code as expressive as possible such that its intention is clear from a first reading.

 **Put conditional statements into a method to make their intention and effect clear**, e.g.

**BAD** if (guessedWord.length()< shortestLength)

**GOOD** if(guessedWordIsTooShort(guessedWord))

**Assignment**

Compile and run the *RandomRectangleGUI* class.

You should see a black rectangle in a JFrame, together with two buttons. If you press the top button (“Click me for a random colour”) nothing happens. If you press the bottom button ("Click me for a random size”) nothing happens.

Please complete the following tasks:

1. Write an inner class to implement the ActionListener interface and listen to the top button. Call your inner class *RandomColorListener.* Make any necessary changes such that when the top button is pressed the colour of the rectangle changes to a random colour. The user can repeatedly press the button for a new random colour.

You should write your *RandomColorListener* class to be as

short as possible. **[10 marks]**

2. Write an inner class to implement the ActionListener interface and listen to the bottom button. Call your inner class *SizeListener.* Make any necessary changes such that when the bottom button is pressed the size of the rectangle changes randomly. The user can repeatedly press the button for a new random size.

You should write your *SizeListener* class to be as short as

possible. **[10 marks]**

***Make sure that you read the*** *RandomRectangleGUI* ***class carefully before you start work****.*

**Deliverable for Assignment**

 An electronic copy of your revised program: *RandomRectangleGUI.java*

**Appendix**

A simple example of renaming methods and variables for greater readability.

***Original***

public class Calculator{

public static void calc(int x){

if (x >= 70){ System.out.println("grade = A"); return;

}

if (x >= 60){ System.out.println("grade = B"); return;

}

if (x >= 50){ System.out.println("grade = C"); return;

}

if (x >= 40){ System.out.println("grade = D"); return;

}

if (x<40) System.out.println("grade = F");

}

public static void main(String[] args) {

calc(90); calc(53); calc(30);

}

}

***Renamed***

public class GradeCalculator {

public static void calculateAndPrintGrade(int finalMark){

if (finalMark >= 70){ System.out.println("grade = A"); return;

}

if (finalMark >= 60){ System.out.println("grade = B"); return;

}

if (finalMark >= 50){ System.out.println("grade = C"); return;

}

if (finalMark >= 40){

System.out.println("grade = D");

return;

}

if (finalMark<40) System.out.println("grade = F");

}

public static void main(String[] args) { calculateAndPrintGrade(90); calculateAndPrintGrade(53); calculateAndPrintGrade(30);

}

}

**[END OF ASSIGNMENT]**