University of London International Programmes

Computing and Information Systems/Creative Computing

CO2220 Graphical Object-Oriented and Internet programming in Java

Coursework assignment 1 2017–18

Introduction
This is Coursework assignment 1 (of two coursework assignments) for 2017–18. Part A asks that you demonstrate an understanding of simple graphics, inner classes, events and the *ActionListener* interface. Part B looks at the object-oriented programming paradigm: classes, constructors; inheritance; instance and static utility methods plus getting our classes to work together to produce a desired result.

IMPORTANT NOTE: Please use the most recent version of Java, Java 8.

Electronic files you should have:

Part a
- *CirclesGUI.java*

Part b
- *Dictionary.java*
- *DictionaryEditor.java*
- *DictionaryFileManager.java*
- *DictionaryUI.java*
- *DictionaryWord.java*
- *ThesaurusFileManager.java*
- *ThesaurusWord.java*
- *Word.java*
- *dictionary.txt*
- *thesaurus.txt*

What you should hand in: very important
There is one mark allocated for handing in uncompressed files – that is, students who hand in zipped or .tar files or any other form of compressed files can only score 49/50 marks.

There is one mark allocated for handing in just the .java files asked for, without putting them in a directory; students who upload their files in directories can only achieve 49/50 marks.

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

At the end of each section there is a list of files to be handed in – **please note the hand-in requirements supersede the generic University of London instructions**. Please ensure you hand in **electronic versions** of your .java files since you cannot gain any marks without handing in the .**java** files asked for. Class files are **not** needed, and any student handing in only a class file will **not** receive any marks for that part of the coursework assignment, **so please be careful about what you upload as you could fail if you submit incorrectly**.

Programs that do not compile will not receive any marks.

The examiners will compile and run your Java programs; students who hand in files containing Java classes that cannot be compiled (*e.g.* PDFs) will not be given any marks for

that part of the assignment.

Please put your name and student number as a comment at the top of each .java file that you hand in.

You are asked to give your classes certain names, please follow these instructions carefully. If your file does not compile ***for any reason*** you will not receive any marks for that part of the assignment.

For example, file names that are different from the names of your classes (*e.g. cwk1-partA.java* file name versus *CirclesGUI* class name) mean that your program will not compile because of the conflict between the file name and the class name. You would therefore receive no marks.

PART A

Compile and run the *CirclesGUI* class. You should note that it uses an inner class to place a *JPanel* onto a *JFrame*. A red circle is drawn on the *JPanel*.

Using your mouse, click anywhere on the panel – you should find that the red circle is redrawn, with the co-ordinates given by where you clicked the mouse, and size given by the diameter variable. You will notice that the co-ordinates for the circle are actually the co-ordinates of the top left corner of an imaginary square, which the circle is drawn inside of. You can see this most clearly if you change:

g2.**fillOval**(X,Y,diameter,diameter); to
g2.**fillRect**(X,Y,diameter,diameter);

Compile and run the program again, and you will clearly see that the square is drawn with its top left corner placed where you click the mouse.

The file that you have been given implements the *MouseListener* interface, so that the *MouseClicked(MouseEvent)* method can be used to find the coordinates of your mouse click. This is because the method takes a *MouseEvent* as a parameter, and the *MouseEvent* class has, among other things, two int fields, *x* and *y*, that give the *x* (*i.e.* horizontal) and *y* (*i.e.* vertical) coordinates of the mouse click. The *MouseEvent* methods *getX()* and *getY()* can then be used to find these coordinates.
See: https://docs.oracle.com/javase/7/docs/api/java/awt/event/MouseEvent.html

You will note that the coordinates of the most recent mouse click are stored in the private instance variables of the *CirclesGUI* class, *X* and *Y*. These variables are then accessed by the inner class *CircleDrawPanel* to draw, and to redraw, the circle. Inner classes have unrestricted access to their containing classes instance variables, including private variables.

Please complete the following tasks:

1. Adjust the parameters of the circle, such that the exact centre of the circle is now where the user clicks the mouse. [3 marks]

2. Add a *JButton* to the *NORTH* of the *JFrame* using *BorderLayout* (2 marks). Set the text on the button. The text should invite the user to click+ the button in order to resize the circle (1 mark). [3 marks]

3. Write an inner class called *DiameterListener* to implement the *ActionListener* interface, and to listen to the *JButton* in the *NORTH* region.

   NOTE: you can only get credit for this part of the question by writing an inner class. [3 marks]

4. When the user clicks on the *JButton* in the *NORTH* region, the current circle should be redrawn with a random diameter. The size of the random diameter is limited by the width of the *drawPanel* variable. The circle's *X* and *Y* coordinates and colour should not change. Each time the user clicks the button the diameter changes again. [3 marks]

5 Add a *JButton* to the *SOUTH* of the *JFrame* using *BorderLayout* (2 marks). Set the text on the button. The text should invite the user to click the button in order to change the colour of the circle (1 mark). [3 marks]

6. Write a second inner class implementation of the *ActionListener* interface, called *RandomColourListener*. This class should listen to the *JButton* in the *SOUTH* region.
NOTE: you can only get credit for this part of the question by writing an inner class. [3 marks]

7. When the *JButton* in the *SOUTH* region is clicked, the colour of the current circle changes to a random colour. The size and position of the circle do not change. Each time the user clicks the button the circle's colour again changes randomly. [3 marks]

8. When the user clicks on the panel again to redraw the circle, the redrawn circle should have the same diameter and colour as the circle it is replacing. Subsequent circles should continue with the same diameter and colour until the user once again clicks on one or both of the buttons to change one or both of them. The user should be able to continue in this way until they decide to quit. [3 marks]

Reading for Part A

- See: http://docs.oracle.com/javase/7/docs/api/java/awt/Color.html for the constructors, for example *Color(int r, int g, int b*), that allow developers to make their own *Colors*.
- Chapter 11 of Volume 1 of the subject guide.
- Section 12.1 of Volume 1 the subject guide.
- Chapter 12 of *Head First Java*, pages 353-381 only.

Deliverable for Part A

- An electronic copy of your revised program: *CirclesGUI.java*

Reminder:
Please put your name and student number as a comment at the top of your file.

PART B

In this part of the coursework assignment you have been given some classes and a text file, that are intended to work together to allow a user to view, edit, add and delete items from a dictionary.

- The *DictionaryWord* class is a child class of the *Word* class. A *Word* has a String *word* field. A *DictionaryWord* has an inherited String word field and is extended to include a String *definition* and a String *usageExample*.

- A *Dictionary* object is a *List* of *DictionaryWords*, *i.e.* the class has one field, a *List* of *DictionaryWords*, and the constructor takes as a parameter a *List* of *DictionaryWords*. The *Dictionary* class has an instance method to find a *DictionaryWord* in the List. *DictionaryWords* are searched for by their *word* field. The class does not have methods to edit the list, since the object was conceived to be *read only* by the developer.

- A *DictionaryEditor* inherits from the *Dictionary* class. The Dictionary class has been extended in order to allow editing of a *Dictionary* object. Hence the *DictionaryEditor* class has methods to edit the *Dictionary*, and the *DictionaryWords* contained in it.

- The *DictionaryUI* class is a user interface, intended for viewing, editing, adding and deleting words from a *Dictionary*. When the class starts, the constructor makes a *List* of *DictionaryWords* read in from a text file. It then makes a *DictionaryEditor* object using this *List*.

- The *DictionaryUI* class also makes use of static methods in the *DictionaryFileManager* class; this class can be viewed as a utility class, since it contains only static methods that perform operations on Lists of *DictionaryWords*. These static methods are to read from a text file and fill a list with *DictionaryWords* read and parsed from the file, and to save the contents of the list to a text file, using the *toString()* method for *DictionaryWord*, and iterating through the list with an enhanced for loop.

List interface

You will note that in the files you have been given, ArrayLists are declared to be of type List, as are methods that return an ArrayList. List is an interface:

http://docs.oracle.com/javase/7/docs/api/java/util/List.html

Explanation below, modified from Effective Java, 2nd edition by Joshua Bloch

You should use interfaces rather than classes as parameter types. More generally, you should favour the use of interfaces rather than classes to refer to objects. If appropriate interface types exist, then parameters, return values, variables, and fields should all be declared using interface types.

Get in the habit of typing this:

```
// Good - uses interface as type
List<Subscriber> subscribers = new ArrayList<Subscriber>();
```

rather than this:

```
// Bad - uses class as type!
ArrayList<Subscriber> subscribers = new ArrayList<Subscriber>();
```

If you get into the habit of using interfaces as types, your program will be much more flexible. If you decide that you want to switch implementations, all you have to do is change the class name in the constructor.

For example, the first declaration could be changed to read:
List<Subscriber> subscribers = new LinkedList<Subscriber>();
and all of the surrounding code would continue to work. The surrounding code was unaware of the old implementation type, so it would be oblivious to the change.

Please complete the following tasks:

1. The *DictionaryEditor* class will not compile as it does not have an appropriate constructor[1]. Write a constructor for the class. [3 marks]

2. There are six methods referenced by the menu in the *DictionaryUI* class, and named by the *processMenuChoice(String)* method. Two of the methods have been completed (*lookup()* and *showAll()*), and four of them (in bold below) contain only an error statement to the user:

    1. lookup()
    2. **addWord()**
    3. **editWord()**
    4. **removeWord()**
    5. showAll()
    6. **quit()**

    Write the four methods. In your answer make sure that the methods perform the tasks as described by the comments in the *DictionaryUI* class. [9 marks]

3. You have *ThesaurusWord*, *ThesaurusFileManager* and *thesaurus.txt*. Write the classes:

    - *Thesaurus.java*
    - *ThesaurusEditor.java*
    - *ThesaurusUI.java*

    All five classes plus the text file should work together in the *ThesaurusUI* class, to allow the user to edit the thesaurus, in the same way that the *DictionaryUI* allows the user to edit a dictionary. The *ThesaurusUI* class should give the user the same six menu options as the *DictionaryUI* class. Note that the *ThesaurusWord* class has the fields *synonym*, and *antonym*, and these fields are Lists. This means that the method to edit the thesaurus must be able to overwrite these fields with new Lists. [9 marks]

4. Explain how and why using *Generic* classes would have simplified writing the dictionary and thesaurus classes?

    You should write no more than two paragraphs in answer to this question (note a paragraph is at most 8 sentences). You may hand in your answer as a PDF, Word, OpenOffice or text file.

    [3 marks]

[1] Or, more accurately, because the default constructor fails in its implicit call to the superclass constructor, due to the mismatch of parameters, *i.e*. the default constructor has none, and the superclass constructor has one (a *List* of *DictionaryWords*).

Reading for Part B
The following chapters and pages of Volume 1 of the subject guide, and associated *Head First Java* readings:

- Chapter 3
- Chapter 5, pages 39–41
- Chapter 7, pages 63-65
- Chapter 8, pages 73-75
- Section 10.3 and 10.4 (constructors, including the default constructor and superclass constructors).

Reading for question 4:

- https://docs.oracle.com/javase/tutorial/java/generics/

Deliverables for Part B
Please put your name and student number as a comment at the top of your Java files. Please only hand in the files asked for.

Please submit an electronic copy of the following:

- *DictionaryEditor.java* (with added constructor)
- *DictionaryUI.java* (with four completed methods)
- *Thesaurus.java*
- *ThesaurusEditor.java*
- *ThesaurusUI.java*
- A PDF, Word, OpenOffice or text file with your answer to question 4. Please name this file with your name and student number.

Reminder:
Please put your name and student number as a comment at the top of your Java files.
Please only hand in the files asked for.

MARKS FOR CO2220 COURSEWORK ASSIGNMENT 1

The marks for each section of Coursework assignment 1 are clearly displayed against each question and add up to 48. There are another two marks available for giving in uncompressed files and for giving in files that are not contained in a directory. This amounts to 50 marks altogether. There are another 50 marks available from Coursework assignment 2.

Total marks for Part A	[24 marks]
Total marks for Part B	[24 marks]
Mark for giving in uncompressed files	[1 mark]
Mark for giving in standalone files; namely, files **not** enclosed in a directory	[1 mark]
Total marks for Coursework assignment 1	**[50 marks]**

[END OF COURSEWORK ASSIGNMENT 1]