© UNIVERSITY OF LEEDS

School of Computing

May/June 2018

COMP1721

Object-Oriented Programming

Answer all three questions

Time allowed: 2 hours

Question 1

This question concerns the two Java classes whose source code appears in Figure 1.

(a) Why are the two methods defined on lines 12 and 14 necessary? What is the potential advantage of defining the `Location` class in this way? Compare and contrast this with how structs work in C. **[4 marks]**

(b) How would you create a `Location` object representing the University of Leeds in the United Kingdom? Write down a line of Java code that would achieve this. **[2 marks]**

(c) There is a compiler error in the `GPSLocation` class. Explain carefully the nature of this error. Your explanation should identify the exact lines of code at which the problem occurs and suggest how the code should be changed to fix the problem. **[4 marks]**

(d) After fixing the error referred to in (c), a programmer uses these classes in a program. At one point in the code, this program creates an instance of `GPSLocation` named `place`. At a later point, the following print statement appears:

```
System.out.println(place.getName());
```

With reference to the class implementations in Figure 1, explain why this does *not* cause a compiler error. **[2 marks]**

(e) The programmer in (d) above runs his program and observes that the print statement produces the following output:

```
University of Leeds
```

He then edits the print statement so that it looks like this:

```
System.out.println(place);
```

When he compiles and runs the program again, the output changes to this:

```
GPSLocation@17ac4923
```

(i) Explain why the programmer is seeing this output. **[2 marks]**

(ii) Describe what the programmer needs to add to the code in Figure 1 in order to see output that is more useful. **[3 marks]**

(f) Sketch the UML class diagram implied by the code in Figure 1. **[3 marks]**

[Question 1 total: 20 marks]

```
3  class Location {
4    private String name;
5    private String country;
6
7    public Location(String n, String c) {
8      name = n;
9      country = c;
10   }
11
12   public String getName() { return name; }
13
14   public String getCountry() { return country; }
15 }
16
17 class GPSLocation extends Location {
18   private double latitude;
19   private double longitude;
20   private double elevation;
21
22   public GPSLocation(String n, String c, double lat,
23    double lon, double elev) {
24     name = n;
25     country = c;
26     latitude = lat;
27     longitude = lon;
28     elevation = elev;
29   }
30
31   public double getLatitude() { return latitude; }
32
33   public double getLongitude() { return longitude; }
34
35   public double getElevation() { return elevation; }
36 }
```

Figure 1: Classes used in Question 1

Question 2

(a) A programmer has a class `Point`, representing the *x* and *y* coordinates of a point. She is writing a new class `Circle`, to represent a circle. Circles need to specify their origin as *x* and *y* coordinates, so she reuses the `Point` class by making `Circle` inherit from `Point`. Explain carefully why this is wrong. **[3 marks]**

(b) Show the correct approach for implementing the `Circle` class from (a). Present your answer in the form of a partial Java class definition. This should contain only the field definitions that the `Circle` class will need; it should contain no methods. Minor errors in syntax will not be penalised. **[3 marks]**

(c) The `Point` class from (a) above has a constructor that initialises fields named `x` and `y`, of type `double`, using values supplied as constructor parameters. Assuming that there is a requirement to prevent `Point` objects from being created with negative values for `x` or `y`, and that the goal is to make the `Point` class usable in as wide a range of situations as possible, how would you go about implementing the constructor?

You can present your answer as actual Java code (in which minor syntax errors will not be penalised) or as a clear step-by-step description of the required code. Whichever format you choose, make sure that you include an explanation of why your approach to preventing `x` and `y` from becoming negative is a good solution. **[7 marks]**
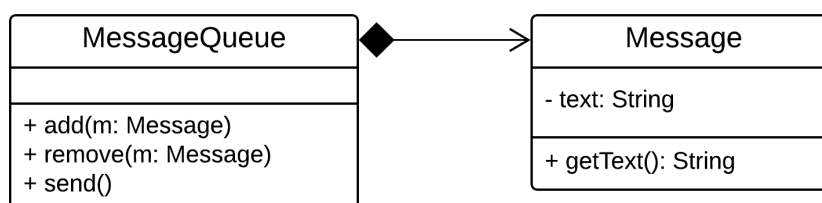
(d) The programmer from (a) is using her `Circle` class to represent the points in a scatter plot of some data. Every point in the plot will need to be represented by a `Circle` object. The programmer cannot decide what representation to use for the resulting collection of `Circle` objects.

Identify *three* different options that she has for implementing the collection of objects, assuming the use of Java as the programming language. Write some brief notes on the advantages and disadvantages of each option, to help the programmer decide which she should use. **[7 marks]**

[Question 2 total: 20 marks]

Question 3

(a) Discuss the relative merits of the procedural and object-oriented approaches to writing programs. Your discussion should focus on differences in approach, not on the features of specific languages such as C and Java. **[5 marks]**

(b) Discuss the potential advantages and disadvantages of porting an existing piece of software written in C to a new version written in Java. Your discussion should ignore Java's object-oriented nature and should instead focus on specific language features, as well as considering differences in how code written in these languages is developed, compiled and executed. **[5 marks]**

(c) The UML diagram below shows part of the design for an email server.

MessageQueue
+ add(m: Message) + remove(m: Message) + send()

Message
- text: String
+ getText(): String

(i) Write down the Java code implied by this diagram. Do not speculate about any features of these classes that are not implied by the diagram, and leave method bodies empty. You will not be penalised for minor syntax errors. **[6 marks]**

(ii) Discuss, without writing any actual Java code, what you would need to do to implement the `add`, `remove` and `send` methods. **[4 marks]**

[Question 3 total: 20 marks]

[Grand total: 60 marks]

END