University of Glasgow

Friday, 5 May 2017
2.00 pm – 4.00 pm
(Duration: 2 hours)

DEGREES of MSc in Information Technology, MSc in Software Development

Advanced Programming (M)

(Answer All Questions)

This examination paper is worth a total of 80 marks

The use of a calculator is not permitted in this examination

INSTRUCTIONS TO INVIGILATORS

Please collect all exam question papers and exam answer scripts and retain for school to collect. Candidates must not remove exam question papers.

1. **(a)** Describe two situations (with justification) in which multi-threaded programming would be beneficial.

[4]

(b) Describe one way in which multi-threading can be implemented in Java, outside of Swing.

[3]

(c) `Thread.join()` and `Thread.sleep()` are two examples of *blocking methods*. What is a blocking method, and what, programmatically, does they necessitate?

[4]

(d) Assuming that `Thready` is a class that extends `Thread`, the following code snippet has an error that stops it actually running concurrently. What is it?

```
public class ExamClass {
    public static void main(String[] args) {
        Thready[] t = new Thready[100];
        for(int i=0;i<100;i++) {
            t[i] = new Thready();
            t[i].run();
        }
    }
}
```

[2]

(e) You have been asked to write code to sort the values in a large array. Assuming you have written a method called `sort` that sorts values in an array but it's too slow. Sketch out the objects you would need to create (and how they would interact) to create a multi-threaded solution to this problem.

[5]

(f) When you implement your solution to part (e) and run it on a standard desktop machine you find that it doesn't improve speed. Why might this be?

[2]

(g) In multi-threaded code, describe what is meant by a *deadlock* and explain a general method for avoiding them.

[4]

(h) The following code snippet defines a class that extends SwingWorker. Assuming that countText is an output component that has a setText() method, explain the roles of publish and process.

```
private class CountTask extends SwingWorker<Void,Integer>{
    protected Void doInBackground() {
        try {
                Integer count = 0;
                while(!isCancelled()) {
                    count++;
                    Thread.sleep(100);
                    publish(count);
                }
            }catch(InterruptedException e) {}
        return null;
    }
    protected void process(List<Integer> counts) {
        int lastVal = counts.get(counts.size()-1);
        countText.setText(String.format("%d",lastVal));
    }
}
```

[4]

2. Imagine you have been tasked with creating a client-server system to allow students to give real-time feedback in lectures. The lecturer would run a server on their laptop that clients, running on students' devices could connect to. The system would allow two types of interaction: the server can send questions to students that they answer (with an integer) and allow students to send free text comments, that will flash up on the lecturer's screen, without prompting from the server. To make sure only students in the lecture can respond, the lecturer provides a password at each lecture at the beginning of the lecture that clients have to provide to connect.

(a) Design a communication protocol for the proposed system. The protocol should be sufficiently detailed that developers could build either the client or the server directly from it.

[6]

(b) Having decided on the protocol, you decide to build a prototype system that allows only a single client to connect to the system. Sketch the code for the:

i) Server

[5]

ii)Client

[5]

In each case, you can just embed your code in a main method. You don't need to implement error checking. Minor syntactical errors will not be punished.

3. The `decorator` pattern can be used to add or change functionality of a class without having to change the class definition. You have been hired by a burger bar to help computerise their order tracking system. They sell plain burgers that can be modified in different ways (e.g. add cheese, add bacon, sell as a meal etc.).

(a) Sketch out the objects required to implement a `decorator` pattern to store and manipulate burgers. Include two decorators of your choice.

[6]

(b) Write a main method that creates a burger and decorates it with both decorators. Minor syntactical errors will not be punished.

[2]

(c) State another design pattern that could be suitable for this problem and describe one advantage and one disadvantage of your pattern over the `decorator`.

[3]

(d) State one advantage of programming with design patterns.

[1]

(e) In the restaurant, kitchen staff are provided with screens on which to see how many of each type of burger are required. The restaurant owner also wants to watch (from her mansion) how burger sales are going. What is a suitable design pattern that will allow both user groups to see up-to-date information? Sketch the components required for your chosen pattern.

[4]

4. Please write the answers to the following 10 multiple-choice questions clearly in your answer booklet. Each correct answer is worth 2 marks. Incorrect answers will result in a penalty of two thirds of a mark to discourage guessing.

(a) Which of the following is **false** in Java:

A. A class can extend only one other class.
B. A class can implement only one interface.
C. A class can extend a class that itself extends another class.
D. A class can implement many interfaces.

[2]

(b) Which of the following statements about assertions is **false**:

A. Assertions are enabled at run-time.
B. When an assertion is false, it throws an exception.
C. Assertions have to be switched on when compiling.
D. Assertions are a useful debugging tool.

[2]

(c) Which of the following correctly describes the output of this code snippet:

```
public static void main(String[] args) {
    Double a = 3.5;
    Double b = a;
    b *= 2;
    System.out.println("a: " + a + ", b: " + b);
}
```

A. a: 7.0, b: 7.0
B. a: 3.5, b: 7.0
C. a: 3.5, b: 3.5
D. a: 7.0, b: 3.5

[2]

(d) Which of the following statements is **correct:**

A. A final method cannot be overridden.
B. A `final` class cannot be instantiated.
C. All Integer objects should be stored in `final` references.
D. Objects stored in `final` references cannot be modified.

[2]

(e) Consider a class `Parent` that is subclassed by a class `Child`. The expression: `Parent c = new Child();` is an example of:

A. Polymorphism
B. Abstraction
C. Overloading
D. Overriding

[2]

(f) Using the objects from the part (e), `Parent` has only one method: `getName()`. `Child` adds another: `isBorn()` that returns an `Boolean` value. What happens if you try and write the code `c.isBorn()` (where `c` is defined as in part (e))?

A. The code compiles and executes without error.
B. The code compiles but an error is produced on execution.
C. The code does not compile.
D. The code compiles and runs but nothing is returned from the method.

[2]

(g) When programming in Swing, which of the following statements about concurrency is **false**:

A. Native `Thread` and `Runnable` objects should not be used.
B. Concurrency can be achieved by creating objects that extend `SwingWorker`.
C. Computationally heavy jobs should not be placed on the event dispatch thread.
D. Objects extending `SwingWorker` should implement a `run()` method.

[2]

(h) In the following code, what is the purpose of the static decorator?

```
public class ExamClass {
      public static class SubClass {
          // Some classy things
      }
      public static void main(String[] args) {
          SubClass s = new SubClass();

}
```

A. An instance can be made without creating an instance of `ExamClass`.
B. All inner classes have to be declared `static`.
C. All classes without a main method have to be declared `static`.
D. It has no purpose and could be removed.

[2]

(i) Which of the following statements describes a sensible approach for overcoming *race conditions*.

A. Do not allow multiple threads to access shared objects.
B. `join` individual threads to make sure they don't do things at the same time.
C. Use the `synchronized` keyword to lock the problematic objects.
D. Create a `JUnit` tests with a high code coverage.

[2]

(j) Which of the following statements is **false**:

A. The composite design pattern is useful where objects have a natural grouping.
B. The leaf object in the composite design pattern inherits from the composite object.
C. The composite design pattern uses polymorphism to allow composites of composites.
D. The leaf and composite objects both inherit from the same base component object.

[2]