MARKING SCHEME

U08186 Advanced Object-Oriented Programming

Examination Rubric

Examination length: **2 hours.**

Answer **three** questions.

Section A is compulsory and contains one question, worth 40 marks.

Two further questions must be answered from Section B. Each is worth 30 marks.

The total number of marks is 100.

Examination Questions

Section A - Compulsory

Question A1

a) Draw the class diagram for the Strategy design pattern, explaining the meaning of the aggregation symbol, the inheritance arrow and the interface label applied to the class Strategy. Your explanations must be in the particular context of that design pattern.

Answer A1.a

4 marks for the diagram (bookwork) and 2 marks for each explanation, examples of which could be:

i) the Context class "has a" Strategy class ie a real word object that Context might represent is considered to be composed of a real world object that the Strategy class represents and possibly something else.

ii) the Concrete Strategy classes have at least the methods of the Strategy class and at least the attributes too (and sometimes more of course)

iii) the Strategy class, being an interface, consists of methods without implementations

10 marks

b) Write the code for an Iterator class for the class `ArrayList` and show how you would add an `iterator` method to the `ArrayList` class that creates and returns an iterator object of that class.

Answer A1.b

```
class ArrayList<E> {
    Iterator<E> iterator(){
        return new ArrayListIterator<E>(this);
    }
}

class ArrayListIterator<E> {
    private int index=0;
    private ArrayList<E> list;

    public ArrayListIterator(ArrayList<E> list){
       this.list = list;
```

```
        }

        private boolean hasNext(){
            return index!=list.size();
        }
        private E next(){
            return list.get(index++);
        }
        private void remove(){
            throw new UnsupportOperationException();
        }
    }
```

2 marks for each method definition, 1 mark for each attribute, so 12 marks in total but truncated to 10

10 marks

c) Which design pattern do you think is being described here in an extract from a design patterns textbook. Explain your answer.

> "Then she took me to the set of documentation. I am not making this up: there were 8 feet of manuals for me to read ... each page was $8\frac{1}{2} \times 11$ inches and in small print! This was one complex system! Now, if you and I and say another four of five people were on a project that needed to use this system, not all of us would have to learn the entire thing. Rather than waste everyone's time, we would probably draw straws, and the *loser* would have to write routines that the rest of us would use to interface with the system"

Answer A1.c

This is the Facade class (1 mark). The loser would write the Facade class, the loser's colleagues would write the client classes, and the classes behind the facade would be those of the complex system.

4 marks

d) What are the similarities and differences between the Adapter pattern and the Facade pattern in terms of what they do and the circumstances in which they are typically used?

Answer A1.d

Both allow client classes to access a class or classes using an interface that is more convenient. An Adapter pattern typically wraps a single class where

the Facade pattern map wrap several classes, leading to gains in simplicity, whereas the Adapter more typically is used to allow access to legacy classes through a different interface.

6 marks

e) The Command design pattern can be used to implement the response to menu item and button clicks on a GUI for Microsoft Word, for example. Show how this may be achieved by drawing a class diagram of the pattern instantiated to this particular situation. Explain the advantage of using the Command pattern in this situation.

Answer A1.e

6 marks for the diagram with, for example, OpenDocument as the ConcreteCommand, Application as the Receiver, and the comment for the action of OpenDocument being to call the open of the Application class. Deductions of up to two points for a class diagram more like the generic one. 4 marks for explaining the Command objects can be logged in a stack and then undone and redone really simply, by pushing or popping from the stack.

10 marks

[Total 40 marks]

Section B - Answer two questions

Question B1

a) Java 7 contains a syntax for binary literals. Illustrate this by writing code that assigns a 32-bit number of your choice to an integer variable `i`. The number should be broken up into 8-bit chunks for ease of reading.

Answer B1.a

```
int i = 0b10101010_01010101_10101010_01010101;
```

5 marks

b) Explain the syntax and semantics of two language features new to Java 7 that, if present, would tell you that a `try ... catch` block was written to be compiled by a Java 7 compiler rather than a Java 6 compiler.

Answer B1.b

The `catch` clause can take the form `catch C1|C2` indicating that a single catch clause can deal with two different classes of Exception at once. The `try` clause can take the form `try (???)` where `???` is a block that creates resources required just for the `try` block, to be reclaimed afterwards.

5 marks

c) Consider the following signature for a method designed to print all the elements of a list.

```
void printAll(List<?> list);
```

What are the advantages of making the parameter be `List<?>` rather than `ArrayList<?>`?

Answer B1.c

As `List` is the interface that `ArrayList` implements, the method can be called not only with `ArrayLists` but also `LinkedLists` and `Vectors`.

4 marks

d) One might expect the parameter to have the type `List<Object>`. That way, it seems likely, you could call it with parameters of type `List<String>` or `List<E>` for any class `E`. Explain carefully what is wrong with this argument and why the type `List<?>` is used instead.

Answer B1.d

Bookwork. This supposes that `List<String>` to be a subtype of `List<Object>` and the rule that methods can be called with parameters having types more specific than those in the parameter list. However, it is easy to construct an example (details for 3 marks) in which this would make it possible to call a method for `String`s on objects of type `Object`.

6 marks

e) In Java 7, it is possible to write `ArrayList<>`, for example, with no type specified for the elements. Give an example of a situation in which this is allowed and explain why it is allowed?

Answer B1.e

You can do this when creating a new object to assign to a variable so, for example, `new ArrayList<>();` is allowed if you can infer the type of the contained objects from the type of the ArrayList. For example, it is obvious in the case below that the `ArrayList` to be created must contain elements of type `String`.

```
ArrayList<String> strings = new ArrayList<>();
```

5 marks

f) Suppose that for some interface type `I`, the method signature was rewritten as follows:

```
void printAll(List<? extends I> list);
```

Explain, without simply reusing the word “extends”, what property this new signature now enforces upon the parameters with which the method `printAll` is called. Include in your answer a possible situation in which this restriction will be useful by suggesting a possible interface `I`.

Answer B1.f

The members of the list must all contain definitions for the methods listed in the interface `I`. A possible application for this would be where `I` is the interface containing a method to print the object in a certain way other than the default suggested by the `toString` implementation. In that case, lists would only be admitted if all of their members are printable.

5 marks

[Total 30 marks]

Question B2

a) Explain what the Spec# precondition:

```
requires forall{int i in (0: a.Length),
  int j in (i: a.Length); a[i] <= a[j]};
```

says about the array a.

Answer B2.a

The elements of the array must be in ascending order. Note ranges in Spec# are half open so that, for example, `i in (m: n)` means `m <= i< n`.

3 marks

b) An informal specification for a method states that it should accept an array of integers a (indexed from zero) and an integer `x` and return the position of `x` in the array. Explain why this specification is inadequate.

Answer B2.b

It does not say `x` can be assumed to be present or what to do if it is not found.

2 marks

c) Write a specification, including a contract in the form of pre- and post-conditions in the manner of Programming by Contract, which is fragile and which allows the programmer of the method to make an assumption. You may express your contract in mathematical notation or in the syntax of Spec# or in very clear English.

Answer B2.c

Pre: `There exists an i in range 0 to a.length-1 inclusive such that a[i] is equal to x.`
Post: `The value returned is the value of i`

4 marks

d) Write a different specification, also in the form of pre- and post-conditions, which is defensive and does not allow an assumption but returns a special value in a certain case. You may express your contract in mathematical notation or in the syntax of Spec# or in very clear English.

Answer B2.d

Pre: `true`
Post: `There exists an i in range 0 to a.length-1 inclusive such that a[i] is equal to x and returned value is i or There does not exist an i in range 0 to a.length-1 inclusive such that a[i] is equal to x and returned value is a special value (such as -1).`

3 marks

e) Explain how having a specification in the form of a contract simplifies the work of a programmer of a method.

Answer B2.e

Programmer knows what they are allowed to assume and what they must deliver. No need to guess at what required or to make up special values.

3 marks

f) Explain how having a specification in the form of a contract simplifies the work of a user of a method.

Answer B2.f

User knows what obligations there are on them and what they can expect in return.

3 marks

g) Write an `assert` statement in Java to test the precondition in part a) and print out a message if it is not satisfied. You may find it helpful to define the precondition itself in a separate boolean-valued method and call it in the `assert` statement itself.

Answer B2.g

```
private boolean ordered(int[] a){
    boolean orderedSoFar = true;
    for (int i=0; i <a.length - 1; i++) {
        orderedSoFar &= a[i] < a[i+1];
    }
    return orderedSoFar;
}
```

6 marks

h) Suppose a Java method called `find` implements the specification in c). Write a test using one of the methods of the JUnit framework to check that the postcondition of `find` is true in a situation where the precondition holds.

Answer B2.h

```
int[] a= new int[] {7,9,30,44};
int index= find (a, 30);
assertEqual(a[index], 30);
```

6 marks

[Total 30 marks]

Question B3

a) Consider the following program fragment:

```
int daysInMonth(int year, int month) {
switch(month) {
    case 2: if(leap(year)) days = 29; else days = 28;
    case 4: case 6: case 9: case11: days = 30;
    default: days = 31;
   }
   return days;
}
```

Assuming that the above fragment is written in Java, explain what would happen when a programmer tried to compile and run a Java program containing the method `daysInMonth` and the statement:

```
n = daysInMonth(2011, 4);
```

Answer B3.a

This is correct Java syntax, but `break` statements have been accidentally omitted. Switch statement branches to case 4 and sets days to 30, but then falls through to default case, setting days to 31 and returning erroneous value of 31.

4 marks

b) Assuming that the above fragment is written in C#, explain what would happen when a programmer tried to compile and run a C# program containing the method `daysInMonth` and the statement:

```
n = daysInMonth(2011, 4);
```

Answer B3.b

This is not correct C# syntax. Compiler will emit a message stating that `break` or `return` or `goto` or `throw` needed at end of each case (apart from default). Erroneous behaviour of Java version is avoided.

4 marks

c) The following fragment of C# includes the declarations of some properties:

```
public class Customer
{
     private int numberOfOrders = 0;

    public int ID
     {
         get
         {
             return numberOfOrders;
         }
         set
         {
             numberOfOrders = value;
         }
     }
}
```

Suppose that a variable customer of class `Customer` has been declared, instantiated and initialised. Write statement(s) in C#, using the properties, that will increase the customer's number of orders by 1.

Answer B3.c

```
customer.numbersOfOrders = customer.numbersOfOrders + 1;
```

or

```
customer.numbersOfOrders++;
```

4 marks

d) Write the statement(s) necessary to increase the customer's number of orders by 1 using "getter" and "setter" methods, named `getNumberOfOrders` and `setNumberofOrders`, as conventional in Java. Assume that these methods have been declared and defined.

Answer B3.d

```
customer.setNumberOfOrders(customer.getNumberOfOrders() + 1);
```

4 marks

e) The .NET framework allows parts written in a variety of programming languages to be composed in a single program. Describe a possible circumstance in which this might be desirable.

Answer B3.e

Scientific parts of a program are written in FORTRAN but user interface part is in C#. Do not need to rewrite the FORTRAN in C#.

2 marks

f) Explain the role of the Common Intermediate Language (CIL) in making this possible.

Answer B3.f

All programming languages for .NET are compiled into the one same intermediate language. This is then compiled just-in-time to the machine code of the actual machine to be used to run the program.

4 marks

g) Likewise, explain the role of the Common Type System (CTS) in making this possible.

Answer B3.g

All languages represent their data types in the same way, as defined by the CTS. Thus, for example, an `INTEGER` in FORTRAN has the same bit representation as an `int` in C#. Similarly for classes, structs etc.

4 marks

h) A programmer running programs using the .NET framework notices a delay when running programs for the first time but no delay on subsequent runs. Explain why this is so.

Answer B3.h

On first run, common intermediate language is complied, “just-in-time” into actual machine code. Delay noticed is because of this. On subsequent runs cached machine code used.

4 marks

[Total 30 marks]

End of Examination Paper

[All marks = 130]