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) A shop makes four different types of cake, each with a different price:

1. sponge cake, priced at 199p

2. sponge cake with icing, priced at 299p

3. sponge cake with icing and sprinkles, priced at 349p

4. sponge cake with sprinkles, priced at 249p

Consider the interface `Priceable` defined below:

```
interface Priceable {
    int getPrice();
}
```

You need to write a class `Cake` that implements `Priceable` and calculates the price of a cake.

i) Write the class using the Strategy Pattern. Note that this will involve writing a number of auxiliary classes. Some of them will be very similar to each other. You do not need to write them all out provided that you make the pattern clear. There is no need to define constructor methods for the classes.

Answer A1.a.i

```
class Cake implements Priceable {
    private PricingStrategy strat;

    public int getPrice() {
        return strat.getPrice();
    }
}

interface PricingStrategy {
    int getPrice();
}

class SpongeStrategy implements PricingStrategy {
    public int getPrice() {
        return 199;
    }
}

class SpongeIcingStrategy implements PricingStrategy {
    public int getPrice() {
        return 299;
    }
}

class SpongeIcingSprinklesStrategy implements PricingStrategy {
    public int getPrice() {
        return 349;
    }
}

class SpongeSprinklesStrategy implements PricingStrategy {
    public int getPrice() {
        return 249;
    }
}
```

4 marks for class `Cake`. 4 marks for the class hierarchy, with full marks awarded, as suggested, for an outline of a correct solution even with some classes (eg the last three) missing. In both cases, marks less than 4 are awarded according to the proportion of required features (classes, methods, attributes, return statements) supplied.

8 marks

ii) Write the class using the Decorator pattern. Again, there is no need to define constructor classes.

Answer A1.a.ii

```
class Cake implements Priceable {
    public int getPrice() {
        return 199;
    }
}

class CakeDecorator implements Priceable {
    private Priceable wrapped;
    public int getPrice() {
        return wrapped.getPrice();
    }
}

class SprinkleDecorator extends CakeDecorator {
    public int getPrice {
        return super.getPrice() + 50;
    }
}

class IcingDecorator extends CakeDecorator {
    public int getPrice {
        return super.getPrice() + 100;
    }
}
```

3 marks for each of the first three classes, up to a maximum of 8.

8 marks

iii) Draw the class diagram for the Decorator pattern, and explain in detail why your solution conforms to the Decorator pattern. You may do this using the Strategy pattern instead if you are more confident with your solution to the first part.

Answer A1.a.iii

2 marks for the class diagram for the Decorator pattern. 2 marks for stating that there must exist implementations of the `Priceable`, one of which must aggregate `Priceable` and have at least one subclass etc. 2 marks for annotating the classes with their corresponding classes.

6 marks

iv) explain why the solution using Decorator pattern is better than the solution that uses the Strategy pattern.

Answer A1.a.iv

The Strategy pattern requires 2^n subclasses where n is the number of features that may or may not be present. (2 marks) The possibility of, for example, a chocolate topping would double the number of subclasses. (2 marks) For the Decorator pattern, on the other hand, the number of subclasses is n, not counting the `Decorator` class. (2 marks)

6 marks

b) The State pattern makes it possible for an object to appear to change its class. Explain how this happens.

Answer A1.b

A Context class has a reference to a state whose type is a subclass of the State class. (2 marks) The Context object has a sends a message to the state which sends a message to the Context to tell it to change the object to which it has a reference. (3 marks) The new object belongs to a different subclass of the State class. (1 mark)

6 marks

c) The Singleton pattern ensures that only one instance of a class is created. Explain how it does this.

Answer A1.c

The constructor is made private so it can only be called from one place. (2 marks) That place is the static getInstance method, which being static, exists before any instances are created (2 marks) and checks that no instance has yet been created before creating one. (2 marks)

6 marks

[Total 40 marks]

Section B - Answer two questions

Question B1

a) Explain, with an example in each case, how the use of generics and enumerated types in Java programs reduces the possibility for run-time error.

Answer B1.a

Without generics, you can add to a collection an object with one type T1 and then after retrieval, as an Object, cast it to a different type T2, causing a ClassCastException. (2 marks) For example, T1= String and T2=StringBuffer when adding to an ArrayList. (2 marks) Without enumerated types, a value can be stored eg in an integer outside the range being used (2 marks) eg an integer outside the range 0..6 used to represent days of the week. (2 marks)

8 marks

b) Write a method that uses an enhanced for loop to concatenate together the elements of an ArrayList of Strings supplied as a parameter.

Answer B1.b

```
public static String concatenate(ArrayList<?> list) {
    String conc = "";
    for (String s: list) {
        conc += s;
    }
    return conc;
}
```

5 marks

c) Explain why the enhanced for loop is considered preferable to the for loop in this case.

Answer B1.c

It is not necessary to keep track of the index. It is not possible to be out of bounds with the index. The approach can be adopted to Collections that do not implement the List interface. (Any of three explanations allowed for two marks.)

2 marks

d) Explain the difference in meaning between the following five uses of the <> notations with reference, if necessary, to where in a program you may encounter them.

i) `ArrayList<Int>`

Answer B1.d.i

The type of ArrayLists with elements belonging to the type Int

3 marks

ii) `ArrayList<E>`

Answer B1.d.ii

The type of ArrayLists with elements belonging to the type E where E is a type variable that once instantiated, instantiates all other occurrences of E. This can be seen in the definition of a generic class ArrayList or in a generic class that extends or aggregates E.

3 marks

iii) `ArrayList<?>`

Answer B1.d.iii

Used as the type of a parameter, matches ArrayLists of any type.

3 marks

iv) `ArrayList<? extends I>`

Answer B1.d.iv

Used as the type of a parameter, matches ArrayLists of any type that inherits from class I.

3 marks

v) `ArrayList<>`

Answer B1.d.v

Used as the name of the constructor when the exact type of the ArrayList eg `ArrayList<Int>` is already known.

3 marks

[Total 30 marks]

Question B2

a) Explain what the precondition of this Spec# method means. Note use of “!”:

```
void High (int !a[], out int high) {
requires
    a.Length != 0;
ensures
    exists {int i in (0..a.Length): a[i] == high} &&
        forall{int i in (0: a.Length); a[i] <= high};
// code goes here
}
```

Answer B2.a

Array reference must not be null (“!”) and the array must have at least one element.

5 marks

b) Explain the meaning of the postcondition of the Spec# method in part a). Note that ranges in Spec# are “half-open”: `i in (m: n)` means that $\mathtt{m} \geq \mathtt{i} < \mathtt{n}$.

Answer B2.b

The value returned in high must be the highest in the array.

5 marks

c) A programmer implements the method from part a in this way.

```
high = -int.MaxValue;
int j = 0;
while (j != a.Length) {
    if (a[j] > high) high = a[j];
    j = j + 1;
}
```

Comment on this and reason about whether it satisfies the specification.

Answer B2.c

Presets high to -int.MaxValue which is not (necessarily) a member of the array but OK because of precondition. It does satisfy the specification.

5 marks

d) Would the above implementation satisfy the specification if the line

```
requires
    a.Length != 0;
```

had been omitted? Explain.

Answer B2.d

No requirement for array not to be empty. Attempts to set high even if no elements in array. In that case sets high to a value that is not and cannot be an element of the array.

5 marks

e) Explain how having a specification in the form of a contract simplifies the work for the 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 is required or to make up special values.

5 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.

5 marks

[Total 30 marks]

Question B3

a) The following fragment of C# includes the declarations of “properties”.

```
public class C {
    private static int size;
    public static int Size {
        get { return size; }
        set { size = value; }
    }
}
```

Write statement(s) in C# using the properties that will set size to 3.

Answer B3.a

```
C.size = 3;
```

4 marks

b) Write statement(s) in C# using the properties that will increase size by 1.

Answer B3.b

```
C.size++;
```

4 marks

c) Write the statement(s) necessary for part b above using conventional “getter” and “setter” methods called `getSize` and `setSize`. Assume that these methods have been declared.

Answer B3.c

```
C.setSize(C.getSize() + 1);
```

4 marks

d) A programmer wishes to use a large scientific program written in FORTRAN but via a graphics user interface which is to be written in C#. Explain the role of the Common Intermediate Language (CIL) of the .NET framework in making this possible.

Answer B3.d

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.

5 marks

e) Explain the role of the Common Type System (CTS) in making this possible.

Answer B3.e

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

f) 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.

Answer B3.f

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

g) A C# compiler will report an error when trying to compile the following. Explain why that is, why reporting an error is a good idea and how the program fragment can be corrected.

```
int DaysInMonth(int year, int month) {
    switch(month) {
        case 2: if(leap(year)) days = 29; else days = 28;
        case 4: case 6: case 9: case 11: days = 30;
        default: days = 31;
    }
    return days;
}
```

Answer B3.g

This is not correct C# syntax; `break` (or `return`, or `throw`, or `exit`) statements have been omitted. Defining this as a syntax error saves the situation of ”falling though” to next case limb, giving erroneous answer. Corrected by adding `break` at end of each case limb.

5 marks

[Total 30 marks]

End of Examination Paper

[All marks = 130]