BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 5 Diploma in IT

Object Oriented Programming

Examiners Report March 2018

Section A

A1. (a) Explain what is meant by the following terms:

i) Class
ii) Object instance
iii) Abstract class
iv) Method overloading
v) Method overriding

(15 marks)

(b) Explain how the Liskov substitution principle is used in object oriented systems. Your answer should include an example to illustrate how this is used.

(10 marks)

Answer Pointers:

(a)

i) Class - A class is a template for an object. It defines all the data items contained in the object and the methods that operate on that data

ii) Object instance - An object is a collection of data and operations. The data describes the state of the object. The operations are the methods defined in the class and those inherited from superclasses

iii) Abstract class – an abstract class is a class that has been declared as abstract. The crucial difference from a class is that it cannot be instantiated, though it can have subclasses.

iv) Method overloading - – a subclass can override a method it has inherited from it's superclass. It will have the same name as the method it is overloading, but can change the signature in that it can have different types and number of parameters, or a different sequence.

v) Method overriding – a subclass can override a method it has inherited from it's superclass. It will have the same name, number and types of parameters and return type as the method it is overriding.

(b)

The substitution principle means that an object of a subclass can be used anywhere an instance of a superclass is expected without alternating the properties of the program calling it. For example, if the Library example in question 3 had both Books and Journals with a Publication superclass, objects of type Book or Journal could be substituted for a Publication object, since they would be subclasses of Publication.

Because a subclass inherits all the attributes and methods of its superclass, this means the substitution is possible, even if the methods had not been overridden or overloaded. For example, the catalogue() method in Book could have overwritten a method with the same name in Publication, plus it could have inherited other methods from Publication that it can also use.

Examiners Comments:

Question (a) examines syllabus section 2 (Concepts), part 2.1
Question (b) examines syllabus section 2 (Concepts), part 2.3

A popular question answered by over 89% of the candidates, with 54% passing. Most candidates could answer part (a) well. Marks were generally lost where candidates mixed up method overloading and method overriding, or described abstract data types instead of abstract classes.

Not many candidates attempted part b, but those who did generally gave good answers.

A2. (a) Give an outline of what you would expect to find in the description of a design pattern for reusable object oriented software. State the reasons why a programmer would want to use a design pattern when developing a system.

(10 marks)

(b) Describe in detail **THREE** design patterns with which you are familiar. Your answer should include the circumstances in which they are applicable and an example of their use (note, actual code is not necessary).

(15 marks)

Answer Pointers:

(a) The following should be covered as part of the documentation of a design pattern:

Motivation – scenario consisting of a problem and a context in which this pattern can be used

Prerequisites – what needs to be in place beforehand

Structure – graphical representation of the pattern. Class diagrams and Interaction diagrams may be used for this purpose

Participants – listing of the classes and objects used in the pattern and their roles in the design

Consequences – description of the results, side effects, and trade-offs caused by using the pattern.

A programmer should use this documentation to evaluate how the pattern will help develop their system.

(b) The candidates can choose any design pattern, for example:

Adapter (Structural)

The Adapter is intended to provide a way for a client to use an object whose interface is different from the one expected by the client, without having to modify either. This pattern is suitable for solving issues such as replacing one class with another when the interfaces do not match and creating a class that can interact with other classes without knowing their interfaces at design time.

Decorator (Structural)

Ordinarily, an object inherits behaviour from its subclasses. The decorator pattern allows the behaviour of an object to be extended and dynamically compose an object's behaviour. Decorator allows this without the need to create new subclasses.

Iterator (Behavioural)

The Iterator pattern provides a way to access the elements of an aggregate object sequentially without having to know the underlying representation. Iterator also provides a way to define special Iterator classes that perform unique processing and return only specific elements of the data collection. The Iterator is useful because it provides a common interface so programmer does not need to know anything about the underlying data structure.

Observer (Behavioural)

The Observer pattern is useful when data is to be presented in several different forms at once. The Observer is intended to provide a means to define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. The object containing the data is separated from the objects that display the data and the display objects observe changes in that data. Often used in MVC.

Singleton (Creational)

The singleton pattern applies to the many situations in which there needs to be a *single instance* of a class, a single object. Print spoolers and window managers are examples of Singletons. The Singleton is intended to provide a way to ensure that a class provides one instance of itself, and to provide a global point of access.

For each category, the candidate should give a short description of one of the diagrams, include a simple example and say when they should be used.

Examiners Comments:

This question examines syllabus section 3.4 (Design), part 3.4

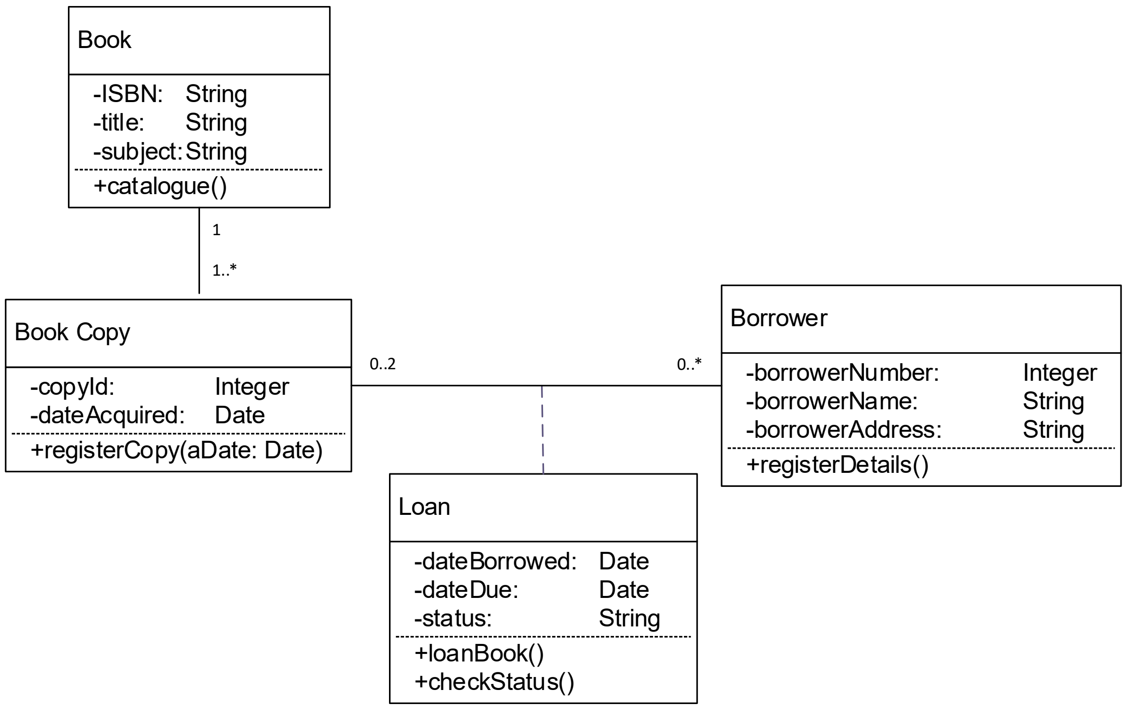
This question was answered by 75% of the candidates and 63% passed. Most candidates could discuss why a programmer would want to use a design pattern in part a, though were weaker at outlining what is expected in the description of a design pattern, sometimes overlooking this completely. To gain full marks both parts of the question needed to be answered.

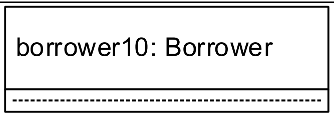
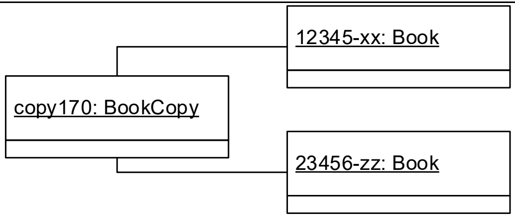
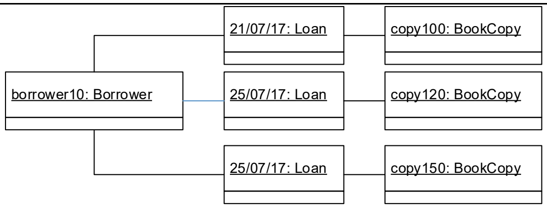
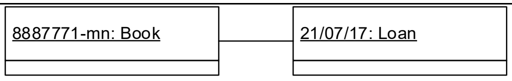
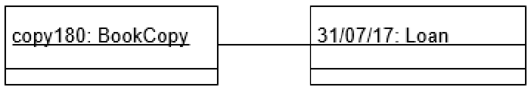
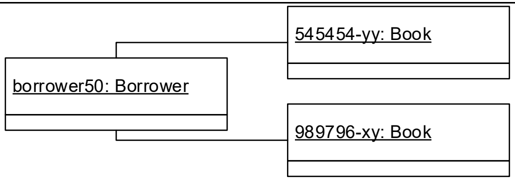
For part b, the most popular design patterns were the adaptor, singleton and observer. Most candidates could give an outline of what the pattern was, but the answer required an example of the circumstances of where they were applicable and an example of their use for full marks. Program code was not necessary, but some candidates ignored this advice and wrote a lot of code with no explanation, which gained no marks.

Some candidates lost marks by describing the three main categories of design patterns, rather than actual design patterns, or described concepts found in a design pattern, such as inheritance, but not the relevant design pattern. Occasionally some answers used the wrong diagram, or notation, for example, they described a UML class diagram without discussing the context with respect to design patterns.

A3 (a) Given the class diagram below, which represents part of a Library system, state which of the object diagrams (i-vi) are legitimate instances. Assume that all links in the object diagram are instances of the association shown in the class diagram. If an object diagram is not legitimate explain why not.

(10 marks)



i)		iv)	
ii)	 21/07/17: Loan copy100: BookCopy borrower10: Borrower 25/07/17: Loan copy120: BookCopy 25/07/17: Loan copy150: BookCopy	v)	
iii)		vi)	

(b) List the techniques that could be used to test the above library system and explain the advantages and disadvantages of each technique. Include some discussion of when the techniques can be used during system development, assuming an object-oriented approach has been used throughout.

(15 marks)

Answer Pointers:

(a)

i. Ok, borrower to loan is optional
ii. Incorrect: a borrower can only loan 2 books at most
iii. Ok, a book copy can have one loan
iv. Incorrect, a copy can only be for one book
v. Incorrect, a loan is for a copy of a book
vi. Incorrect, the direct relationship is between borrower and the book copy

(b)

An open-ended question, where the Candidate can look at different approaches, for example:

- Fault based testing
- Scenario based testing
- How different UML diagrams can be utilised

Or may discuss black-box and white-box testing with respect to object oriented programming.

Some discussion of when testing should be done for full marks, it can be iterative and incremental, not just left to the end.

Examiners Comments:

Question (a) examines syllabus section 3 (Design), part 3.2

Question (b) examines syllabus section 4 (Practice), part 4.4

This question was answered by 53% of the candidates, with 68% passing.

For part a, marks were generally lost by not giving the reason for their answer. A number of candidates argued that iii was incorrect because an instance of Borrower should also be present, which was accepted.

Part b was generally answered well, with a number of candidates achieving full marks. Marks were lost by not explaining both the advantages and disadvantages of the technique chosen, or did not consider where in the system development the technique could be used.

Section B

B4. The fidelity of an audio clip is defined by a number of parameters, including the number of channels (1 or 2), the resolution (8, 16 or 24 bits per sample), and the sampling rate (22050, 44100, or 88200 samples per second).

a) In an object oriented programming language of your choice, write a definition for an `audioClip` class that contains:

(i) fields for storing the `channels`, `resolution` and `sampleRate` with appropriate visibility;

(ii) setter and getter methods for manipulating these fields, such that the setters methods ensure that `channels` can only have values 1 or 2, `resolution` can only have values 8, 16 or 32, and `sampleRate` can only have values 22050, 44100 or 88200;

(iii) a constructor that initialises new objects to have the lowest quality, where `channels` is set to 1, `resolution` is set to 8, and `sampleRate` is set to 22050.

(15 marks)

b) Write a new method called `isStudioQuality` that will return true or false, depending upon whether the audio clip stored has the maximum possible quality (i.e., two channels, 24-bit resolution, and a sample rate of 88200 samples per second).

(5 marks)

c) Write a new method called `dataSize` that accepts the duration that an audio clip lasts in seconds (as an integer), and returns the number of bytes that this audio clip would occupy on disk or in memory.

The formula for calculating number of bytes, b, where is d duration (in seconds), c is channels, r is resolution (in bits), and s is sample rate, is:

$$b = d \times c \times \left(\frac{r}{8}\right) \times s$$

(5 marks)

Answer Pointers:

a)

```
class audioClip
{
  private:
    int channels;
    int resolution;
    int sampleRate;
  public:
    audioClip() {channels   = 1;
                 resolution = 8;
                 sampleRate = 22050;}
    int getChannels() { return channels; }
    bool setChannels(int newChannels)
    {
      if(newChannels==1 || newChannels==2)
      {
        channels = newChannels;
        return true;
      }
      return false;
    }
    int getResolution() {return resolution;}
    bool setResolution(int newResolution)
    {
      if(newResolution==8  ||
         newResolution==16 ||
         newResolution==24)
      {
        resolution = newResolution;
        return true;
      }
      return false;
    }
    int getSampleRate() { return sampleRate; }
    bool setSampleRate(int newSampleRate)
    {
      if(newSampleRate==22050 ||
         newSampleRate==44100 ||
         newSampleRate==88200)
      {
        resolution = newSampleRate;
        return true;
      }
      return false;
    }
};
```

b)

```
public:
  bool isStudioQuality()
  {
    if(channels==2 && resolution==24 && sampleRate==88200)
      return true;
    return false;
  }
```

c)

```
public:
  int dataSize(int duration)
  {
    return duration*channels*sampleRate*(resolution/8);
  }
```

Examiner Comments:

This question covers Syllabus section 2 (Concepts) parts 2.1 and 2.3.

This question covers Syllabus section 4 (Practice) part 4.3.

This question was attempted by 57% of candidates, with 79% of those achieving a pass mark. In part (a), a number of candidates incorrectly set the visibility of the data members of the class to public, and did not verify that the argument sent to the setter functions was valid before reflecting this change to the data members. Another error made by some candidates was to store the data members as strings, but then not to reflect this choice in the other methods (i.e., by enclosing literals for comparison in double quotes). An part (b), some candidates incorrectly accepted arguments and verified that those arguments represented a studio quality recording, rather than referring to the data members of the class in which this function sits. In part (c), similarly to part (b), some candidates accepted arguments for channels, sample rate, and resolution, rather than referring to the private data members. Furthermore, some students duplicated variables unnecessarily or didn't use parenthesis correctly.

B5

a) When would private and protected class members be used in an object oriented program? Clearly distinguish between them.

(5 marks)

b) How are private and protected class members represented in a UML class diagram? Give an example class definition in UML that contains a field of each type.

(5 marks)

c) Name and describe one other class member visibility modifier, and state how it would be represented in a UML class diagram.

(5 marks)

d) When would composition and aggregation inter-class relationships be used in an object oriented program? Briefly describe the difference them.

(5 marks)

e) Show how composition and aggregation inter-class relationships would be represented in a UML class diagram.

(5 marks)

Answer Pointers:

a)

Class members defined as private are only visible inside the class in which they are defined. Class members defined as protected are visible inside the class in which they are defined, and in classes derived from that class.

b)

Class members defined as private are prefixed with a minus symbol (-) in a UML class diagram, whereas class members defined as protected are prefixed with a hash symbol (#).

```
 _______________________________
|           myClass             |
|_______________________________|
|                               |
| -myPrivateField: Integer      |
| #myProtectedField: Integer    |
|                               |
|_______________________________|
|                               |
|_______________________________|
```

c)

Another class visibility setting is public. Public members can be access inside the class, in derived classes, and directly via other the classes; e.g., through objects of that class without using methods. Fields and methods designated as public are prefixed in a UML class diagram with a + symbol.

d)

Aggregation implies that a relationship exists between two classes, but each class can exist independently (i.e., they are separable, like a book and a library). It is a “part of” or “has a” relationship. Composition is a stronger form of aggregation, wherein there is an ownership relation such that if an owner object is destroyed, the owned objects are also destroyed (i.e., they are non-separable and cannot exist/function independently, like the nodes in a linked list and the linked list itself). It is sometimes called an “owns a” relationship. These containment principles are used to interconnect classes to implement a particular object oriented design.

e)

In a UML class diagram a composition relationship is represented as a filled diamond, and aggregation relationship as an empty diamond. The candidate should draw examples of these, each connecting two classes.

Examiner Comments:

This question covers Syllabus section 2 (Concepts) 2.1-2.4.

This question covers Syllabus section 3 (Design), parts 3.1 and 3.2.

This question was attempted by 100% of candidates, with 73% achieving a pass mark. The vast majority of candidates did reasonably well in answering parts (a) to (c), with most correctly identifying public as the visibility operator not previously mentioned. In part (d), a number of candidates confused aggregation and composition, and others mixed up the symbols used to represent these relationships in part (d).

B6

The Fibonacci numbers are a sequence of integers starting 1, 1, generated such that every subsequent number is the sum of the previous two. For example, the third number in the Fibonacci sequence is 2 (because 1+1=2), and the fourth number is 3 (because 2+1=3).

Fibonacci numbers are used in several algorithms in Computer Science, including the Fibonacci Search and in the generation of fractals.

a) In an object oriented programming language of your choice, create a class called **`fibonacciClass`** capable of holding Fibonacci numbers in an array called **`F`**.

   Include a data member called **`currentNumbersHeld`** that will record how many of the numbers are currently being held. Include a constant called **`maxNumbers`**, set to `100`, that stipulates the maximum number of Fibonacci numbers that can be held. At this stage, do not add any methods.

(5 marks)

b) Add a getter method that returns **`currentNumbersHeld`**.

(5 marks)

c) Add a method called **`generateSequence`** that will populate the array with the Fibonacci numbers accepting one integer argument, **`N`**, that specifies how many numbers to generate. Check that **`N`** is ≤ **`maxNumbers`**. If **`N`** exceeds **`maxNumbers`**, generate the maximum number of Fibonacci numbers.

(10 marks)

d) Implement a method called **`displaySequence`** that will display the Fibonacci sequence currently stored in the array in the format

```
F[1] = 1
F[2] = 1
F[3] = 2
```

   and so on.

(5 marks)

Answer Pointers:

a)

```
#include <iostream>
using namespace std;
class fibonacciClass
{
  private:
    const static int maxNumbers = 100;
    int F[maxNumbers];
    int currentNumbersHeld;
};
```

b)

```
  public:
    int getCurrentNumbersHeld() {return currentNumbersHeld;}
```

c)

```
 public:
   void generateSequence(int N)
   {
     if(N<1)
       return;

     if(N==1)
     {
       F[0] = 1;
       currentNumbersHeld = 1;
       return;
     }
     else if(N==2)
     {
       F[0] = 1;
       F[1] = 1;
       currentNumbersHeld = 2;
       return;
     }

     if(N>maxNumbers)
       N = maxNumbers;

     currentNumbersHeld = N;
     F[0] = 1;
     F[1] = 1;
     for(int i = 2;i<N;i++)
       F[i] = F[i-1]+F[i-2];

   }
```

d)

```
public:
  void displaySequence()
  {
    for(int i=0;i<currentNumbersHeld;i++)
      cout << "F[" << i+1 << "] = " << F[i] << endl;
  }
```

Examiner Comments:

This question covers Syllabus section 4 (Practice) parts 4.2 and 4.3.

This question was attempted by 23% of candidates, with 47% of those achieving a pass mark. In part (a), a fairly large number of candidates did not declare maxNumbers as a constant. Many candidates were unable to develop the generateSequence function needed for part (b), with many failing to realise that iteration is required to traverse the elements of an array.