BCS HIGHER EDUCATION QUALIFICATIONS
Level 5 Diploma in IT

October 2010

EXAMINERS' REPORT

Object Oriented Programming

General Comments

Candidates should take care to ensure that they write their answers in the correct answer book. In addition, they should record the questions they have attempted on the front of the answer book.

In the answers to this paper there was clear evidence that some of the candidates revised by studying past papers. This approach is to be commended as the answer pointers given in previous examiners' reports provide concise examples of the key concepts examined in this paper.

Section A

Question A1

a) Compare and contrast:
   i) Object oriented programming;
   ii) Procedural programming;
   iii) Structured programming.

(15 marks)

b) You have been invited to give a talk to trainee programmers outlining the reasons for the widespread use of object oriented programming within the software development industry. Summarise the points you would present in your talk.

(10 marks)

Answer Pointers

a)

i) Object oriented programming is a programming style in which the solution to a problem is represented as a set of interrelated objects. Objects in this sense are collections of related data which represent the state of the object and methods which manipulate the data.
ii) In procedural programming the programmer concentrates on the steps that must be undertaken to solve the problem rather than the data and the operations which apply to that data.
iii) Structured programming is a type of procedural programming. In structured programming a programmer uses a limited set of control structures in the construction of the program. Examples include while, if and for. The advantage of a structured approach over plain procedural programming is that it supports a top-down approach to design.

b) The following is indicative of the points that might be made in response to the question. A large number of alternatives would be equally acceptable.
There are three main ways in which object-oriented programming has been found to improve programmer productivity:

i) By providing an environment which assists programmers to improve the quality of their code;
ii) Making it easier for programmers to reuse their own code;
iii) Providing simple mechanisms to make use of existing code libraries.

Object-oriented programming embodies practices which have been known for some time to lead to well constructed programs. It associates procedures with the data those procedures use. It can be used to form a clear separation between underlying data structures and functionality. The concept of object is a good abstraction mechanism that helps a designer separate out a small part of a problem and concentrate on that simpler part consequently increasing the probability of that part of the design being correct. Object-oriented programming languages encourage and support object thinking.
In general the fewer lines of code a programmer has to write the more productive they will be. This can be facilitated by providing powerful features in a language (such as a matrix multiplication operator). The disadvantage of such features is that often in a bespoke environment they will not do exactly what a programmer needs. Consequently programmers will write their own code to overcome this. A classic example is a date checking routine. Often such routines are textually copied from one program to the next. This creates a maintenance nightmare. Object-oriented programming facilitates and encourages the use of re-usable classes which allow programmers to reuse the same code over and over again without physically copying it.
Just as programmers may reuse their own classes, they may also easily reuse classes provided by third parties. This is particularly useful where programmers have to produce complex interfaces to their programs. The inheritance mechanism allows programmers to enhance third party classes to meet their individual requirements without the need to alter the original code and therefore tailored software can be developed. The majority of code in any application will often be incorporated in this way and therefore productivity is greatly enhanced.
The disadvantages of object-oriented programming include the learning curve necessary to become proficient in it and the fact that code produced by an object-oriented language compiler is unlikely to be as efficient as code produced by the best machine code programmers.

Examiners Comments
This question examines Syllabus 8A: Foundations
Approximately three quarters of all the candidates answered this question. The majority of the answers were satisfactory or better. In general, candidates performed better in part a) of the question which required them to describe three programming paradigms. In the second part of the question some candidates simply wrote everything they knew about object oriented programming which did not attract as much credit as the answers which addressed the scenario set out in the question.

Question A2

a) Explain the following terms:
   i) Method signature;
   ii) Method overloading;
   iii) Operator overloading;
   iv) Method overriding.

(12 marks)

b) Give the meaning of the term polymorphism and explain why polymorphism is an important feature of object oriented programming.

(6 marks)

c) Give an example of code which demonstrates polymorphic behaviour.

(7 marks)

Answer Pointers

a)

i) The signature of a method is a prototype of a method, enabling the programmer to easily identify the purpose of a method (via its name), the data it requires to operate (via its argument list), and any end result that is produced (its return type), such as the outcome of a calculation or simply a status value that specifies whether the method was successful or unsuccessful.
ii) Method overloading is the creation of several functions with the same name which differ from each other in terms of the type of the input and the type of the output of the function. In overloading the name of the function is the same but some other part of the signature is different.
iii) In operator overloading, operators like +, =, or == can have different implementations depending on the types of their arguments. Sometimes the overloadings are defined by the language; sometimes the programmer can implement support for new types.
iv) Method overriding allows a subclass to provide its own implementation of a method already provided by one of its superclasses. A subclass can give its own definition of methods which also happen to have the same signature as the method in its superclass.

b)

Polymorphism is the idea of allowing the same code to be used with different types, resulting in more general and abstract implementations.
In strongly typed languages, polymorphism usually means that type A somehow derives from type B, or type A implements an interface that represents type B.
The primary usage of polymorphism is the creation of objects belonging to different types to respond to method, field, or property calls of the same name, each one according to an appropriate type-specific behaviour. The programmer (and the program) does not have to know the exact type of the object in advance, and so the exact behaviour is determined at run time

c)

```
public class Feline {
    void makeNoise() {
        System.out.println("Non-specific cat sound");
    }
}

public class DomesticCat extends Feline {
    void makeNoise() {
        System.out.println("Meow Meow");
    }
}

public class Lion extends Feline {
    void makeNoise() {
        System.out.println("Roar! Roar!");
    }
}

public class Test {
    public static void main(String[] args) {
        Feline f1 = new DomesticCat();
        Feline f2 = new Lion();
        f1.makeNoise();
        f2.makeNoise();
    }
}
```

Here f1 and f2 both reference Felines but the makeNoise method gives different outputs when applied to the different objects

Examiners Comments
This question examines Syllabus 8B: Concepts
Over 80% of the candidates attempted this question. Over 60% of the answers were satisfactory or better. The majority of candidates made good attempts at Parts a) and b) of the question. In effect these two parts were looking for definitions. Part c) required the candidates to demonstrate practical programming skills but there were very few good answers to this part. Many candidates chose not to supply an answer to part c).

Question A3

a) Explain the following terms:
   i) Class;
   ii) Object;
   iii) Method;
   iv) Instance variable;
   v) Class variable;

(10 marks)

b) Give an example of code which makes use of both an instance variable and a class variable.

(7 marks)

c) Discuss the use of inheritance in the development of object oriented programs.

(8 marks)

Answer Pointers

a)

i) 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) 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) A method is an operation which will manipulate the data contained in an object.
iv) An instance variable stores data which describes an individual object. If a class defines an instance variable then there will be space allocated for that variable every time a new object is created.
v) A class variable holds information which may be accessed by every object belonging to the class. There will only be one copy of a class variable in existence regardless of the number of objects created for that class

b)

```
public class ClassInstanceExample {

    private static int objectCount=0;
    private int objectId;

    public ClassInstanceExample(){
        objectCount++;
        objectId = objectCount;
    }

    public void printId(){
        System.out.println(objectId);
    }

    public static void main(String[] args) {
        ClassInstanceExample o1 = new ClassInstanceExample();
        ClassInstanceExample o2 = new ClassInstanceExample();
        o1.printId();
        o2.printId();
    }

}
```

objectCount is a class variable
objectId is an instance variable

c)

Inheritance is a mechanism used to create new classes which are based on existing classes. The class created is called a subclass whilst the class on which the new class is based is called a superclass. In inheritance, the subclass introduces new data above and beyond that held in the superclass and/or new methods in addition to those implemented in the superclass. The major use of inheritance is to permit incremental development. In this type of development existing classes are reused and new functionality is provided by building classes which inherit from them.

Examiners Comments
This question examines Syllabus 8B: Concepts
This question was attempted by almost 90% of the candidates. Only just over half of the candidates who attempted the question provided an answer which was satisfactory or better. In part a) sub-parts i), ii) and iii) were generally well answered. Candidates were less certain as to the definition of class and instance variables. As a consequence, there were very few correct answers to part b). Very few of the candidates who were able to explain the meaning of the terms ‘class variable’ and ‘instance variable’ knew how to declare these variables in the programming language of their choice. Answers to part c) were better, however, candidates lost marks by simply stating what inheritance is and failing to explain its role in the development of programs.

Section B

Question B4

a) Draw a Use Case diagram for the Wrekin Athletics Running Club described in the Appendix.

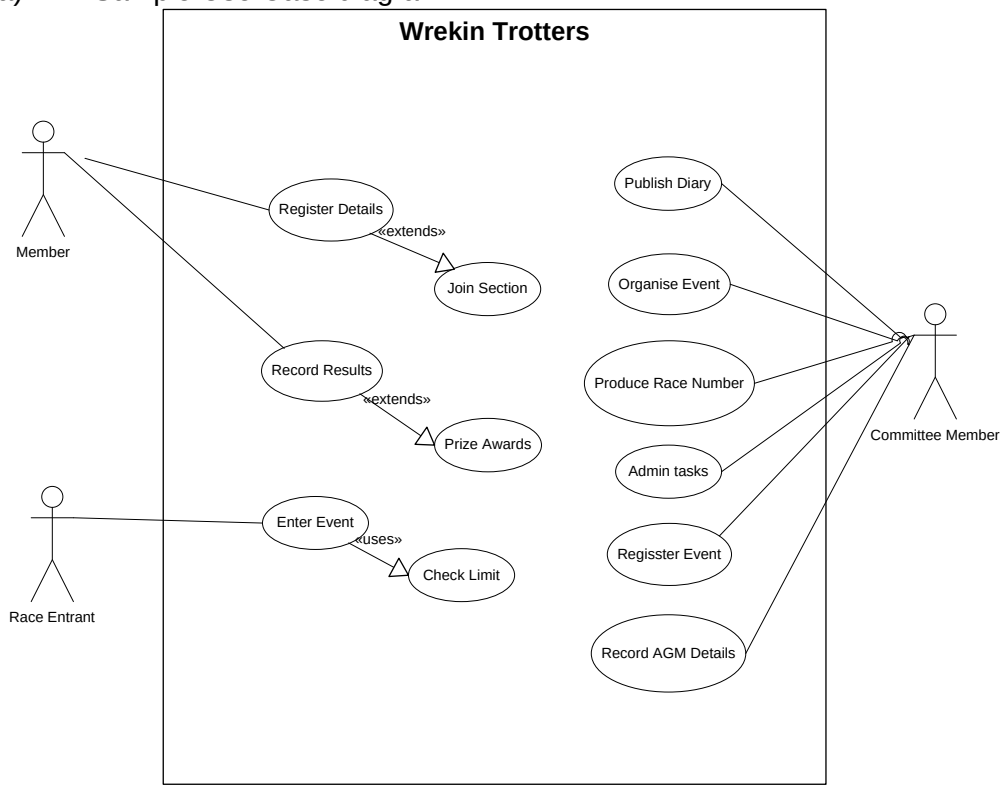
(15 marks)

b) Discuss how Use Case diagrams and scenarios contribute to the development of a system. Within your answer include an example scenario from the Wrekin Athletics Running Club.

(10 marks)

Answer Pointers

a) Sample Use Case diagram:



b)

The candidate should discuss where Use Cases should be used in developing a system. For each Use Case on the Use Case Diagram the user should develop a set of scenarios. Scenarios are natural language descriptions of an instantiation of the Use Case. These can be used in the following areas:

- Initial investigation
- For identifying what functionality is required from the system
- The descriptions give the fuller detail
- Testing purposes

The candidate should include a brief example of a scenario to illustrate their points, e.g., Enter Event.

Examiners Comments

This question examines Syllabus 8C Design

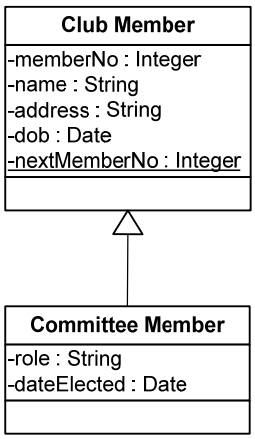
Over 65% of the candidates attempted this question, with over 75% passing. Part a) involved drawing a use case diagram for the Wrekin Trotters system. Most candidates answered this part well and obtained good marks, since they could identify the main actors and use cases. A small minority could not distinguish the use cases, instead including all the actions, which led to a much cluttered diagram. Lower marks were given to candidates who missed some of the use cases and actors; did not connect them to the correct actors or connected the Extends/Uses relationships inappropriately.

For this question, most marks were lost in part b), where candidates failed to discuss where use cases could be used in the development process. Some candidates only provided use case descriptions, but did not then discuss how they were useful to the development of the system.

There were also a few instances where the wrong type of diagram was provided in part a), such as an Entity Relationship diagram or Class diagram, which were not appropriate and so lost marks.

Question B5

The following is a Class diagram that represents part of the Wrekin Athletics Running Club described in the Appendix:



a) Using an object oriented programming language with which you are familiar with:

i) Write code to show the declaration of the Club Member and Committee Member classes, you do **not** need to include *setter* and *getter* methods.

(9 marks)

ii) Write code to declare a constructor for each class, the Club Member constructor should take 3 parameters: `name`, `address` and `date of birth` (dob). Note that the `nextMemberNo` contains the next number that should be used for a new member and should be incremented appropriately. Committee Member should take the same 3 parameters, plus 2 additional ones for `role` and `date elected`.

Within your code show how both constructors could be invoked.

(10 marks)

b) Discuss how the code in part (a) can be tested.

(6 marks)

Answer Pointers

a) Any object-oriented programming language can be used, below is the total code for part a), using Java.

```
import java.util.Date;

class ClubMember {
      int memberNo;
      String name;
      String address;
      Date dob;
      static int nextMemberNo = 1;

public ClubMember (String aName, String anAddress, Date aDOB) {
      name = aName;
      address = anAddress;
      dob = aDOB;
      memberNo = nextMemberNo;
      nextMemberNo++;
      }
} // ClubMember

class CommitteeMember extends ClubMember {
      String role;
      Date dateElected;

public CommitteeMember (String aName, String anAddress, Date aDOB,
  String aRole, Date aDE) {
      super (aName, anAddress, aDOB);
      role = aRole;
      dateElected = aDE;
      }
} // CommitteeMember

      ClubMember aClubMember = new ClubMember("Barnaby", "Wolves",
            new Date());
      CommitteeMember aCommitteeMember =
            new CommitteeMember("Felix", "Telford", new Date(),
            "Chair", new Date());
      }
}
```

b) This is an open-ended question. The Candidate may look at different approaches, for example:

- Fault based testing
- Scenario based testing

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

Examiners Comments

This question examines Syllabus 8D: Practice

Over 58% of the candidates attempted this question, with just over 74% passing. Part a) proved to be a question that some candidates answered extremely well. The majority of answers could define the basic class definitions, marks were lost for not showing how the inheritance was implemented; did not define `nextMemberNo` as a class variable, nor used it to assign the next member number. Marks were also lost if examples of instantiation were missing.

Most candidates made a good attempt at part b), however, some discussed how to fix compilation errors, rather than how to test a system.

Question B6

a) Explain what is meant by the term *design pattern* in the context of object oriented programming. Within your discussion, highlight what is the motivation for using a design pattern from a programmer's point of view.

(10 marks)

b) Design patterns can be classified according to the problem they solve. For example:

Creational Patterns:

Abstract factory, Builder, Object Pool and Singleton patterns

Structural Patterns:

Adapter, Decorator, Façade and Proxy patterns

Behavioural Patterns:

Command, Iterator, Observer and State patterns

Pick ONE design pattern from EACH of the above classifications and give a detailed description of it, stating the problem it addresses and the basis of the solution it offers.

(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, which can be any of the following:

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 underling 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.
Abstract factory (Creational)
Provides an interface for creating families of related or dependent objects without specifying their concrete classes
Object Pool (Creational)
Manages the reuse of objects for a type of object that is expensive to create or only a limited number of a kind of object can be created.
For each category, the student 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 8C & 8D: Design and Practice
This question proved to be less popular, with just over 25% of candidates attempting it and just under half of these passing. Part a) was often not attempted and in part b), a lot of candidates could only discuss a design pattern from one or two of the categories and some even discussed more than one pattern from the same category, with the observer and iterator patterns being the most popular. It is suspected that some candidates knew a few design patterns well, but not necessarily one from each of the given categories.
Marks were lost too, if basic descriptions were given, with no attempt made to discuss what problems they provided a solution for.

Appendix

Wrekin Athletics Running Club

Wrekin Trotters is a running club for members of all abilities, from beginners to athletes who represent their country. If a new member wants to join the club they can register their details online, by providing their name, address, telephone number, and emergency contact name and telephone number.

During the week, the club runs several activities, for example, speed work and training runs. These activities are advertised in a Club Diary, which is produced and put online by a Committee Member.

During the year, the club runs a number of external events that are open to both members and non-members of the club, for example, the Wrekin Half Marathon. Runners must register for each event separately, even if they are members of the club. Entrants need to provide their name, address, date of birth, fee payable and club name. One week prior to the event, a committee member will produce a race number for the event, which is posted to the entrant. Each event has a maximum number of participants and once the limit is reached, no further entries are allowed.

For each event, several administration tasks have to be carried out by a Committee Member, such as order refreshments and medals. A record needs to be kept of the date these were ordered and the quantity. For insurance reasons, a member of the committee must also register the event with the Amateur Athletics Association (AAA).

The Club has a League for its members, the results from the club events, and other external events, are recorded online by each member and at the end of the year prizes are awarded for various categories, such as the Best Overall Senior Runner, or Best Newcomer Runner.