THE BCS PROFESSIONAL EXAMINATIONS
Diploma in IT

October 2006

EXAMINERS' REPORT

Object Oriented Programming

Question 1

1. *a)* Describe the features that differentiate object-oriented programming languages from structured programming languages which do not support objects. **(10 marks)**

Answer Pointers

a) Answers were expected to include (but were not limited to) the following features:

- Object – the combination of data and the operations that apply to that data.
- Class – the pattern for all objects which have identical data and behaviour.
- Inheritance – the facility to derive new classes from existing classes.
- Methods – the way in which behaviour is defined.
- Messages – the mechanism for invoking behaviour.

Two marks for the name of each concept and a short description. 10 marks

b) You have been asked to advise the manager of an IT department on the choice of programming language. The manager wishes to know whether the use of an object-oriented programming language would increase programmer productivity. Write a report that sets out the potential benefits and disadvantages of deploying an object-oriented language. **(15 marks)**

Answer Pointers

b) There are three main ways in which object-oriented programming might 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 almost certainly in a bespoke environment they will not do exactly what a programmer needs. Consequently programmers will write their own code to do 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. 15 marks

Examiner's Comments
(This question examines part 1 of the syllabus: Foundations)

Most candidates who attempted this question were able to answer part a) successfully with the answers given matching those above. Some candidates were unable to distinguish those features of a programming language that relate to object-oriented languages from those which are not. For example, a number of candidates cited sequence, selection and iteration as being present in object-oriented languages but not in structured programming languages. (Candidates should note that non-OO languages are likely to be examined in other papers.)

Part b) of the question was not answered as well as part a). Candidates tended to repeat the list of features they had provided as the answer to part a) without additional comment. Marks were awarded for this part if the candidate demonstrated why a particular feature was likely to aid programmer productivity and not just for saying what it was.

Question 2

2. *a)* What is the difference between an *object* in an object-oriented language and a *variable* in a structured programming language? **(4 marks)**

Answer Pointers

a) A variable is simply a name given to an area of memory which the program will use. In a typed language the compiler is able to constrain what type of data is place into this memory location. An object also identifies an area of storage for data, additionally however, it also determines what operations may be applied to that data. 4 marks

b) Using an example of code written in a language with which you are familiar, explain what is meant by *object creation* and *initialisation*. Your answer should explain the role of *constructors* in *object initialisation*.

(9 marks)

Answer Pointers

b) Suppose we have the following class in Java:

```
class MyClass{

    protected int count;

    MyClass(){
        count=0;
    }

    MyClass(int x){
        count=x;
    }
}
```

An object can be created by either of the following two statements:

```
MyClass o1 = new MyClass();
MyClass o2 = new MyClass(1);
```

During the creation of object o1 the first constructor is invoked to set the value of count to 0. In the creation of object o2 the second constructor is invoked to set count to 1.

9 marks

c) Expand your answer to part *b)* to show the order in which constructors are invoked when the object to be initialised belongs to a class X which is the subclass of class Y where class Y also defines constructors.

(6 marks)

Answer Pointers

c) If following on from b) we have the following subclass:

```
public class MySubclass extends MyClass {

   /** Creates a new instance of MySubclass */
   public MySubclass() {
      count=count+1;
   }

   public MySubclass(int x){
      count=count+x;
   }
}
```

Then an instance of MySubclass can be created by:

```
MySubclass o3 = new MySubclass();
MySubclass o4 = new MySubclass(3);
```

In this case the constructor of MyClass would be invoked to set count to zero. Consequently count in o3 would be set to 1 and in o4 to 3. 6 marks

d) Some object-oriented languages allow the programmer to write methods known as *destructors*. Explain the term *destructor*, the purpose of these methods, and at what point they are invoked. **(6 marks)**

Answer Pointers

d) When a complex object (an object where the constituent parts are also objects) is created more than one object may come into existence. Typically the main object will contain object references to the subsidiary objects. When the object is no longer required it is desirable that the memory it occupies is returned to the available memory pool. Destructor functions define what happens to the subsidiary objects when the main object's memory is released. They are invoked just before the object's memory is deallocated. Destructors are generally only found in languages which do not support automatic garbage collection. 6 marks

Examiner's Comments
(This question examines part 2 of the syllabus: Concepts)

Part a) of the question required the candidates to explain why the concept of object is not simply another word for a variable. The majority of candidates answered this successfully. Part b) of the question asked candidates to write code. This year, a number of answers were written using VB.NET. The quality of these answers was equivalent to those written in Java and C++. This part of the question sought to ascertain whether the candidates understood how objects were created and initialised. The majority of answers were correct. Part c) which linked object creation and initialisation to inheritance was less well answered. Only a few answers showed an understanding of the way constructors are invoked when an object belonging to a subclass is created. Part d) was very poorly answered and in the few instances where candidates knew what a destructor is; only a small number could explain its purpose.

Question 3

3. *a)* Explain the following terms:

i) Inheritance
ii) Abstract class
iii) Delegation

Your answer should explain how a programmer might make use of each of these concepts and illustrate their use with an example of code. **(15 marks)**

Answer Pointers

a) i) Inheritance is a way to form new classes using classes that have already been defined. The former, known as derived classes, take over (or inherit) attributes and behaviour of the latter, which are referred to as base classes. It is intended to help reuse of existing code with little or no modification.

```
class MyClass{

    protected int x=0;

    public void increment(){
        x++:
    }

}

class MySubclass extends MyClass{
```

```
}
```

Objects of MySubclass implement increment and have access to x.

ii) A class that contains one or more *abstract methods* (methods without a concrete implementation), and therefore can never be instantiated. Abstract classes are defined so that other classes can extend them and make them concrete by implementing the abstract methods

```
class MyClass{

    public abstract void sort ();{

}

class MySubclass1 extends MyClass{

    public void sort(){
        .
        .
    }
}

class MySubclass2 extends MyClass{

    public void sort(){
        .
        .
    }
}
```

Since MySubclass1 and MySubclass2 inherit from MyClass they must implement sort(). The semantics of sort() can, however, vary between the two subclasses.

iii) Delegation occurs when a complex object implements some of its functionality by referring a message to one of its constituent objects. Programmers use this where a class which does part of what they require is available.

```
class MyClass{

    private Delegate d = new Delegate();

    public method(){
        d.method();
    }
}
```

Class contains an object to which it delegates a operation. 15 marks

b) Discuss the contribution of *inheritance*, *abstract classes* and *delegation* to promoting software reuse. Illustrate your answer by explaining how these features can be used to implement collection classes.

(10 marks)

Answer Pointers

b) Inheritance and delegation can be used to create new classes from classes already in existence. Use inheritance when you wish to introduce an object which is a special type of an existing object and delegation where the new object is not a type of the existing object but needs to make use of its functionality. Abstract classes can be used when a number of objects should present the same interface but vary the way in which they carry out their tasks.

In the Java collections framework, for example, a TreeSet is a special type of Set which is defined by its implementation. HashSet is another type of Set which has a different implementation. Therefore both TreeSet and HashSet inherit from Set so that both have the same interface. Set contains a number of abstract methods (in Java inherited from AbstractCollection) so that TreeSet and HashSet may specify their own implementations. Set will also contain some concrete methods where the code is independent of the implementation.

A Stack in Java inherits from Vector. This means that Stack supports many methods that are not normally associated with a stack in traditional computer science. To produce a stack which behaves in the way expected, embed a Stack object in a class which only supports push, pop and empty methods. 10 marks

Examiner's Comments

(This question examines part 2 of the syllabus: Concepts)

Part a) of the question was fairly well answered. The majority of candidates understood inheritance; many could describe an abstract class, whilst slightly fewer understood delegation. A few candidates who could explain in general terms what an abstract class was; were unable to describe where it might be used.

Part b) required the application of the theoretical knowledge of part a) to a practical problem. To offer a good answer, candidates needed to understand the programming language concepts and also know something about collection classes. Only a few candidates achieved this. There appeared to be little knowledge of the basics of collection classes and a lack of understanding as to when the techniques available for constructing classes with new functionality using existing classes might be employed.

Question 4

4. *StartRite Kiddies* is a nursery school for young children under 4 years of age. The company wishes to keep information on the children and the staff who work there. Two types of staff are employed: Administrators and Child Minders. Personal details, such as name and address, are recorded by the Administrator for all staff and if they are a Child Minder they must also pass a police check. The system must record when this has been passed and when it must be renewed, because a Child Minder cannot work with the children until this condition is met.

When a child starts at the nursery, an Administrator records the child's personal details such as name, address, date of birth and at least one emergency contact number, up to a maximum of three. As part of the registration process, the Administrator also records the details of who is allowed to collect the child, such as their name, address and photo. The Child Minder will print these details, so that they can check that the person collecting the child at the end of a session is authorised to do so.

Some children have special needs requiring medication, such as asthma, and the Administrator will record what the condition is, what medicine can be used and what to do in an emergency.

There are three different classes, after an initial assessment the Child Minder is responsible for allocating each child to a class appropriate to his/her age and ability.

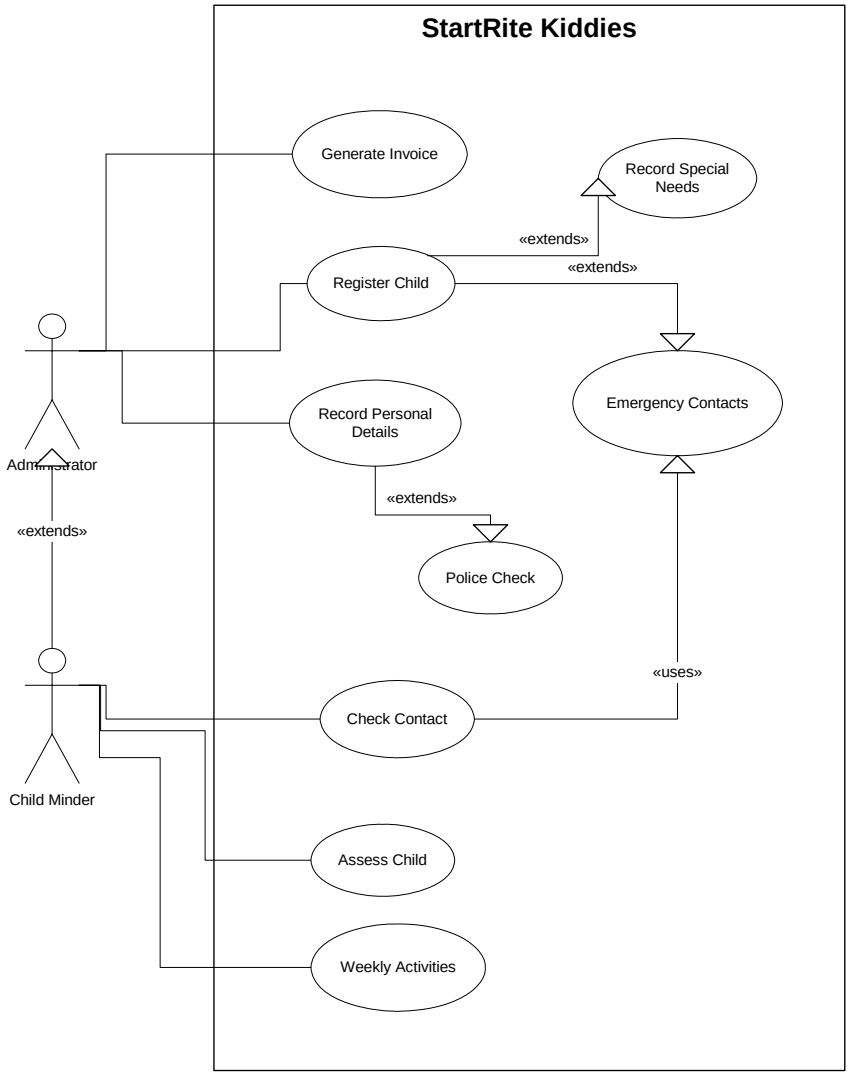
Each week the children take part in a number of activities and for these they may need to bring in additional items from home, for example, boots if they are going to the park. The Child Minder will generate a weekly letter for the parents to advise them of what they need to bring in.

At the end of each month an Administrator will generate an invoice for the parents to pay.

a) Draw a Use Case diagram for this system. **(15 marks)**

Answer Pointers

a)



15

1 mark per use case

1 for correct actors

1 mark per extends

1 mark uses

b) Discuss the role of Use Cases (diagrams and descriptions) in the development of an object-oriented system. **(10 marks)**

Answer Pointers

b) Student 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 two main areas:

- Initial investigation

For identifying what functionality is required from the system
The descriptions give the fuller detail

- Testing purposes

The Use Case descriptions can be used by the testers to develop test plans, checking that the system meets the requirements of the system.

Use Cases also act as the focus for other UML diagrams and to aid the documentation of the system. 10 marks

Examiner's Comments
(This question examines part 3 of the syllabus Design)

The candidates made a good attempt at this question on the whole and was a popular question. In part A, in most cases the actors and the main use cases were correctly identified. One of the main errors was in mixing up the registration of new members of staff and the children. Other marks were lost if the actors were not named, or an actor was linked to the wrong use case. Often candidates did not see the scope to extend some of the use cases.

Some candidates misread part b and provided use case descriptions for the system, which was not required, unless they were given as examples to the show the purpose of use case diagrams.

Question 5

5. *a)* Describe what is meant by the process of *iterative and incremental development*. **(7 marks)**

Answer Pointers

a) Iterative development allows a software system to be developed incrementally, allowing the developer to learn from the earlier, incremental, deliverable versions of the system

Iterative and incremental development basically means the system is developed in little steps, with the developer compiling and running the program after making each small step.

Advantage in that the developers do not have to write the program in one go and can introduce the system gradually to the users. 7 marks

b) Using examples from an object-oriented programming language with which you are familiar, discuss the suitability of object technology for iterative programming development. **(10 marks)**

Answer Pointers

b) By using classes, object-oriented technology naturally splits an application into manageable units. These are the classes that go together to make the final product. Object technology results in software with clean interfaces that can be tested in isolation. In addition once an interface has been constructed it is not necessary to produce all the code for a class at once.

The candidate should include some examples from a programming language they are familiar with (e.g., an outline class definition). 10 marks

c) When developing any system it needs to be tested. Describe TWO different approaches that could be used when using object-oriented technology. **(8 marks)**

Answer Pointers

c) 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.

4 marks per approach

Examiner's Comments
(This question examines part 4 of the syllabus: Practice)

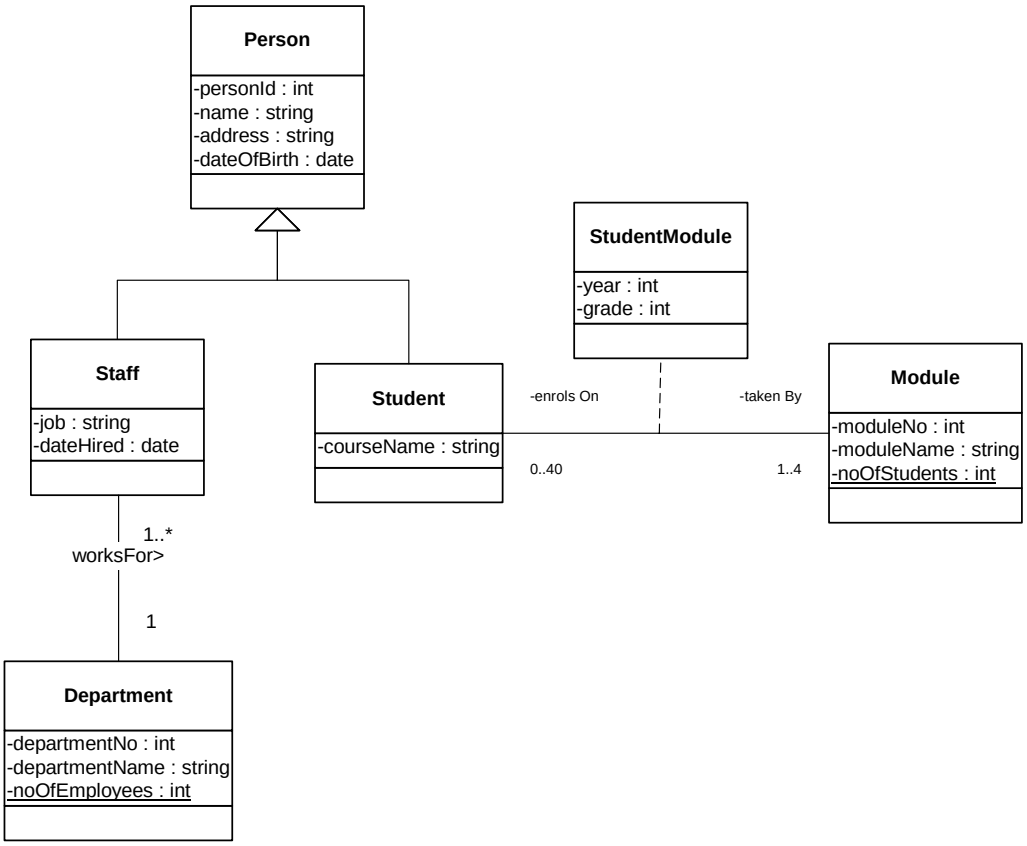
This question was less popular with the candidates. In part a), some answers confused iteration with iterative programming and gave examples of code showing iteration. Most candidates were able to say what incremental development is.

Candidates lost most marks in part b) by not providing examples of code. Exact syntax was not required; outline class definitions would have been acceptable.

On the whole most candidates could answer part c) adequately. A lot of candidates described white and black box testing, better marks were given however to those who tailored their answer to object-oriented programming and did not just say what white and black box testing was.

Question 6

6. The class diagram below represents a student system that records what course they are doing and the modules they take.



a) Describe what the diagram above represents. Include all structural constraints. **(15 marks)**

Answer Pointers

a)	The diagram has the following classes:	15 marks
	- Person	
	Class with the instance variables personID, name, address and DOB	1 per class desc
	Person has two subclasses:	
	- Staff	2 for
	- Student	inheritance

Staff has the instance variables job and dateHired. dateHired has a default value of the current date when the object is instantiated.

Student has the additional instance variable courseName. 1 per class var

There are two other classes:

- Module
- Department

Module has the variable moduleNo, moduleName and noOf Students. noOF students is a class variable, the others are instance variables.

Department has the variables departmentNo, departmentName and noOf Employees. noOf Employees is a class variable, the others are instance variables. 2 per relationship

2 for assoc. class desc

There are a number of associations in the classes:

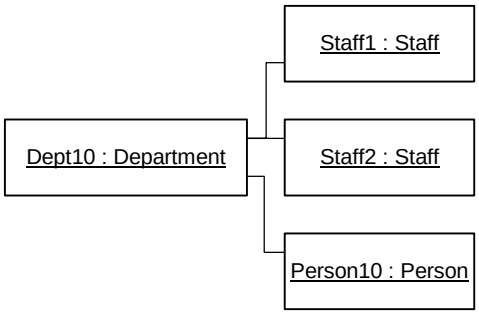
- WorksFor between Staff and Department. Each member of staff can work for 1 department and 1 department only. A department must have at least 1 member of staff working in it and can be many.
- Between Student and Module, this has an Association class called StudentModule, which has the instance variables year and grade.

A student can enrol on between 1-4 modules, whilst a module may be taken by no students, up to a maximum of 40 students.

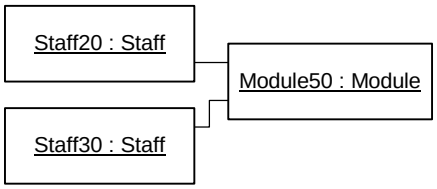
b) Given the object diagrams below *(i-vi)*, state which are legitimate instances. If an object diagram is not legitimate explain why not. **(10 marks)**

Answer Pointers

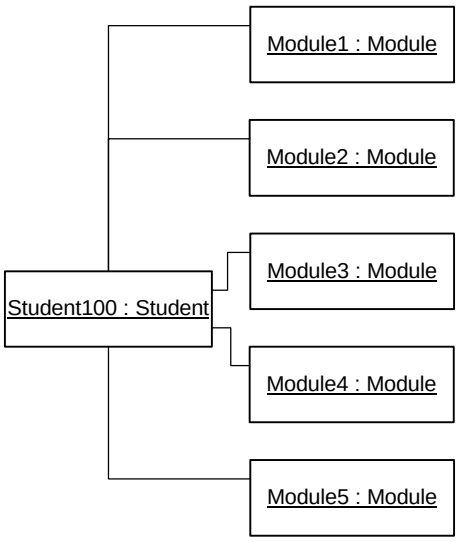
i)



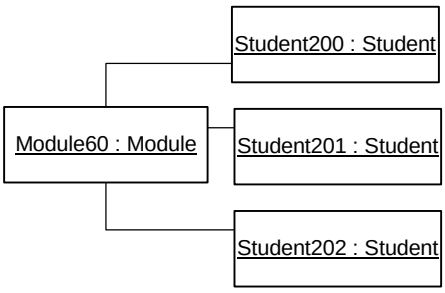
ii)



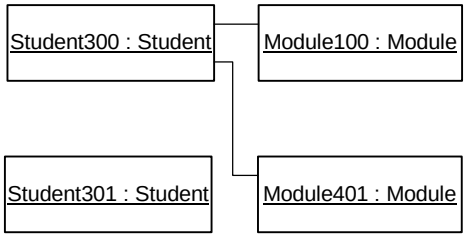
iii)



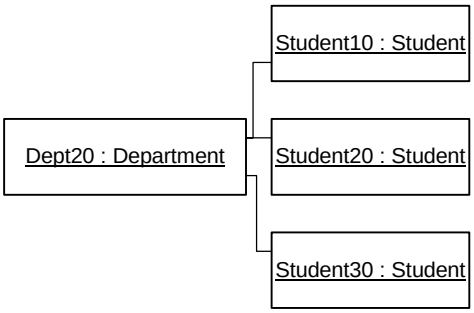
iv)



v)



vi)



Answer Pointers

b) Only iv) is correct. 10 marks

- i) Person does not work for department (also accepted that Person can not be instantiated if candidate identified Person as an abstract class in part a)
- ii) No link between staff and module
- iii) No of module instances exceeds what a student can take
- iv) Ok
- v) Student must be registered on at least one module
- vi) No link between student and department

2 per reason for incorrect ones

Examiner's Guidance Notes

(This question examines part 3 of the syllabus: Design)

This proved to be a popular question and most candidates made a good attempt. Marks were lost by not always providing enough detail, for example, just stating the names of the classes, without mentioning what properties they had, or omitting the cardinality constraints on the relationships.

Part b) was also answered well, with most candidates identifying what was wrong with the object diagrams. The one that caught a few candidates out was number v), where a student must be registered on at least one module.