THE BCS PROFESSIONAL EXAMINATION

Professional Graduate Diploma

September 2014

EXAMINERS' REPORT

Object Oriented Programming

Question A1

(a) Early programming languages did not support an object oriented approach to programming. What are the limitations of theses languages when compared to object oriented programming languages?

(10 marks)

Answer pointers

Early programming languages separated data and process. They concentrated on the things that had to be done and the order they had to be done in. Structured programming represented an improvement on the very early languages in that structured programming languages clearly identify the basic building blocks of an algorithm and allow a programmer to connect these together. It was recognised, however, that the data that is manipulated is also an important aspect of programming and that programs are largely the combination of data structure and algorithms. Programs may be constructed from abstract data types where the data structure is defined by the operations that can be performed on it. OO languages implement the idea of encapsulation where data is only accessed via a subroutine or a function call. Object oriented programming languages were created to support this type of programming whilst at the same time permitting incremental development of new functionality via inheritance.

(b) Describe FIVE features you would expect to be present in an object oriented programming language. Give an example of how each feature is realised in an object oriented programming language with which you are familiar.

(15 marks)

Answer pointers

i) Class

```
Class MyClass{
.
.
}
```

ii) Object

```
MyClass myObject = new MyClass();
```

iii) Data member

```
Class MyClass{
.
int x;
.
}
```

iv) Method

```
Class MyClass{
.
  void myMethod(){
     .
  }
.
}
```

v) Inheritance

```
Class MyClass{
.
.
}


Class MySubclass extends MyClass{
.
.
}
```

Examiners Comments

This question examines part 1 of the syllabus: Fundamentals

85% of the candidates chose to answer this question. Answers to part a) were disappointing given that this question frequently appears in the OOP paper. The question seeks to probe whether candidates understand why object oriented programming has come into existence by looking at the advantages that this paradigm has against older programming languages. Only a few candidates were able to clearly set out the advantages gained by associating operations with the data structures they operate on.

In part b) the answer given in this report sets out five possible responses, however, other appropriate features were also given credit. Most candidates were able to set out five characteristics of an object oriented languages. Many candidates lost marks because they were not able to illustrate a feature with corresponding code.

Question A2

(a) Explain the meaning of the term memory management and discuss why it is a necessary part of computer programming.

(6 marks)

Answer pointers

All programs require computer memory in order to store values for variables etc. Memory management is the task of allocating an appropriate amount of memory in an appropriate location to hold the data a program operates on. Since the amount of memory is finite, any allocated memory should be deallocated when it is no longer required.

(b) Write code which demonstrates the way in which memory is allocated and initialized in an object oriented programming language with which you are familiar.

(10 marks)

Answer pointers

In Java memory is allocated at object creation time via the new operator.

```
MyClass myObject = new MyClass();
```

Initialisation is normally achieved through the use of a constructor.

```
        Class MyClass{
                int myVar;
                MyClass (int initValue){
                        myVar = initValue;
                }
        }
MyClass myObject = new MyClass(5);
```

initialises `myVar` to 5

(c) Some object oriented programming languages have methods which are known as destructors whilst others implement garbage collection. Compare and contrast these approaches.

(9 marks)

Answer pointers

A destructor is a method which is automatically invoked when the object is destroyed. It can happen when its lifetime is bound to scope and the execution leaves the scope, when it is embedded into another object whose lifetime ends, or when it was allocated dynamically and is released explicitly. Its main purpose is to free the resources (memory allocations, open files or sockets, database connections, resource locks, etc.) which were acquired by the object along its life cycle and/or deregister from other entities which may keep references to it.

Garbage collection is a form of automatic memory management. The garbage collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program

The use of destructors allows tighter control of memory when compared with garbage collection. The garbage collector is usually an independent process which runs when the system decides to run it and not when an object is released. A poorly implemented garbage collection mechanism can seriously slow down the execution of a program. On the other hand manual memory management is a difficult task which programmers struggle to do and therefore garbage collection speeds the development time.

Examiners Comments

This question examines part 2 of the syllabus: Concepts

Part a) of the question which asked for a definition of memory management appeared to be difficult for the majority of candidates despite the fact that the term is explicitly referenced in the syllabus. There seemed to be a lack of awareness that data manipulated by a computer program has to occupy space in a computer's memory and that allocation and deallocation of memory needs to be managed.

Given that candidates were generally unable to give a definition of memory management it was not surprising that they were unable to give examples of memory management in the programming language of their choice. Very few could identify the statements in the language that cause memory to be allocated.

The majority of candidates who answered the question were able to define the terms "destructor" and "garbage collection". Only some were able to contrast the memory management approaches implied by the terms.

Question A3

(a) In the context of the Object Constraint Language (OCL) define the following:

i) Invariant;

ii) Pre-condition;

iii) Post-condition;

(9 marks)

Answer pointers

An invariant is a condition that can be relied upon to be true during the execution of a program, or during some portion of it. It is a logical assertion that is held to always be true during a certain phase of execution.

A precondition is a condition or predicate that must always be true just prior to the execution of some section of code or before an operation in a formal specification. If a precondition is violated, the effect of the section of code becomes undefined and thus may or may not carry out its intended work.

A postcondition is a condition or predicate that must always be true just after the execution of some section of code or after an operation in a formal specification.

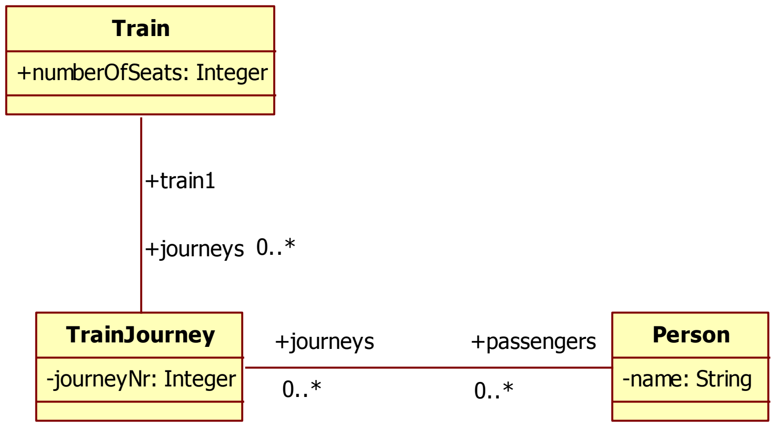
(b) Explain how OCL might be used in the design of an object oriented program.

(10 marks)

Answer pointers

It is good practice to produce a design for a program before coding it. Typically in object oriented programming a design might be realised by a set of Unified Modelling Language diagrams. The class diagrams would set out the names of the classes, the names of the variables and the names of the methods etc. Other diagrams would indicate other aspects of the behaviour of the system. Many of these diagrams are able to capture some aspects of the constraints which apply to the system. UML diagrams, however, are not sufficient to express all possible constraints nor are they able to precisely state the constraints they can represent. OCL can be used to provide further details of the requirements of system which goes beyond what can be achieved in a UML diagram and allows those details can be represented in a very precise way.

(c) Given the following UML diagram:



explain the following OCL statement:

context Train Journey
    inv: passengers->size() <= train.number Of Seats

(6 marks)

Answer pointers

The constraint says that for any train journey, the number of people travelling on a train making the journey must be less than or equal to the number of seats on that train.

Examiners Comments

This question examines part 3 of the syllabus: Design

This was the least popular question on the paper with only 17% of the candidates choosing to answer it. This may indicate a general unfamiliarity with OCL although the topic appears explicitly in the syllabus. Those who did choose the question were on average able to produce a satisfactory answer.

Part a) which asked for definitions of terms was generally well answered. Part b) which asked candidates to set out the benefits of using OCL was less well answered and a number of candidates were unable to provide an answer for this question.

Part c) is a simple example of OCL which has been used in previous papers and therefore would have been known to candidates whose revision included working through past papers and their solutions. Virtually all the candidates who attempted this part of the question were able to gain full marks for it.

Question B4

The *Library of Birmingham* is a public library that stores various items that can be borrowed, including books, journals, music, photographs and films. The Library is open to both members and non-members, but only members can borrow items. There is a limit of ten items in total that can be borrowed. Members must join first, by providing proof that they live in the Birmingham area. The library also provides facilities such as Wi-Fi and photocopiers, which both members and non-members can apply to use.

Books can be borrowed for two weeks and other items, such as music and films for one week. If the borrower keeps the item longer than this, they are subjected to a fine, which is increased daily.

When a member borrows an item, they provide their libraryNo, if this is valid their loan details are checked to ensure that the items to be borrowed will not take them over the maximum number of permitted items. A check is also made to see if they have any fines. If they have a fine, then they cannot borrow any items until the fine is paid. If all the checks are ok, then the item is issued to the member and the return date is assigned to the loan. At this point the member can optionally ask for a printout, which summarises all of the items they have on loan and when each item is due back.

Library members can reserve items that are currently out on loan. If an item proves to be very popular, then the librarian will order a new copy, provided the cost does not exceed the budget.

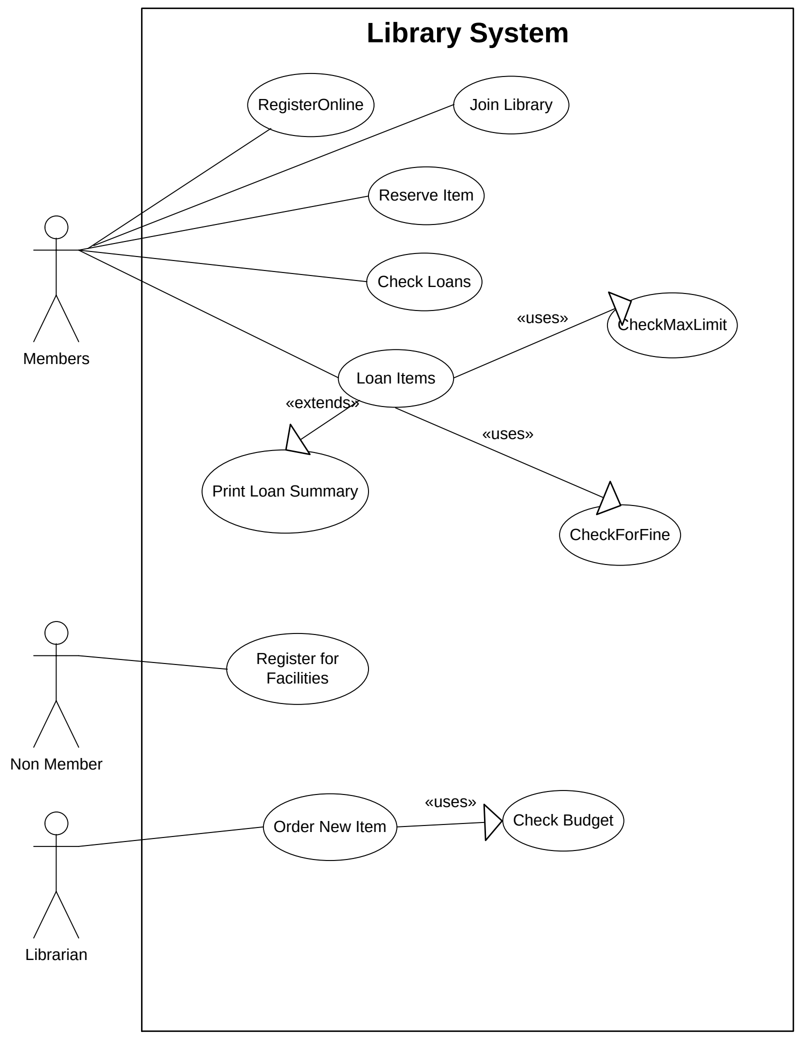
Members can register for online facilities so that they can check their own loan details at any time. Once registered a member can also renew their loans online, provided that the item has not been reserved.

(a) Draw a use case diagram for the library system

(15 marks)

Answer Pointers

Sample Use Case Diagram:



(b) Write down a use case description of the way a member borrows a book. Your answer should include a normal sequence and three alternative sequences.

(10 marks)

Sample use case description:

Pre-Condition
A member is registered with the library

Actor Action	**System Response**
1. User requests to borrow a book	2. Check libaryId is valid and provides libraryId
	3. Check user not exceeded their maximum limit
	4. Check user has no fines
	5. If ok, book is loaned to user and return date allocated

Alternative sequences

Step 2. Invalid card, loan refused.
Step 3. User has exceeded limit, loan refused.
Step 4. User has fine, loan refused.
Step 6. Request printout of loans

Examiners Comments

This question examines part 4 of the syllabus: Practice

This proved to be a popular question with the candidates, with 84% attempting it and 67% passing. Most candidates produced a good Use Case Diagram, some achieving full marks by identifying all the appropriate use cases, correctly linked to the correct actors. Marks were lost if the use cases were incorrectly linked to the wrong actors, or other use cases. Lower marks went to candidates who did not identify the use cases correctly, making them over-complex, for example, adding the steps that were more appropriate to part b, or connected the Extends/Uses relationships inappropriately.

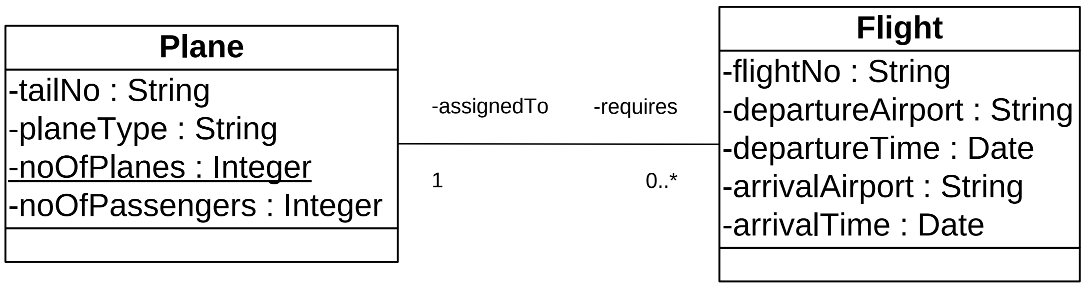
A very small minority provided the wrong type of diagram, such as a UML class diagram, which was inappropriate and lost marks.

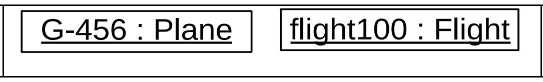
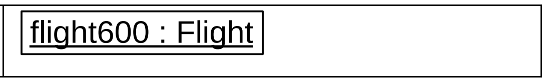
Part b asked to provide a description for the way a member can borrow a book, some candidates included use cases from other parts of the system, which were not always relevant to the steps of borrowing a book. A number of candidates chose not to answer this part, or incorrectly included a sequence diagram instead of a use case description.

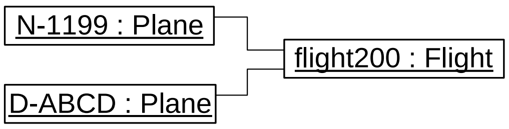
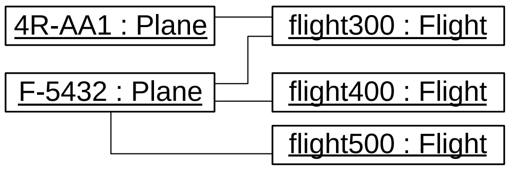
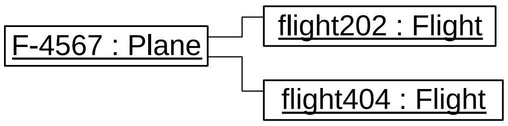
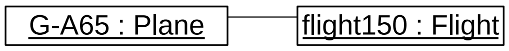
Question B5

(a) Given the class diagram below, 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)		v)	 4R-AA1 : Plane F-5432 : Plane flight300 : Flight flight400 : Flight flight500 : Flight
iii)		vi)	

Answer pointers

- i) Invalid because Flight must have a Plane. Ok for Plane not to be associated with a Flight.
- ii) Invalid because Flight can have only one Plane.
- iii) Ok
- iv) Invalid, because a Flight must have a Plane
- v) Invalid, because an Flight only has one Plane
- vi) Ok

(b) In an object-oriented programming language that you are familiar with, write code to implement the class diagram above. Within your code provide a constructor for each class that sets the variables to appropriate initial values. The class variable should be set and incremented appropriately.

(15 marks)

Sample code using Java:

```
class Plane{
    static int noOfPlanes = 0;
    String tailNo;
    String planeType;
    int noOfPassengers = 0;

    public Plane(){
            noOfPlanes++;
            tailNo = "Not known";
            planeType = "Not known";
            noOfPassengers = 0;
    }
}

import java.util.Date;

class Flight {
    String flightNo;
    String departureAirport;
    Date departureTime;
    String arrivalAirport;
```

```
        Date arrivalTime;
        Plane assigned;

        public Flight() {
                flightNo = "Not known";
                departureAirport = "Not known";
                departureTime = new Date();
                arrivalAirport = "Not known";
                arrivalTime = new Date();
                assigned = new Plane();
                }
}
```

Examiners Comments

This question examines part 4 of the syllabus: Practice

This question was less popular, with 72% attempting it and over 75% passing it. Most candidates identified the incorrect and correct object diagrams in part a, however, a common mistake was to think iv) was acceptable, which was not because a flight must have a plane associated with it, whilst a plane may exist without a flight.

Part b was sometimes not attempted. For those that provided code common mistakes included not representing the association between the two classes, or not defining the static variable correctly. Most candidates created the basic class definition, but those with lower marks often failed to provide the correct constructors.

Question B6

(a) In the context of object-oriented development, what is meant by the term *design pattern*? Explain the motivation for using them from a programmer's point of view

(10 marks)

Answer pointers

In OO terms a pattern is a named problem/solution pair that can be applied in new contexts with advice on how to apply it in novel solutions. Basically general principles and idioms codified in a structured format.

For a programmer design patterns are devices that allow programs to share knowledge about their design. Programmers encounter many problems that occur again and again and they often have to ask themselves is "how we are going to solve it *this* time". Documenting patterns can lead to reuse, possibly allowing the information to be shared with other programmers about how it is best to solve a specific program design problem.

(b) Explain what the terms *black-box testing* and *white-box testing* mean and how these techniques could be used to test object-oriented software. Include the advantages and disadvantages of each in your discussion and suggest other methods that can be used to test such software.

Answer pointers

Black-box is a software testing technique whereby the internal workings of the item being tested are not known by the tester.For example, in a black box test on software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications.

Advantages include (www.webopedia.com):

- The designer and the tester are independent of each other so should be unbiased
- The tester does not need knowledge of any specific programming languages
- The test is done from the point of view of the user, not the designer
- Test cases can be designed as soon as the specifications are complete

Disadvantages include:

- The test can be redundant if the software designer has already run a test case.
- The test cases are difficult to design.
- Testing every possible input stream is unrealistic because it would take an inordinate amount of time; therefore, many program paths will go untested.

White-box testing is a software testing technique where explicit knowledge of the internal workings of the item being tested is used to select the test data. White box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. The tester can then see if the program diverges from its intended goal. White box testing does not account for errors caused by omission and all visible code must also be readable.

For a complete software examination, both white box and black box tests are required, not just one. In O-O black box and white box testing can be applied to classes to test that the class meets its specification and that all branches within methods are exercised.

Examiners Comments

Part a of this question examines part 3 of the syllabus Design
Part b of this question examines part 1 of the syllabus: Foundations

This was attempted by 77% of the candidates attempted this question, with 47% passing.

A number of candidates did not attempt Part a, or gave the answer to a previous examination paper. To obtain a high mark the candidate needed to say what a design pattern was and discuss them from a programmer's point of view.

Part b was popular, whilst most candidates could describe what white and black box testing was, to gain a high mark, a discussion was needed of what the advantages and disadvantages were, and also to say what other methods could be used. A few candidates got black and white box testing the wrong way round.