THE BCS PROFESSIONAL EXAMINATIONS
BCS Level 5 Diploma in IT

October 2007

EXAMINERS' REPORT

OBJECT ORIENTED PROGRAMMING

Question 1

a) The majority of programs manipulate data structures. Explain the way in which object-oriented programming languages can be used to create data structures.

(5 marks)

Answer Pointers

A data structure is a specialized format for organizing and storing data. Data structures may be characterised by the operations that may be performed on them. For example a stack is characterised by the operations push and pop. In OO languages data structures are made available to users via classes which implement the methods which characterise the data structure. The classes describe complex objects whose constituent parts may themselves be complex objects.

b) During the execution of a program any data structures it uses must be assigned to some part of the computer's memory. Explain how this is achieved in an object oriented programming language with which you are familiar.

(5 marks)

Answer Pointers

The example given here uses Java.

The declaration:

```
MyClass myObject;
```

Allocates space for a reference to an object of type MyClass but at this point no memory is allocated to the data structure implemented by MyClass.

Memory is allocated by invoking the new operator:

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

At this point the system reserves enough memory to hold an object of MyClass and places a reference to it in myObject.

c) Some data structures (such as counters) need to be initialised when they are created. Show how this initialisation can be achieved using an object oriented programming language.

(5 marks)

Answer Pointers

Constructors can be used to initialise elements of a data structure.

```
class MyCounter{

        int counter;

        MyCounter(int start){
                counter = start;
        }
.
}
```

The statement

```
        MyCounter myObject = new MyCounter(3);
```

will set the initial value of the counter to 3.

d) Show how the mechanism you described in part c) operates when a data structure is derived from another simpler structure via single inheritance.

(5 marks)

Answer Pointers

```
class SimpleStructure{

        SimpleStructure(){
                .
                .
        }
        .
        .
}

class MoreComplexStructure extends SimpleStrucure{

        MoreComplexStructure(){
                .
                .
        }
        .
        .
}
```

If the following statement is executed:

```
MoreComplexStructure myObject = new MoreComplexStructure();
```

then the constructor of SimpleStructure is executed prior to the execution of the code in the constructor for MoreComplexStrucutre.

e) When a program has finished using a data structure, the memory it occupies can be returned to the pool of available memory. Describe a mechanism that achieves this without the programmer having to explicitly release memory.

(5 marks)

This mechanism is known as garbage collection. In the answer to part a) the precursor to allocating memory to a data structure was the creation of a variable to hold a reference to the object created by the new operator. In garbage collection when the system detects that there are no longer any references to an object or data structure it deallocates the memory used by that object.

Examiners Comments
(This question examines part 8b of the syllabus Concepts)

This question was designed to explore whether candidates were able to use object oriented concepts to construct data structures. It was answered by a minority of the candidates sitting the paper. The candidates who attempted this question fell into two categories. Either they understood the way in which data structures are described and allocated memory in object oriented programming or else they had little grasp of the concepts addressed by the question. The first category of candidate obtained very good marks the second attracted extremely low marks. Very few candidates lay outside of these two groups. Candidates who scored high marks demonstrated evidence of having practical experience of programming in an object oriented language.

Question 2

a) Explain the effects of the following modifiers in the declaration of variables and methods:

i) public;
ii) private;
iii) protected.

(6 marks)

Answer Pointers

Public variables and methods are accessible anywhere within a program. Private variables and methods may only be accessed within the class in which they are declared. Protected variables and methods may be accessed from within the class in which they are declared and any subclasses of the class.

b) A class MyClass has only one constructor. Discuss the way the class might be used if its declaration is as follows:

```
class MyClass {

        private MyClass(){
                .
                .
        }
        .
        .
        .
}
```

(5 marks)

Answer Pointers

Here the only constructor is declared as private. Consequently the statement:

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

can only be valid if it is within the code in the declaration of MyClass. The only place where such code would make sense is in a class method of MyClass. Such a mechanism could be used to limit the number of instances of a class that may be created and is most often used to implement the Singleton pattern.

c) Justify the inclusion of a mechanism for supporting encapsulation in an object oriented programming language.

(7 marks)

Answer Pointers

Encapsulation is a mechanism for hiding the internals of an object from a program that is using that object and only presenting those operations which may be used in a consistent and safe way. Much of the art of programming is the assembly of small simple structures into more complex structures. These complex structures must be manipulated correctly if they are to make sense. For example a queue is not a queue if it is possible to select an element from the middle of the data structure. Classes implementing objects provide an interface which is sufficient and complete in terms of the operations which may be executed on objects of that class. It is therefore unwise to allow any other type of manipulation of the object. Encapsulation ensures that the only operations carried out on data are those operations which leave the data in a valid state.

d) Write a set of guidelines that set out good practice when writing code in a language that supports encapsulation.

(7 marks)

Answer Pointers

In general, unless there is a very good reason, all variables should be declared as private. Every variable should have a method to set its value and another to get its value. Such methods are known as getters and setters. If these methods have been created there is no reason why any variable would be accessed directly by any method. This includes constructors. Every class will have a set of methods which it makes available to other objects. These methods must be declared public. For stand alone classes internal methods will be declared as private. Where a class is a superclass, if its internal methods are used by subclasses they should be declared as protected. Typically getters and setters within a superclass will be declared as protected unless they are intended for use by other external classes.

Examiners Comments
(This question examines parts 8a and 8b of the syllabus Foundations and Concepts)

This question was answered by the majority of candidates. Part a) was generally answered well and candidates obtained the majority of the marks available. Part b) examined the candidates' awareness of the contribution that encapsulation makes to the Singleton design pattern. Many candidates had encountered this pattern (which is specifically mentioned in the syllabus) and therefore were able to make a good attempt at this part of the question. Parts c) and d) were less well answered. Perhaps the most common reason for this was the candidates' common misconception that the private modifier is a security mechanism which somehow prevents external parties gaining access to an organisation's data. Candidates should be made aware that the purpose of access modifiers in object oriented programming is to prevent programmers from accessing the internals of an object and unwittingly causing unexpected side effects. That is, they are an aid to the programmer rather than a mechanism to prevent hacking.

Question 3

a) Explain the meaning of the following terms within the context of an object oriented programming language:

i) method;
ii) message;
iii) overloading;
iv) overriding.

(8 marks)

Answer Pointers

A method is an operation on an object.

A method is invoked by sending a message to an object. The message will contain any data which the method requires in order to execute.

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.

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 a method in its superclass.

b) The terms in part a) of this question describe aspects of programming that are implemented in all the major object-oriented programming languages. Similar concepts are not found in structured programming languages. What advantages do object-oriented languages gain through the implementation of these features?

(8 marks)

Answer Pointers

Structured programming languages imposed a decomposition of task which was supported by code chunks known as subroutines and/or functions. At a point in the programs execution control was directed to another part of the code and was eventually returned after a task had been undertaken. This was not far removed from the concept of a GOTO command where control moved to a particular section of the program. The concept of method is more closely associates with the data it operates on than subroutines and functions were. A method can be more easily insulated from the rest of the program because it is an operation which can be understood within the context of the data associated with the object. A message is a higher level concept than a subroutine call because it is not associated with a jump to a particular address, instead it is simple a request to an object to perform a predefined operation. Since subroutines and functions were associated with memory addresses it followed that each address would require a separate name. As however method and message are not associated with addresses it is possible to introduce the useful concepts of overloading and overriding which allow the same name to be used for operations that are related to each other but which have different implementations.

c) Explain how overloading and overriding contribute to the implementation of polymorphism in object oriented languages.

(9 marks)

Answer Pointers

Polymorphism is the idea of allowing the same code to be used with different types, resulting in more general and abstract implementations

Overloading allows multiple functions taking different types to be defined with the same name; the compiler or interpreter automatically calls the right one. This way, functions appending lists of integers, lists of strings, lists of real numbers, and so on could be written, and all be called *append*—and the appropriate *append* function would be called based on the type of lists being appended. This differs from parametric polymorphism, in which the function would need to be written *generically*, to work with any kind of list. Using overloading, it is possible to have a function perform two completely different things based on the type of input passed to it.

Overiding is used to implement parametric polymorphism. Using parametric polymorphism, a function or datatype can be written generically so that it can deal equally well with objects of various types. Object oriented languages employ the idea of *subtypes* to restrict the range of types that can be used in a particular case of parametric polymorphism. In these languages, subtyping polymorphism allows a function to be written to take an object of a certain type *T*, but also work correctly if passed an object that belongs to a type *S* that is a subtype of *T* (according to the Liskov substitution principle).

Examiners Comments
(This question examines parts 8a and 8b of the syllabus Foundations and Concepts)

Part a) of the question asked for the definitions of terms used in object oriented programming. This was well answered and candidates in general obtained good marks for this session. Parts b) and c) explored the advantages that object oriented languages gain by implementing the features set out in part a). Candidates fared less well in providing answers for these parts of the question. In particular, very few candidates were able to identify why object oriented languages prefer message passing to subroutine and function calls as found in languages such as C, FORTRAN and COBOL. Polymorphism was understood to a limited extent and the more able candidates were successful in defining the term and explaining the two ways it is exhibited in object oriented programming.

Question 4

Zodak Printers are a company that specialises in digital printing over the web. New customers first have to register to use the system by supplying a username, password and their email address, a confirmation email is sent to this address and customers must respond to the email to complete their registration. Once validated new customers can then use the system.

All customers must login using their username and password; if these match the stored details they can then access the system.

The customer must create an album for each set of photographs they want to upload for printing. Each customer has a file storage limit of 200mb, if they exceed this limit, they can not load any further photographs.

Once the customer has loaded their photographs, they can place their order by choosing how many copies of each photograph they want and what size. Then they can proceed to check out, where they will be prompted for their delivery and credit card details. If these are validated, then their photographs are put on a queue for printing. The customer can optionally print out the details of the order.

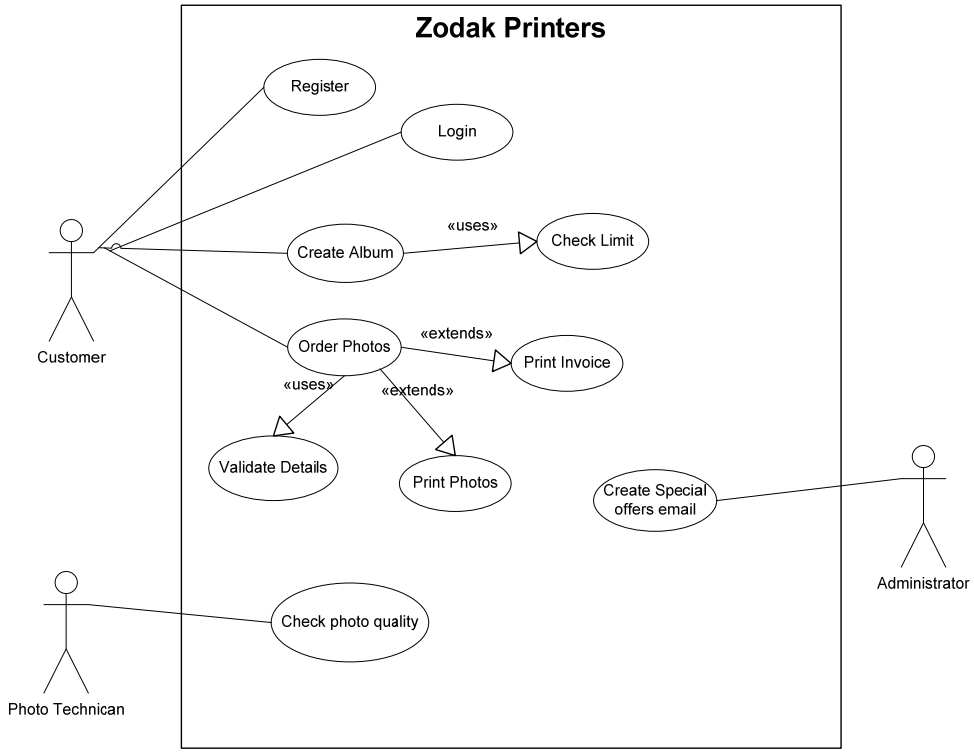
At regular intervals a photo technician checks the quality of the photographs being printed and will add the appropriate toner if required.

Once a week, the Administrator sends out an email of special offers to customers who have not placed an order for 3 months.

a) Draw a use case diagram for this system.

(15 marks)

Answer Pointers



b) Discuss how use case diagrams and descriptions provide an overview of the user requirements of a system. Within your answer include examples from the above system.

(10 marks)

Answer Pointers

The candidate should discuss where use cases should be used in developing a system. For each use case the developer should create a set of scenarios. Scenarios are natural language descriptions of an instantiation of the use case.

These can be used in several 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

The candidate should include a brief example of a scenario to illustrate their points.

Examiners Comments
(This question examines parts 8c of the syllabus Design)

This question was answered by the majority of candidates. Part a) involved drawing a use case diagram for the Zodak system. In general, this part was well answered and candidates obtained good marks for this section, identifying 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 omitted some of the use cases; 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 just provided use case descriptions, which was not required, unless they were used as an example to backup any points made, e.g., to show an example of providing fuller details of the requirements.

Question 5

A Bank wishes to keep information on its customers. The proposed Customer class has the following instance variables:

customerNo: String
customerName: String
date of Birth Date
credit rating: Integer

Customer number and name are string data types, credit rating is a number between 0-20, where 0 represents customers not yet rated.

A class variable is also required, called noOfCustomers, which will be incremented each time a Customer instance is created.

Using an object-oriented programming language that you are familiar with, write code to perform the following. Where appropriate include suitable integrity checks:

i) Show the declaration of the Customer class, including any *setters* and *getters* methods.

(15 marks)

Answer Pointers

```
import java.util.Date;

class Customer {
    String customerNo;
    String customerName;
    Date dob;
    int credit_rating;
    static int noOfCustomers = 0;

/* getters */

    String getCustomerNo() {
      return customerNo;
    }
    String getCustomerName() {
      return customerName;
    }
    Date getDOB() {
      return dob;
    }
    int getCreditRating() {
      return credit_rating;
    }

/* setters */

    void setCustomerNo(String aName) {
      customerNo = aName;
    }
    void setCustomerName(String aName) {
      customerName = aName;
    }
    void setCredit_rating(int credit) { /* or may return an error code */
      if ((credit < 0) || (credit > 20)) {
```

```
            System.out.println("Credit Limit not between 0-20");
            System.out.println("Data not added")
            }
      else
      credit_rating = credit;
   }
   void setDOB(String aDate){
      dob = new Date(aDate);
   }
}
```

ii) Declare two constructors as follows, both constructors should increment the class variable appropriately:

- The first is a default constructor that has no parameters and sets the instance variables to either "not known" for the strings, 0 for the integer and 1st January 1900 for the date (assume there is a Date constructor that accepts dates in a string format).
- The second takes 4 parameters, one for each of the instance variables.

(8 marks)

Answer Pointers

```
   public Customer() {
      customerNo = "Not known";
      customerName = "Not known";
      credit_rating = 0;
      dob = new Date("01-Jan-1990");
      noOfCustomers++;
      }
   public Customer(String cno, String cname, Date aDOB, int cr) {
      setCustomerNo(cno);
      setCustomerName(cname);
      setDOB(aDOB);
      setCredit_rating(cr);
      noOfCustomers++;
      }
```

iii) Show how both constructors could be instantiated.

(2 marks)

Answer Pointers

```
Customer cust1 = new Customer();
Customer cust2 = new Customer("0123",'"Joe Smith", "01-Feb-1978",5);
```

Examiners Comments
(This question examines parts 8d of the syllabus Practice)

This proved to be a popular question, answered by a large percentage of the candidates, some of whom obtained full marks. Most candidates could provide the basic class description and constructors; most marks were lost on defining the setter and getter methods. Some omitted these completely, or tried to merge the setters and getters into one method, rather than providing an individual method for each variable. Other marks were lost by not defining the class variable or did not declare any integrity checks on the credit rating. The candidates who obtained high marks demonstrated evidence of having practical experience of object-oriented programming, such as Java or a .NET language.

Question 6

The Unified Modelling Language 2.0 (UML) comprises of 13 different diagrams. These can be broadly categorised as:

i) Structure diagrams
ii) Behaviour diagrams
iii) Interaction diagrams

a) For each category, give a description of one diagram that falls into the category; include a simple example of its use and explain when you would use it.

(15 marks)

Answer Pointers

The following indicate which types of diagrams fall into each category:

Structure Diagrams emphasise what things must be in the system being modeled:

- Class diagram
- Component diagram
- Composite structure diagram
- Deployment diagram
- Object diagram
- Package diagram

Behavior Diagrams emphasize what must happen in the system being modeled:

- Activity diagram
- State Machine diagram
- Use case diagram

Interaction Diagrams, a subset of behavior diagrams, emphasize the flow of control and data among the things in the system being modeled:

- Collaboration (UML 1.x)/Communication diagram (UML 2.0)
- Interaction overview diagram (UML 2.0)
- Sequence diagram
- UML Timing Diagram (UML 2.0)

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.

b) Discuss how object-oriented code can be tested; within your answer explain which UML diagrams can be used to aid testing.

(10 marks)

Answer Pointers

This is an open-ended question. For the first half, the candidate may look at different approaches, for example:

- Fault based testing
- Scenario based testing

Or the answer could include a discussion of black-box and white-box testing with respect to OO programming.

The candidate must discuss which UML diagrams can be used for testing, e.g., use case scenarios.

Examiners Comments
(This question examines parts 8c & 8d of the syllabus Design and Practice)

This question proved less popular with candidates. It tested the candidate's ability to identify which UML diagram fell into each of the three categories. For part A, the most popular diagrams were class diagrams, use case diagrams and sequence diagrams for the three areas. In general, the candidates could give a description of the diagram, but then lost marks by not giving an example of its use or omitted to explain where they could be used. For example, class diagrams are the primary view for portraying the structural model, capturing the classes, their behaviour and relationships to other classes in the system. Some marks were lost where candidates incorrectly allocated the diagram to the wrong category, for example, including a class diagram in the interaction section.

In part B, a lot of candidates just discussed white and black box testing, however, marks were lost if this was not related to object-oriented coding. To get a higher mark, the answer must discuss how UML diagrams help test a system, for example, use case scenarios.