**BCS THE CHARTERED INSTITUTE FOR IT**

**BCS HIGHER EDUCATION QUALIFICATIONS**

**BCS Level 5 Diploma in IT**

**EXAMINERS' REPORT**

**Object Oriented Programming**

**Question A1**

a) Define the following terms:
   i) Abstract data type;
   ii) Encapsulation;
   iii) Typed language;
   iv) Coupling;
   v) Cohesion.

**(10 marks)**

## Answer Pointers

Abstract data types or ADTs are data types described in terms of the operations they support—their interface—rather than how they are implemented

Encapsulation supports the notion of gathering together related aspects of a design, grouping them and hiding the implementation.

In typed languages, there usually are pre-defined types for individual pieces of data (such as numbers within a certain range, strings of letters, etc.), and programmatically named values (variables) can have only one fixed type, and allow only certain operations: numbers cannot change into names and vice versa..

Coupling refers to the degree to which each program module relies on each other module.

Cohesion refers to the degree to which each part of a module is associated with each other part, in terms of functional relation.

b) A stack is a last in, first out linear data structure. A stack can have any object as an element. It is characterised by two fundamental operations, called push and pop. The push operation adds a new item to the top of the stack. If the space allocated to hold the stack is full when the push operation is attempted then an error condition is raised. The pop operation removes an item from the top of the stack. A pop reveals previously concealed items, or results in an empty stack. If the stack is empty when a pop operation is attempted then an error condition is raised (it means no items are present in stack to be removed).
   Using an object oriented programming language with which you are familiar, write code which implements a stack. Your code should store the stack elements in an array and should not make use of a stack class from a class library.

**(15 marks)**

**Answer Pointers**

```
public class Stack
{
   private Object[] stackArray;
   private int maxSlot;
   private int currentSlot = 0;

   public Stack(int size)
   {
      maxSlot = size;
      stackArray = new Object[size];
   }

   public void push(Object o) throws OverflowException
   {
      if (currentSlot == maxSlot)
      {
         throw new OverflowException();
      }
      stackArray[currentSlot] = o;
      currentSlot++;
   }

   public Object pop() throws UnderflowException
   {
      if (currentSlot == 0)
      {
         throw new UnderflowException();
      }
      currentSlot--;
      return stackArray[currentSlot];
   }
}
```

**Examiner's Comments**

**This question examines part a of the syllabus: Foundations**

This question was answered by more than 70% of the candidates. Unfortunately, there were few high scoring answers. Part a) was book work and the definitions that were looked for have appeared in past papers. The answers required were identical to those in previous examiners' reports. Despite this, only a few candidates gained the full marks for this part of the question. The majority of candidates who answered this question chose not to provide an answer for part b) therefore sacrificing 15 marks. The reason for this is not clear. A stack is the simplest data structure covered in standard programming courses and for those who were uncertain as to what a stack is, a definition was given.

**Question A2**

a) Define the following terms:
   i) Class;
   ii) Object;
   iii) Data member;
   iv) Member Function;
   v) Single inheritance;
   vi) Multiple inheritance.

**(12 marks)**

**Answer Pointers**

A class is a template for the construction of an object. It defines the data an object stores and the operations which can be applied to the data.

An object is an instantiation of a class. It has memory allocated to it to record its state. It responds to messages passed to it.

An object will have a number of data members which define the data held by the object.

An object also has a number of member functions which specify the nature of the operations on the object.

In single inheritance a subclass may inherit data and operations from a single superclass

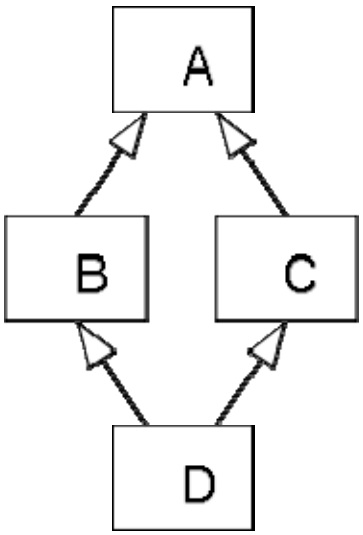
In multiple inheritance a subclass may inherit data and operations from a number of superclasses.

b) Languages which support multiple inheritance can be used to create ambiguous references to methods. Give an example of such a situation.

**(6 marks)**

**Answer Pointers**

Given the following classes:




If D invokes a method from A and B and C have overridden the method in different ways, which version of the method executes?

c) Discuss whether support for multiple inheritance is a necessary prerequisite for support for polymorphism in an object oriented programming language.

**(7 marks)**

**Answer Pointers**

Polymorphism is the ability of different objects to behave in different ways when the same message is applied to them. It is clear that multiple inheritance is not a necessary prerequisite for polymorphism as languages which can only support single inheritance exhibit polymorphism. Multiple inheritance might however, appear to be a way in which one could enforce an object to support a given set of methods. If class X supports a set of methods then any class which inherits from X will also support those methods and hence can respond to same messages as any other subclass of X.

**Examiner's Comments**

**This question examines part b of the syllabus: Concepts**

This was the most popular question on the paper although it did not have the highest average mark. As with question A1, part a) was bookwork which as appeared in previous papers with the answers contained in the associated examiners' report. There were more completely correct answers to part a) of this question than for part a) of question A1. Part b) was also bookwork though perhaps covering material more difficult to remember than simple definitions. The majority of candidates had clearly encountered the material in their studies and were able to express an adequate answer. Part c) required a discussion and although it dealt with issues the candidates were certain to be aware of, required an answer which is probably not to be found in standard text books. In general this was answered poorly although all candidates who were able to define polymorphism received some credit for this.

**Question A3**

a) Describe the way in which the Unified Modelling Language (UML) can be used to design an object oriented program.

**(10 marks)**

**Answer pointers**

UML can be used to specify, visualize, modify, construct and document the artefacts of an object-oriented software-intensive system under development

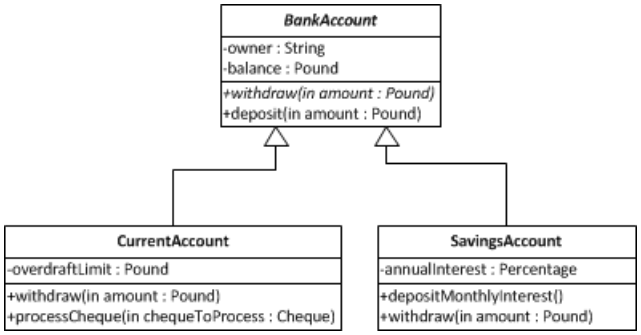
Use case diagrams can be used to define the way in which the program and external actors interact to achieve a goal.

Class diagrams can be used to show the classes the program uses, their data, their operations and their interactions.

Object interaction diagrams can be used to illustrate how individual objects may interact during run time.

Object constraint language (OCL) may be used to formally specify the behaviour of parts of the system.

b) Using an object oriented programming language of your choice, write code which represents the UML diagram shown below:




**(15 marks)**

## Answer pointers

```
public abstract class BankAccount
{
   private String owner;
   private Pound balance;

   public abstract void withdraw(Pound amount);

   public void deposit (Pound amount)
   {
      //Code here
   }
}

public class CurrentAccount extends BankAccount
{
   private Pound overdraftLimit;

   public void withdraw(Pound amount)
   {
      //Code here
   }

   public void processCheque (Cheque chequeToProcess)
   {
      //Code here
   }
}

public class SavingsAccount extends BankAccount
{
   private Percentage annualInterest;

   public void withdraw(Pound amount)
   {
      //Code here
   }

   public void depositMonthlyInterest()
   {
      //code here
   }
}
```

**Examiner's Comments**

**This question examines part c of the syllabus: Design**

This question was answered by 85% of candidates and had the highest average mark of any question on the paper. In part a) most candidates were able to identify five types of UML diagram and give a brief explanation of their use. Part b) was also answered well although some candidates failed to recognize that withdraw() was an abstract method and therefore BankAccount would be an abstract class.

## Question B4

Consider the following class definition, written by a novice programmer, that represents a specific working day in a 12-week project period.

```
public class DayAndWeek
{
  public int day;  // Day of the Week, Varies from 1-5
  public int week; // Week of the Project, Varies from 1-12
};
```

a) Provide a code fragment showing a redesigned DayAndWeek class that more appropriately encapsulates the two instance variables.

**(5 marks)**

### Answer pointers

```
public class DayAndWeek
{
  private int day;
  private int week;
}
```

b) Provide instance methods that will enable the two instance variables to be queried and changed via an object of the class.

**(5 marks)**

### Answer pointers

```
int getDay()                { return this.day;        }
void setDay(int day)        { this.day = day;         }
int getWeek()               { return this.week;       }
void setWeek(int week)      { this.week = week;       }
```

c) Incorporate an error-checking routine into your instance variable change methods (above) that prevent data items from assuming invalid values. This may be accomplished either by returning a flag to signify a successful/failed update, or using exception handling.

**(5 marks)**

## Answer pointers

```
bool setDay(int day)
{
  if(day>=1 && day <=5)
  {
    this.day=day;
    return true;
  }
  return false;
}

bool setWeek(int week)
{
  if(week>=1 && week <=12)
  {
    this.week=week;
    return true;
  }
  return false;
}
```

d) Provide an instance method that will ensure the instance variables day and week will be initialised to default values (1,1) immediately upon creation of a new instance.

**(5 marks)**

## Answer pointers

```
DayAndWeek() { this.day=1; this.week=1; }
```

e) Demonstrate the use of the redesigned DayAndWeek class you developed above by creating and configuring an object, including checking the success/failure of the instance variable changes you apply.

**(5 marks)**

## Answer pointers

```
public static void main(String args[])
{
  DayAndWeek aDay = new DayAndWeek();

  if(aDay.setWeek(3))
    System.out.println("Updated Week Successfully");
  else
    System.out.println("Week Update Failed");

  if(aDay.setDay(5))
    System.out.println("Updated Day Successfully");
  else
    System.out.println("Day Update Failed");
}
```

## Examiner's Comments

**This question examines parts d of the syllabus: Practice**

This question was attempted by approx. 1/3 of candidates, with varying degrees of success. Some answers were perfect, scoring all available marks, whilst others were very confused, alluding to a lack of practice at constructing

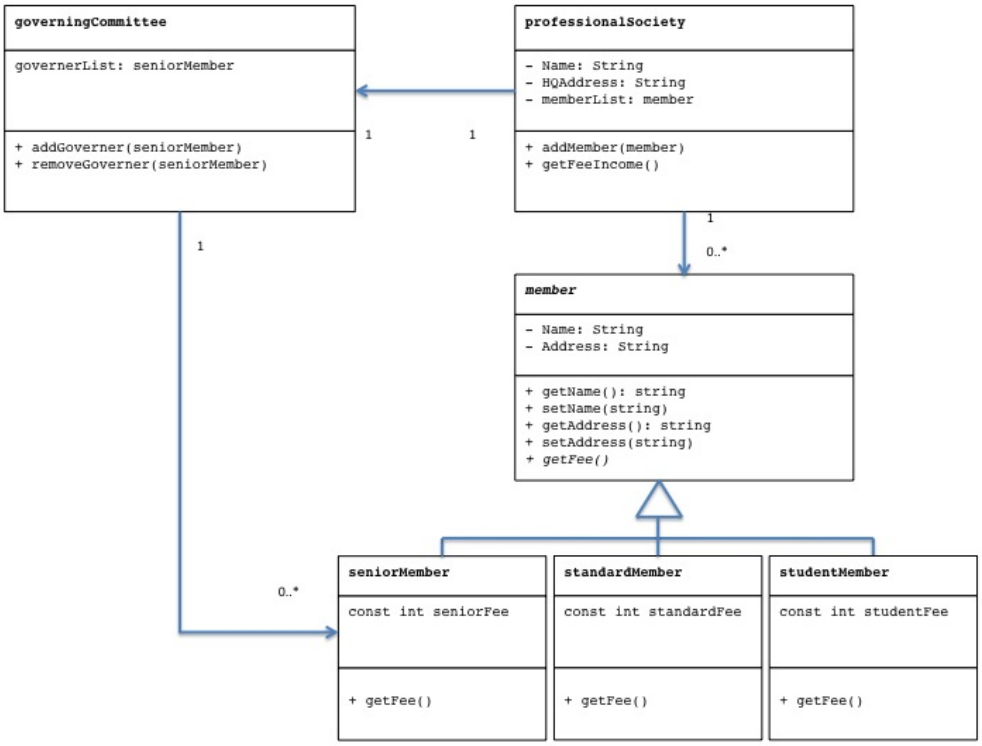
simple object oriented programs. Most candidates had realised that the data members of the class should not have been declared public, and provided a more secure version of the class in question (a). However, as the questions became more complex, a significant proportion of candidates provided answers that seems to show a lack of understanding of the question, perhaps because they were not aware of the meaning of some of the technical terms used in object oriented programming, or had insufficient familiarity with a programming language to fashion an answer.

**Question B5**

a) Construct a UML class diagram showing the structure of a professional society, wherein members pay an annual fee. Your class diagram should incorporate the following 6 classes: member, studentMember, standardMember, seniorMember, society, and governingCommittee, which should be connected with appropriate relationships, and be populated with appropriate instance variables and methods to enable the names, addresses and fees of members to be stored, along with the management committee members, and the name and HQ address of the society. The governing committee will comprise a number of senior members.

**(20 marks)**

**Answer pointers**




Above is shown valid basic structure for the class diagram; any reasonable variant would also be accepted.

If the candidate correctly assembles a member inheritance hierarchy and uses the correct symbols for inheritance, up to 10 marks were awarded.

If the candidate has other appropriate relationships connecting the member hierarchy to the other classes, including their multiplicity, up to 10 marks were awarded.

If the candidate peppers the classes with appropriate instance variables and methods, and correctly designates them as private, public or protected, up to 5 marks were awarded.

b) A colleague suggests that the member class is abstract. What do they mean by this, and how is an abstract class represented in a UML class diagram?

**(5 marks)**

## Answer pointers

The member class is abstract because it is the super-class of an inheritance for specification hierarchy, giving rise to sub-classes standardMember and seniorMember. It will never be instantiated, and is there to hold shared properties between standardMember and seniorMember classes only, and potentially to define methods that must be implemented by those sub-classes. An abstract class name is shown in italics in a class diagram, and sometimes annotated with the label <<abstract>>.

## Examiner's Comments

### This question examines part c of the syllabus: Design

Consistent with previous years, the question requiring that candidates construct a class diagram representing a described system remained popular. It tested knowledge and understanding of issues in object oriented design, focusing particularly on UML. It was attempted by 85% of candidate, and achieved a good pass rate (more than two-thirds of candidates won 40% of the available marks or more). Most candidates provided a diagram with the classes requested, but some did not read the question carefully enough and included other classes whilst neglecting to include those specifically requested. One or two candidate provided a use-case diagram instead of a class diagram, and scored 0 marks. Most candidates realised that there was an inheritance hierarchy connecting the different types of member, but some confused the symbols used to represent inheritance, composition and aggregation, meaning that some class diagram were improperly interconnected. The majority of candidates were familiar with the concept of abstract classes, and provided a reasonable answer to part (b).

**Question B6**

a) Some object-oriented programming languages, such as Python, are interpreted. Others, such as C++, are compiled. Distinguish between the terms *interpreted* and *compiled*.

**(6 marks)**

**Answer pointers**

In an interpreted language, the code you write is saved as source code, and executed by conversion to machine code at run-time. The interpreter must be present to enable this. [for 3 marks]

Conversely, in a compiled language, source code is converted into machine code program before it can run, and can thereafter be run without the presence of a development environment/compiler. [for 3 marks]

b) Polymorphism is broadly classified into two types: *parametric* and *ad-hoc*. Distinguish between these two forms of polymorphism.

**(6 marks)**

**Answer pointers**

Ad-hoc polymorphism refers to the scenario in which polymorphic code must be explicitly provided by the programmer to account for each different data type that the code may encounter (e.g., function overloading). It is said that this kind of polymorphism vanishes upon closer inspection of the code. [for 3 marks]

In parametric polymorphism, on the other hand, code can be written that can operate upon any data type, dynamically adapting its behaviour without the requirement for specific, detailed instructions that describe how to deal with each individual data type that may be encountered (e.g., template classes). [for 3 marks]

c) Distinguish between *mutable* and *immutable* objects, providing one example situation in which each would be an appropriate choice.

**(6 marks)**

**Answer pointers**

An immutable object has a state that cannot be modified following its creation; e.g., the values of its instance variables are fixed. [for 3 marks]

Conversely, a mutable object can be updated as required to assume a new state when needed, accomplished by setting (mutating) its instance variables. [for 3 marks]

d) One of the principal features of object oriented languages is subtype polymorphism. In this context, explain the terms *covariant*, *contravariant*, and *invariant*.

**(7 marks)**

## Answer pointers

Covariant describes the ability to convert from a more generic to a more specific types (e.g., from book to textBook).

Contravariant describes the ability to convert from a more specific to a more generic type (e.g., from circle to shape).

Invariant describes an inability to convert between types.

## Examiner's Comments

**This question examines part a & b of the syllabus: Foundations and Concepts**

This question, which tested candidates knowledge of basic object oriented programming terminology and concepts, was attempted by less than 1/3 of candidates (being the least popular question in this year's paper). Furthermore, those that did attempt it did not score well. Although most were able to provide a reasonable description of the different between compiled and interpreted languages, other questions were mostly answered incorrectly. Some candidates confused mutable/immutable with private and public access restriction. No candidates answered question (d) correctly, and the question asking for ad-hoc and parametric polymorphism to be distinguished was poorly answered, despite featuring in previous years exams.