THE BCS PROFESSIONAL EXAMINATION
Diploma

April 2005

EXAMINERS' REPORT

Object Oriented Programming (Version 1)

The Version 1 syllabus will no longer be examined and has been replaced by Version 2.

Question 1

1. *a)* Explain the process of *iterative and incremental development*. **(8 marks)**

Answer Pointers

Candidates were expected to describe a process similar to prototyping where an application is developed in small chunks. A component of the system is quickly developed (perhaps only partially) and then tested. The component may then be revised or enlarged. A good candidate might have remarked that this approach is similar to Boehm's spiral model of development.

b) How does incremental development help contain the risks inherent in system development? **(7 marks)**

Answer Pointers

The issue here is that only small parts of the system are built at any one time. In a 'big bang' approach success or failure cannot be assessed until the project is complete. By then an enormous amount of money may have been spent in developing the product. Incremental development reduces risk by introducing a number of points at which the success of the project and whether it is fulfilling requirements can be assessed.

c) Discuss the suitability of the use of object technology in an iterative development process. **(10 marks)**

Answer Pointers

Object-oriented technology, by its very nature 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.

Examiner's Guidance Notes

The majority of answers to this question attracted good marks. Some answers to part (c) of the question, whilst being correct did not make all the points being sought.

Question 2

2. Explain what is meant by ANY FIVE of the following used in the context of object-oriented development, giving suitable examples:

i) Public, protected and private visibility of class methods and attributes
ii) The principle of substitution
iii) Polymorphism and dynamic binding
iv) Designing to an interface
v) No concrete superclasses
vi) Templates and/or generic classes **(25 marks)**

Answer Pointers

(i)
The answer needs to highlight the distinction between these levels of visibility and how they are deployed by a developer.

The public features of a class are visible to all other classes. A public method of one class can be called by a method in another class. Equally, a public attribute can be referenced and modified from elsewhere. This is generally not wise since it exposes the implementation of a class to others. Constant public attributes are generally considered safe.

The private features of a class can only be referenced by that class. For example, a private attribute can only be referred to in the body of any of the class methods. Generally, privacy of attributes is used to secure the implementation of a class. A private method is used in a support role for the other class methods.

The protected features of a class are private to other classes and public to subclasses. This way a subclass can directly reference the features of its immediate superclass.

(ii)
The answer should highlight the flexibility and adaptability achievable through this technique.

The substitution principle was first enumerated by Liskov. The principle permits an instance of a subclass to be used where an instance of a superclass is required. For example, a method expecting an instance of a Person object as a parameter may be invoked with an instance of an Employee object, where class Employee is a subclass of Person.

Substitution makes systems more adaptable to change. For example, an object model in which there is a one-to-many relation between Bank class and Account class, could at run time relate a Bank instance with various instances of subclasses of Account, such as CurrentAccount or DepositAccount. Further, the Account class hierarchy could be extended both horizontally and vertically without impact on the Bank class.

(iii)
This questions follows from part (b). Here, we are looking to explore the benefits of the effect of dynamic binding.

Polymorphism = many forms. This captures the notion of substitutability. A Bank object may associate with many Account objects. The latter may be a mix of different types of accounts that are specialisations of the Account class.

A Bank object is permitted to send the same message to each of its Account objects. The actual behaviour will be determined by the type of the recipient object. In the Account class hierarchy a method, possible deferred, will be introduced in the Account superclass and redefined in one or more subclasses. The actual method executed will be determined by the type of Account object receiving the message.
(iv)
The answer needs to identify that we aim to decouple an abstraction from its implementation.

The key to this issue in OO is to define interface classes at the root of a class hierarchy. An interface is characterised by having only deferred methods. In a particular problem a concrete subclass will provide a realisation of that interface.

This provides the flexibility whereby we can extend the interface and provide further realisations of that new interface. A new application can operate to that new interface and the first remains unchanged.

(v)
The rationale for this principle should be the aim of the answer.

A principle related to part (d) is that no superclasses in a class hierarchy should be concrete. Violation of this principle can lead to significant problems as a design evolves. The problem that frequently arises is that the superclass ends up having more than one role, namely an interface for all its superclasses and a default implementation for one or more of its methods. In particular this happens when specialised functionality must be provided with instances of the concrete superclass.

We can avoid this situation by making all superclasses abstract.

(vi)
A template or generic class lets us prepare a family of classes by providing the definition one.

The most common example of these classes are the container classes used in OO modelling. A container might be required to hold a collection of bank accounts or a collection of employees. Rather than describing each container separately, we define a container in terms of some arbitrary type. The generic container can then be instantiated any number of times to create the particular container required by the application.

Examiner’s Guidance Notes
In general candidates were aware of the majority of these terms. Part (i) received very good answers. Candidates were less sure of the definition of generic classes, dynamic binding and the principle of substitution.

Question 3

3. **Figure 1** below is a representative class diagram that we might construct as part of an object oriented analysis and design. In the diagram we have a number of Employees that are employed by a Company. Some employees are Managers with responsibility for a team of employees. Such a typical class diagram shows class symbols, association relations and a specialisation relation.

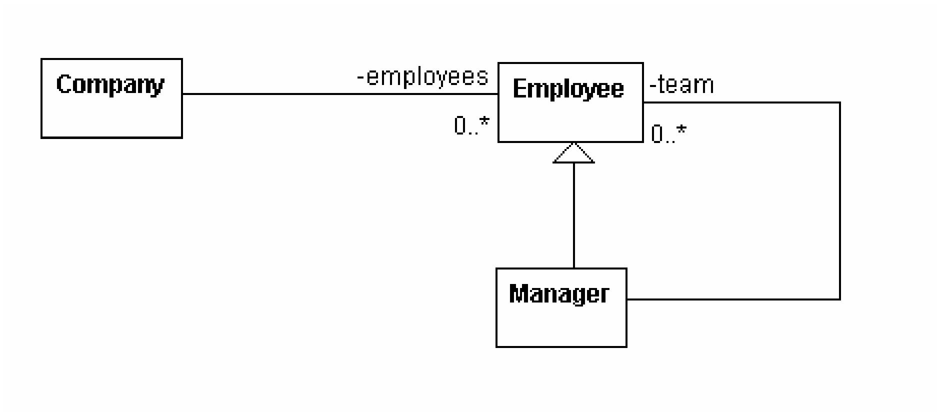
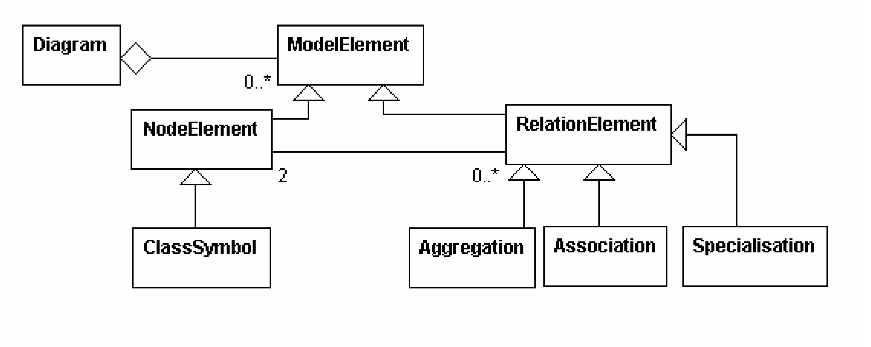


Figure 1

a) Prepare a class diagram for a graphical computer aided software engineering (CASE) tool to Compose and edit typical class diagrams (as shown above). In support of this *meta-class diagram* (i.e. a class diagram of a class diagram) you must provide an analysis of the objects in the problem and the relationships that exist between them. **(10 marks)**

Answer Pointers

An example of a suitable class diagram is given below. A diagram consists of any number of model elements. In turn, the latter can be either a node or a relationship. A node represents any vertex within the graph and a relation is any connection between nodes. A specific kind of node is, of course, a class symbol. An aggregate, associate or specialisation are particular kinds of relationship. Not shown in this class diagram, we might additionally realise a relation composed of any number of line segments.



b) For each class you should propose typical features (attributes and operations) you would expect of such classes, explaining their purpose. **(5 marks)**

Nodes will have geometric properties such as their location on the diagram. Additionally, we might expect nodes to have names (specifically, used for the class name). A class symbol will have a list of features including attributes and operations. A relation might include the role names, multiplicities, etc.

c) **Figure 2** below presents a possible collaboration diagram based on the above class diagram. In it we have three employees that work for an organisation. One of these employees has a managerial role for the other two.

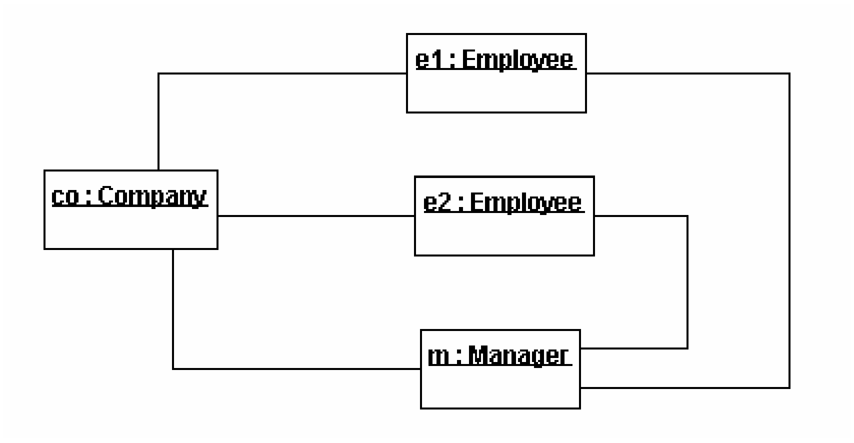
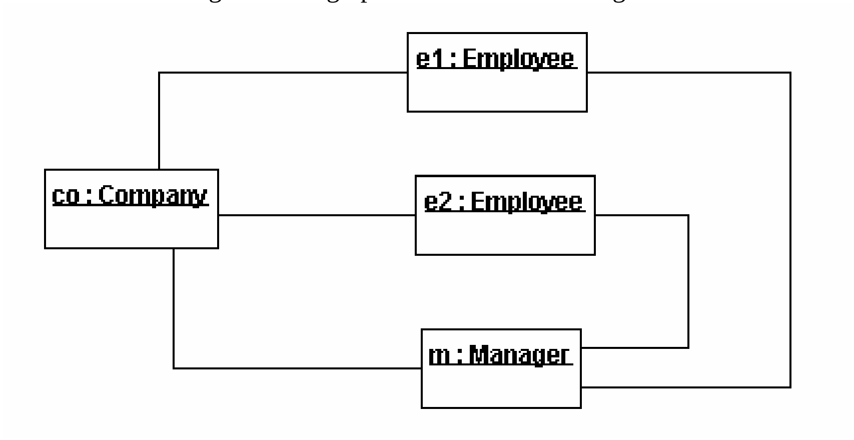
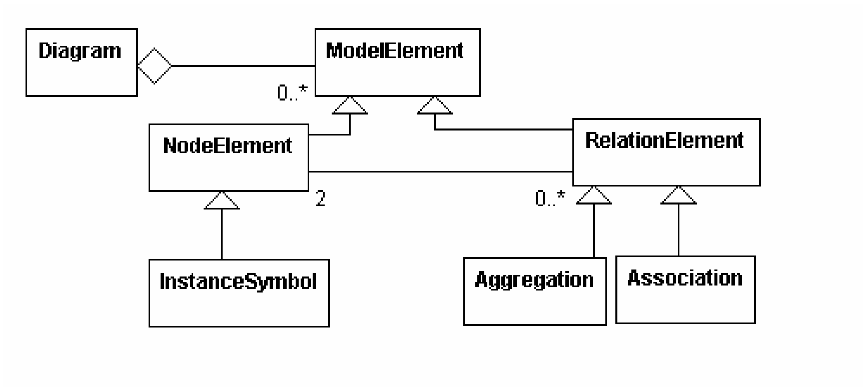


Figure 2

i) Construct a class diagram for a graphical collaboration diagrammer. **(5 marks)**

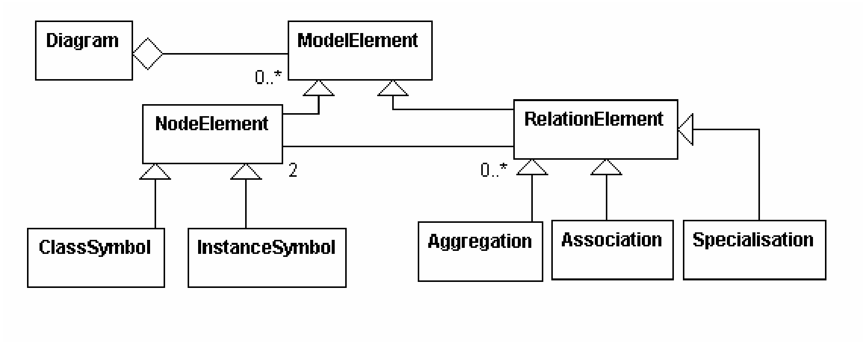


The original class diagram is essentially unchanged. Specialisation is absent from collaboration diagrams and the nodes represent instances. The new figure is:



ii) Unify the class diagrams from *a)* and the diagram above to propose a single class diagram that would describe both types of diagrammer. Identify how such a CASE tool would ensure that only the correct symbols may appear in the correct diagram. **(5 marks)**

The final composite diagram is shown below. The benefit of the one diagram is that it unifies the structure for any diagram type. A diagrammer for a class diagram would restrict the user to creating class symbols and all three relations. In a collaboration diagram we would have only instances, aggregate and association relationships.



Examiner's Guidance Notes
This was not a popular question and was poorly answered. Candidates did not read the question carefully enough. The class diagram given in the question is an example of what might be drawn using a CASE tool. The question clearly asks for the class diagram of a set of classes necessary to construct such a tool. Most candidates attempting this question simply redrew the class diagram and collaboration diagram given as examples.

Question 4

4. *a)* A class is required to hold employee details. The proposed Employee class has the following instance variables:

empno:	String
empname:	String
salary:	Integer

A class variable is also required, called noOfEmployees, which will be incremented each time an Employee instance is created.

Using an object oriented programming language that you are familiar with:

i) Write code to show the declaration of the Employee class, including any 'set' and 'get' methods. **(12 marks)**

ii) Write code to declare two constructors, the first is a default constructor that has no parameters and sets the instance variables to either "not known" for the strings, or 0 for the integer. The second takes 3 parameters, one for each of the instance variables. Both constructors should increment the class variable appropriately. Within your code show how both constructors could be instantiated. **(8 marks)**

Answer Pointers

(i)

```
Class Employee{
        static int noOfEmployees;
        String empNo;
        String empName;
        int salary;
/* setters */
        public void setEmpno(String s){
                empNo = s;
        }
        public void setEmpName(String s){
                empName = s;
        }
        public void setSalary(int i){
                salary = i;
        }
/* getters */
        public String getEmpno(){
                return empNo;
        }
}
        public String getEmpName(){
                return empName;
        }
        public int getSalary(){
                return salary;
        }
}
```

(ii)

```
public Employee() {
        empNo = "not known";
        empName = "not known";
        sal = 0;
        noOfEmployees++;
        }

public Employee (String mEmpNo; String mName; int mSal)
        {
        empNo = mEmpNo;
        empName = mName;
        salary = mSal;
        noOfEmployees++;
        }
```

3 marks for each const.

```
public class EmployeeTest {
  public static void main(String[] args) {
        Employee e1, e2;
        e1 = new Employee();
        e2 = new Employee("1234", "Fred", 21);
        }
```

2 marks

b) Discuss how memory management is typically achieved in an object-oriented programming language.

(5 marks)

Answer Pointers

Memory management is either done by automatic garbage collection by the programming language, such as in Java, or by the programmer as in C. Can be more efficient if done by the programmer, but difficult to do, particularly if inexperienced. Automatic garbage collection requires more processing power and may not always free up all memory that could be deallocated safely.

Examiner's Guidance Notes

The question examined Part 2 of the syllabus “Concepts”.

The candidates made a good attempt at this question on the whole. Java was the most popular language used for the code and most candidates could define the instance variables for the class. What caused some problems was the definition of the *set* and *get* methods. Either candidates omitted them completely, or combined them all into one method. Some answers also failed to include the code for instantiating the objects.

For part b, most answers referred to the automatic garbage collection, provided by languages such as Java, but little else. A few candidates did not answer this part.

Question 5

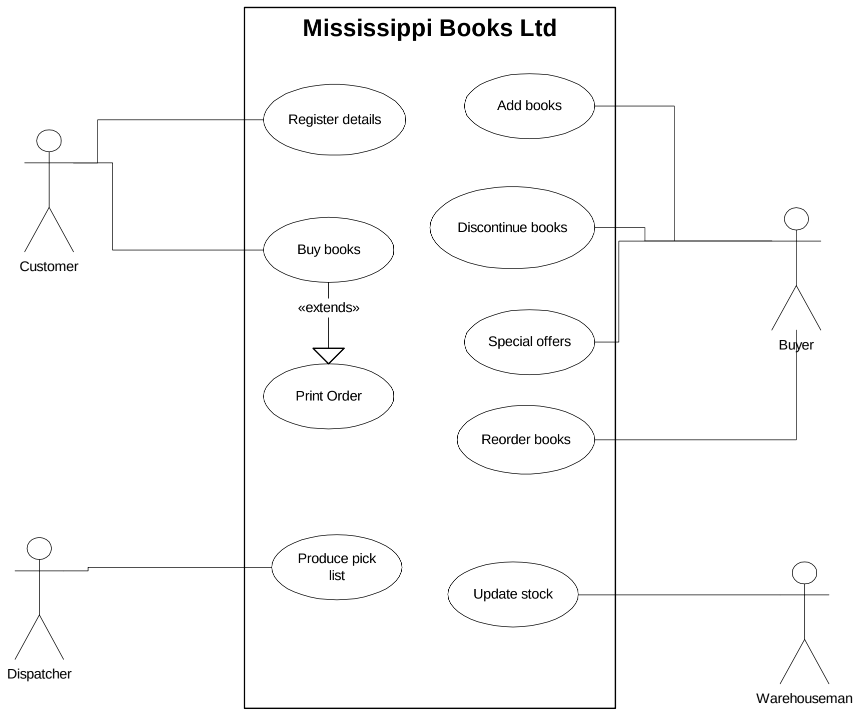
5. Mississippi Books Limited wishes to implement a Web-based system to allow customers to buy their books online. The system also requires facilities to help with their stock control. The outline requirements for the system are as follows:

All customers must be registered with the system. New customers will be presented with a separate form that allows them to add their details. If they fill in all the mandatory details required, and a valid email address, they will be issued with a customer registration number. Registered customers may buy books online. Typically they will browse through details of the books available and select one or more items. If a book is out of stock it will not be displayed whilst a customer is browsing. When they have selected all the items they require they can then move to a checkout screen that enables them to enter their customer number and complete the purchase. If required, the customer can request to print the order details in a printer-friendly version.

Once ordered, the books are sent to a customer by a dispatcher. Every morning the dispatcher requests a list of books that are to be sent out. The list contains the names and addresses of all the customers who have made an order on the previous day and the books they have ordered. A buyer determines what books are for sale. A buyer can add new books to the online catalogue, put existing books on special offer or discontinue the sale of unpopular books. Each morning the buyer requests a list of those books for which the quantity in stock is lower than their re-order level. Books on this list are re-ordered. When new books arrive they are received by a warehouseman who records the new stock levels on the computer system.

a) Draw a use case diagram for this system. **(15 marks)**

Answer Pointers



15 marks

1 per use case

1 per actor

2 for extends

b) Develop a use case description of the way a customer orders books. Your answer should show a normal sequence and also list some alternate sequences. **(10 marks)**

Answer Pointers

The normal sequence:
A customer connects to the web site, browses the catalogue, selects some books, moves to the purchase screen, enters customer number, the customer number is accepted (perhaps with password) and the order is confirmed.

Alternate sequences could be:
- A customer orders a quantity of books greater than the items in stock
- The customer number given is invalid.

Customer prints order

Examiner's Guidance Notes

This question examines section 4 of the syllabus "Practice"

This was a popular question with the candidates. Overall a good attempt was made at the Use Case diagram, with most candidates correctly identifying the actors and the main use cases. Marks were lost if the actors were not named, or an actor was linked to the wrong use case.

In part b, some candidates gave very vague descriptions, or lost marks for not including any alternatives.

Question 6

6. *a)* Describe the following:

i) abstract data type
ii) structured programming
iii) encapsulation
iv) untyped languages
v) typed languages **(15 marks)**

Answer Pointers

i) A user defined type, that defines the attributes and associated operations required and are specified independent of any particular implementation 3 marks
ii) A technique for organizing and coding computer programs in which a hierarchy of modules is used, each having a single entry and a single exit point, and in which control is passed downward through the structure without unconditional branches to higher levels of the structure. 3 marks
iii) Encapsulation hides the internals of the program 3 marks
iv) A programming language where the type of a variable is limited to a certain range of values, e.g., integers, strings, etc. E.g., Java or C++. Typed languages are a way of preventing execution errors. 3 marks
v) A programming language where the type of a variable is not restricted to a range of values, e.g., Smalltalk. 3 marks

b) Choose THREE of the above concepts and discuss how each has contributed to the development of object oriented languages. **(10 marks)**

Answer Pointers

i) Basic points:
Abstract data types allow programmer to associate attributes and methods together, this forms the basis of a class definition, which allows the programmer to define the processing associated with a given data structure.
ii) Encapsulation promotes reuse. Internals of class can be modified but public interface remains the same.
iii) Modular programming allows the programmer to break down the system into a set of sub tasks, which should not interfere with other parts of the program.
iv) Strong typing helps the programmers recognise when they assign data to an inappropriate variable, e.g., try and store a string in an int. Also ensures all potential "message not understood" errors are weeded out at compile time.
v) No checks on types, can lead to errors during run-time, but no (lengthy) compile time checking. Claims that it is good for prototypes but not safety-critical systems.

Object-orientation takes these concepts further and helps the programmer construct reliable software easily

Examiner's Guidance Notes

This question examines Part 1 of the syllabus "Foundations"

Most candidates understood the first three concepts, particularly encapsulation, although some answers showed confusion over abstract data types and abstract classes and a few did not seem to know what structured programming was. Some candidates had no understanding of the differences between typed and untyped languages, particularly the latter.

The first three terms were the most commonly chosen concepts for part b. Strangely, a few candidates who gave the wrong answer for abstract data types in part a, were able to correctly say why they were important in part b.