THE BCS PROFESSIONAL EXAMINATION
Diploma

April 2004

EXAMINERS' REPORT

Object Oriented Programming

Important
This syllabus will be examined for the last time in April 2005 as it is being replaced by the Version 2 syllabus.

Question 1
1. A University's library stores various items that can be borrowed, including books and journals. Both staff and students can borrow books, but only staff members can borrow journals. Students can borrow up to a maximum of 5 books and staff can borrow up to a maximum of 10 books and 3 journals. Books can be borrowed for two weeks and journals one week. If the borrower keeps the book or journal longer than this, they are subjected to a fine, which is increased daily.

When a user borrows a book, they provide their *libraryId*, if this is valid their loan details are checked to ensure that they have not already borrowed above the maximum permitted number of books. They will not be allowed to borrow above the maximum number. 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 user and the return date is assigned to the loan. At this point the user can optionally ask for a printout, which summarises all of the items they have on loan and when each item is due back.

Users can check their own loan details at any time. Librarians are permitted to check the loan details of any user.

Library users can reserve books that are currently out on loan. Journals can not be reserved. If three reservations have already been made for a given book, and a further reservation is made, a new copy will be ordered by the librarian.

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

b) Write down a use case description of the way a user borrows a book. Your answer should include a normal sequence and three alternative sequences. **(10 marks)**

Answer Pointers

This question examines part 4 of the syllabus: Practice

(a) 15 marks

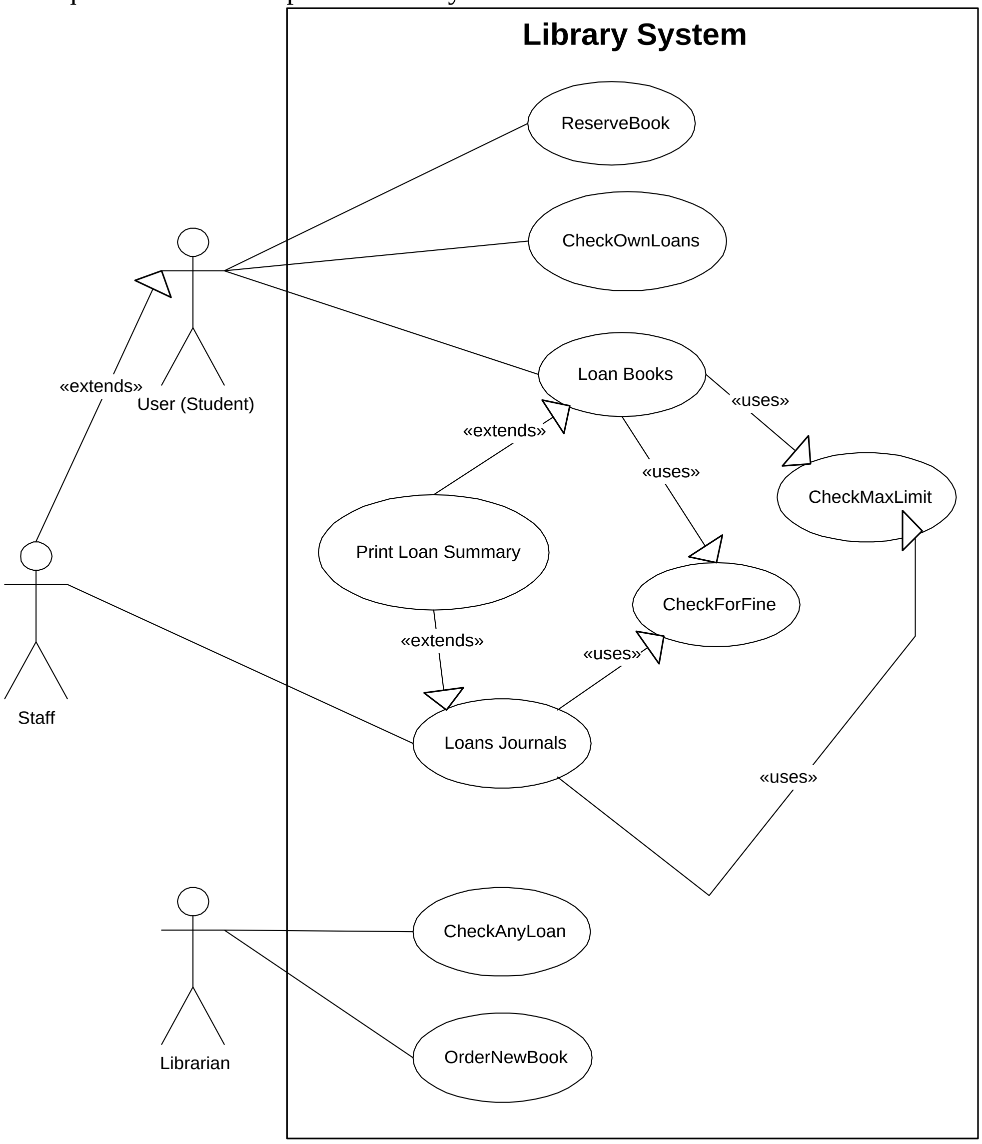
1 per use case

1 per actor

1 for the uses

1 for Print extends

1 for Staff extends



(b) 10 marks

1 per step

Actor Action	System Response
1. User requests to borrow a book and provides libraryId.	2. Check libaryId is valid 3. Check whether staff or student 4. Check user has not exceeded their maximum limit 5. Check user has no fines 6. If ok, book is loaned to user and return date allocated

Alternative sequences

Step 2. Invalid card, loan refused.

Step 4. User has exceeded limit, loan refused.

Step 5. User has fine, loan refused.

Step 7. Request printout of loans

Examiners' Comments

A popular question and on the whole a reasonable attempt was made of part a). Most candidates correctly identified the basic use cases, but not many saw the potential for the *extends* and *uses* relationships. Many candidates also omitted the Librarian actor and few identified that a Staff actor could be a subtype of the general (Student) user.

In some cases the use cases were correctly identified, but the relationships between them were wrongly connected, missing or in some cases, each use case was connected to every other one.

Part b) was generally answered well, a few candidates provided the normal sequence only, or only one sequence rather than three.

2. *a)* Carefully explain the occasions on which you would use the following:

i) a constant instance variable (or field)
ii) a class variable (or field)
iii) a class method (or operation)
iv) a concrete class
v) an abstract class **(10 marks)**

b) A lending library holds a large number of publications that may be books or journals. Both are given a title e.g. "Object-Oriented Programming" and a unique reference number e.g. 123. The reference number is not expected to change. In addition each book also has an author e.g. "John Smith" and an international standard book number e.g. 0-13-12344567-8. Each journal also has a date of publication e.g. 15-12-2003 and the name of its editor. Finally all publications in the library need to hold the name of the library they are in e.g. "City Library".

Using an object-oriented programming language of your choice, provide sample code that demonstrates the use of the five concepts in part *a)* above when implementing the lending library scenario. **(15 marks)**

Answer Pointers

	This question examines Parts 2 and 4 of the syllabus "Concepts" and "Practice"	
(a)	A constant instance variable is used to hold an object's unchanging data. It is given a value once and once only. The compiler will normally prevent changes.	2 marks
	A class variable is used to hold data common to all instances of the class. Only one class variable is created per class regardless of the number of objects of the class that are created.	2 marks
	A class operation is used to manipulate a class variable. It is illegal for a normal operation to do so. The class operation is normally invoked through the class name. Class operations cannot access instances variables as they cannot be assumed to exist.	2 marks
	A concrete class is used to represent objects that correspond to real objects. It is the default for a class.	2 marks
	An abstract class is used to represent the common protocol (and possibly common implementation) of subclasses. It is expected to have abstract methods. It is illegal to create an object of this kind of class. However we may declare a reference to it.	2 marks

(b)

```
abstract public class Publication {
   private final int theReferenceNumber;
   private String  theTitle;
   //
   private static String theLibraryName;
   public static int getLibraryName() { return theLibraryName; }
   // …
}

public class Book extends Publication {
   private String theAuthor;
   // …
}

public class Book extends Publication {
   private String theDateOfPublication;
   private String theEditor;
   // …
}
```

Code fragments such as:

```
   Publication  publication = new Book(…);
   Publication.getLibraryName();
```

15 marks

3 marks for each concept illustrated

Examiners' Comments

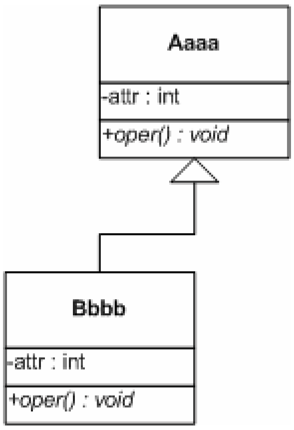
Generally a lot of candidates only attempted part a) and not part b). Out of part a, the candidates mostly answered parts i)-iii) correctly, but gave hazy definitions to parts iv) and v), some mixing up abstract data types with abstract classes.

Few attempted part b), out of those that did, again not many could distinguish between abstract and concrete classes; some had problems defining a class variable too. Java, and C++ were the most common languages used, with a few instances of VB.net.

Question 3

3. *a)* Describe in detail **TWO** design patterns with which you are familiar. Your answer should include the circumstances in which they are applicable, when they can be applied and trade-offs when using them within a larger design. **(16 marks)**

b) The following class diagram includes the classes Aaaa and Bbbb in which the latter is a specialisation of the former. Both classes include a definition for the method oper and the attribute attr. Identify any issues arising from this arrangement and consider their implications. **(9 marks)**



Answer Pointers

This question examines Part 4 of the syllabus "Practice"

(a) **Singleton**

Intent: ensure a class has only one instance and provide a common global point of access to it. — 2 marks

Applicability: use the singleton pattern when there must be exactly one instance of a class, and it must be accessible to clients from a well-defined point. — 2 marks

Consequences: controlled access to sole instance; avoid global variable name space pollution; subclassing to configure the particular instance required; can be extended to permit a set number of instances. — 4 marks

Observer

Intent: define a one-to-many dependency between objects so that when one object changes state, all its dependencies are notified and updated automatically. — 2 marks

Applicability: when a change to one object requires changes on others; and when an object should be able to notify other objects without making assumptions about who they are. — 2 marks

Consequences: loose coupling between the subject and its observers; a subject simply has a list of observers that conform to an interface; support for broadcast communication. — 4 marks

(b) The class diagram reveals that the subclass Bbbb has provided a redefinition for the operation oper. This is in keeping with the designer seeking to deploy dynamic binding of this method and — 4 marks is typical of specialisation architectures.

5 marks

The re-introduction of the attribute attr in the subclass Bbbb might be considered an error. If permitted, this would mean that an instance of Bbbb would have two attr attributes as part of its state. Further, methods in the subclass can reference that attr defined by Bbbb but not that defined in Aaaa.

Examiners Comments

The Singleton and Observer patterns were the most often described patterns. Some candidates only knew one design pattern well. Many could describe the pattern, but were weak at saying where they applicable, when they could be applied and the trade-offs.

Few candidates attempted part b) of this question. Those that did mostly did not see the consequences of redefining the attribute and method in the subclass.

Question 4

4. *a*) Explain the following terms:

- i) Object;
- ii) Class;
- iii) Inheritance;
- iv) Superclass;
- v) Subclass.

(10 marks)

b) Object oriented programming languages implement *inheritance*, some languages however implement *single inheritance* whilst others implement *multiple inheritance*. Distinguish between these types of inheritance and discuss why a language designer might choose to implement one but not the other.

(5 marks)

c) A bank operates accounts. The basic operations on accounts are deposit() and withdraw(). The bank operates a number of types of accounts amongst which are savings accounts and current accounts. Savings accounts have no overdraft facility associated with them. Current accounts have a limited overdraft facility. Using an object-oriented language with which you are familiar develop code which shows how inheritance and polymorphism can be used to model the bank.

(10 marks)

Answer Pointers

	This question examines Part 2 of the syllabus: Concepts
(a)	i) a self contained entity which brings together data and the procedures which operate on that data; ii) defines the common properties of an object i.e. all objects with the same data and same methods belong in the same class; iii) inheritance is when one class is based on a another class iv) a superclass is a class which provides the basis for inheritance v) a subclass is the class which is based on a superclass and builds upon it.
(b)	In single inheritance a class may only inherit from one superclass. In multiple inheritance classes may inherit from one or more superclasses. There are conceptual difficulties associated with multiple inheritance. This is perhaps best illustrated by considering the naming problem. If class A inherits from superclasses X and Y and both X and Y contain a method with the same name which version of the method does A inherit? This issue is resolvable but its resolution may not be easy for a programmer to master.

Examiners' Guidance Notes

Part a) which asked for explanations was generally answered well. In part b) most candidates were able to distinguish between the two types of inheritance but many were unable to state any problems that might be encountered in a language that implemented multiple inheritance. Part c) which required the students to write code illustrating the use of inheritance only produced a few good solutions. On the whole candidates did not demonstrate an extensive experience of programming in an object-oriented language.

Question 5

5. *a*) Give the meaning of the following terms:

- i) Abstraction;
- ii) Encapsulation;
- iii) Data hiding.

(9 marks)

b) A programmer wishes to create a set of classes that implement collections (e.g. Set, SortedSet, List, SortedList). Explain how abstraction, encapsulation and data hiding can be used to create generic classes for this purpose.

(10 marks)

c) Describe the contribution that abstraction, encapsulation and data hiding make to the potential of a language to encourage software reuse.

(6 marks)

Answer Pointers

	This question examines Part 3 of the syllabus: Principles	
(a)	i) the identification of common features and operations; ii) the process of combining elements to form an new entity; iii) a mechanism for hiding the details of the implementation of an object from the code that uses it.	9 marks
(b)	A process of abstraction should tell us that all collection classes do essentially the same task. In particular every collection class will be able to add to the collection and remove items from the collection. Via polymorphism the nature of these operations may however be different so adding to a Set will be implemented in a different way from adding to a List (duplicates are allowed in lists but not in sets). Data hiding will conceal these differences from us so that we needn't worry about them but just be aware that adding may in some cases fail. Encapsulation will allow us to package different parts of an entity together as an object. Collection classes which will operate on objects can be used in any context so that it is not necessary to implement a specialist class for each different type of object we wish to collect together.	10 marks
(c)	By identifying the most general cases and hiding the details of specific implementations from the code using it, it is possible to develop set of classes which operate on all objects rather than just objects belonging to specific classes. Such classes are easily reusable.	6 marks

Examiners Guidance Notes

This question was not answered very well with many of the candidates unable to demonstrate that they had appreciated the principle of abstraction. Parts b) and c) were consequently also poorly answered as they required candidates to discuss instances where abstraction yields general solutions to a problem. Many candidates confused abstraction with the use of the abstract keyword in Java and submitted description of the syntax for defining abstract classes.

Question 6

6. *a)* Give a simple example of each of the following diagrams and describe the context in which you would use them.

i) Use case diagram;
ii) Object interaction diagram (sequence diagram or a collaboration diagram);
iii) State transition diagram.

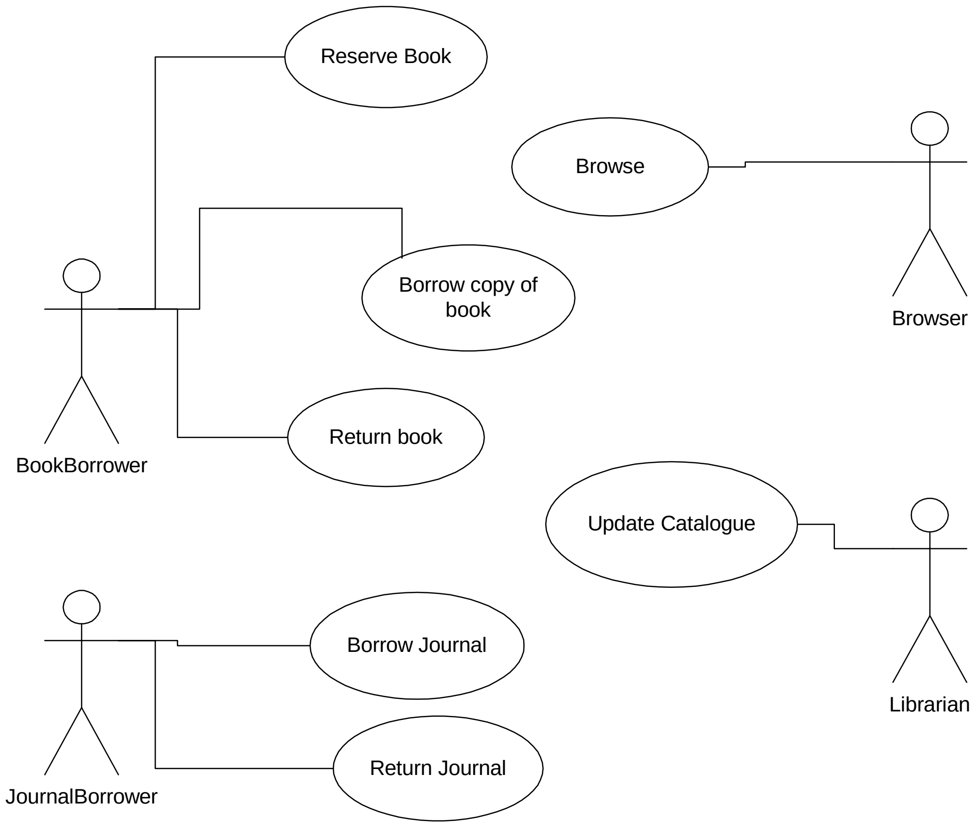
(15 marks)

b) Describe an approach you might follow to derive test cases for a software product from the use case diagrams that specify it.

(10 marks)

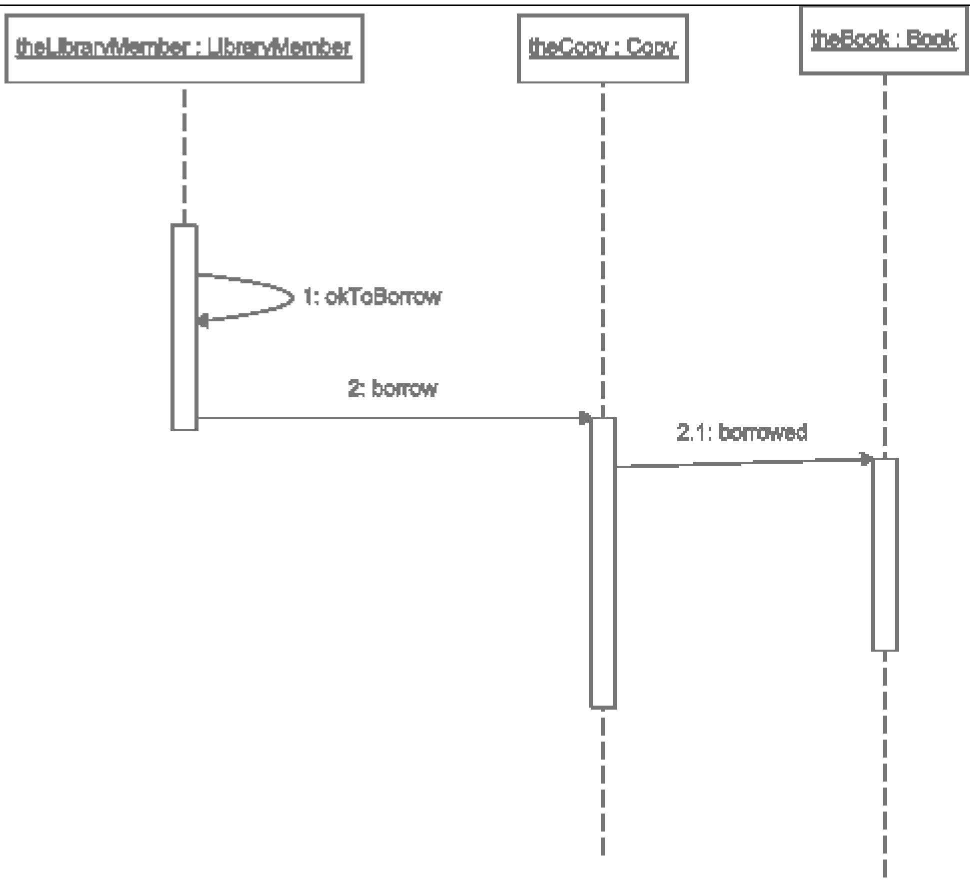
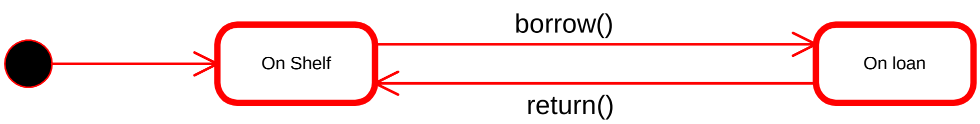
Answer Pointers

<table>
<tr><td></td><td>This question examines Part 4 of the syllabus: Practice</td></tr>
<tr><td>(a)</td><td>Use Case Diagram identifies the users of a system and the tasks they undertake.



Object interaction diagrams show how objects interact to perform a task. This shows how a system realises a Use Case. The diagram below is a sequence diagram, a collaboration diagram showing interaction is also acceptable.

State diagrams show the dependencies between actions on an object and the state it is in.</td></tr>
</table>

	 
(b)	For each Use Case on the Use Case Diagram develop a set of scenarios. Scenarios are natural language descriptions of an instantiation of the Use Case. Choose the scenario which reflects the way in which the system is expected to behave. So for example if the Use Case is "borrow a book", select the scenario where the book is available, the borrower has not exceed their limit etc. For each step in this scenario consider all the exceptions that can happen. Exceptions are events which cause an alternative scenario to be followed. For each exception state what the system should do if that exception occurs. When the software is built test the normal scenario and all the exceptions.

Examiners' Comments

The performance on question six was on the whole quite good. The answers for part a) that asked the candidates to demonstrate UML diagramming techniques were answered better than part b) that required a description of the generation of test cases given a Use Case diagram.