THE BCS PROFESSIONAL EXAMINATION
Diploma

April 2004

EXAMINERS' REPORT

Object oriented Programming (Version 2)

Question 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 cannot 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)

Write down a use-case description of the way a user borrows a book. Your answer should include the normal sequence and three alternative sequences such as when invalid data is provided. (10 marks)

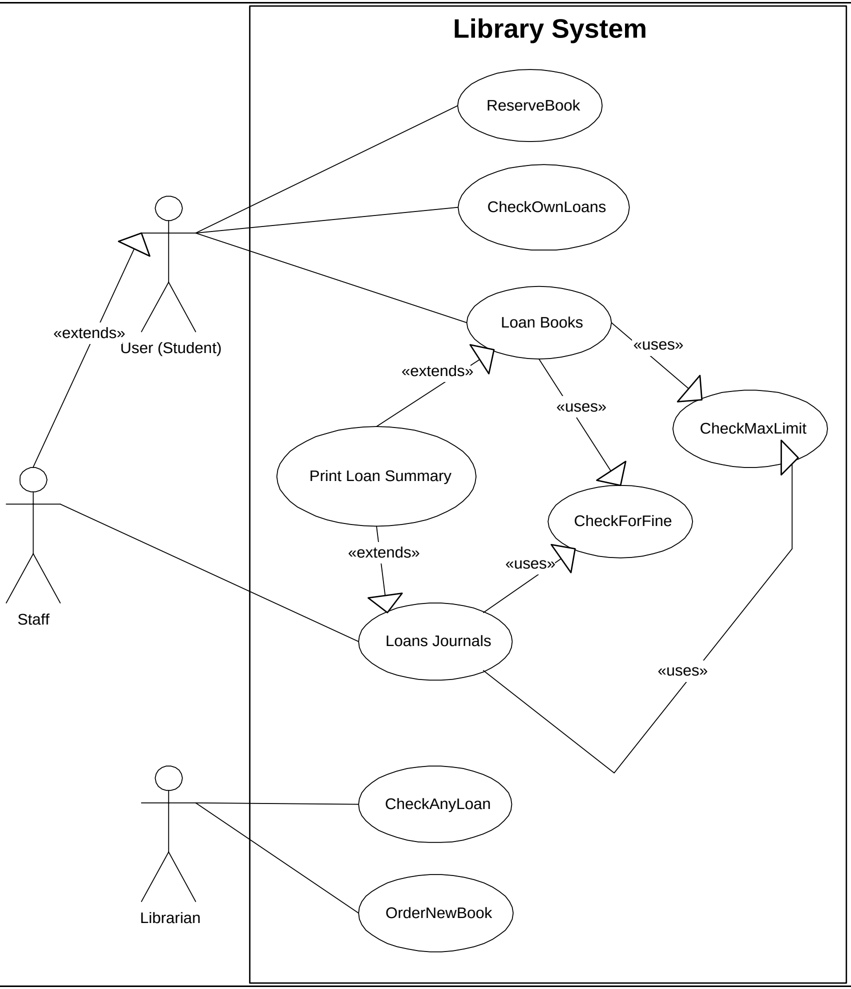
Examiner's Comments

This question was a popular choice. Perhaps it was too easy?

Overall this part was answered well. However very few students were aware of the possibility of extending an actor. This is a pity, as it would have simplified the use case diagram significantly.

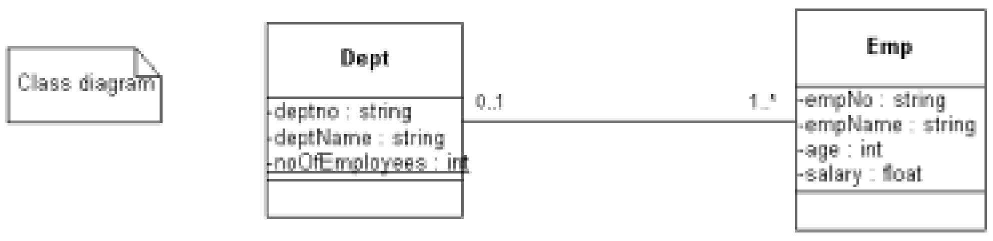
The format of many of the answers was arbitrary. Some consisted of just a narrative. Perhaps we should have asked for a discussion of the standard format used.

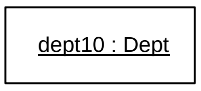
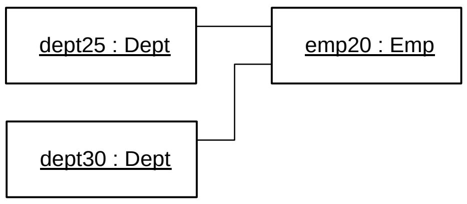
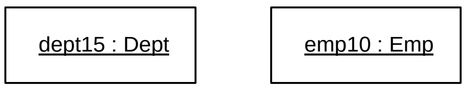
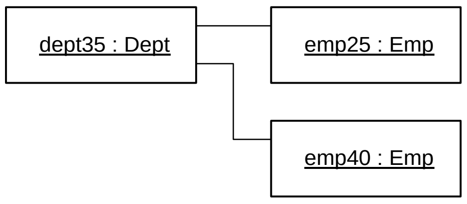
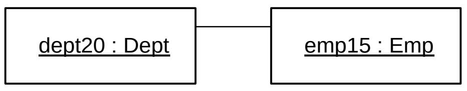
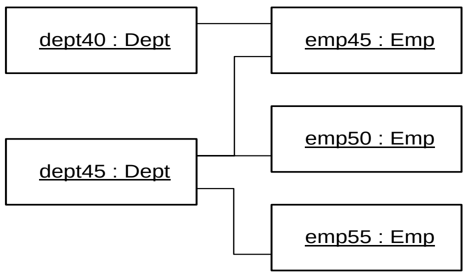
Question		Mark
1	This question examines part 4 of the syllabus: Practice	

(a)	 Library System ReserveBook CheckOwnLoans User (Student) «extends» Loan Books «uses» «extends» «uses» CheckMaxLimit Print Loan Summary CheckForFine «extends» «uses» Staff Loans Journals «uses» CheckAnyLoan OrderNewBook Librarian	15 1 per use case 1 per actor 1 for the uses 1 for Print extends 1 for Staff extends
(b)	Actor Action System Response User requests to borrow a book 2. Check libaryId is valid and provides libraryId. 3. Check whether staff or student 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	10 1 per step

Question 2

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)



ii) i)		iv) iv	
ii ii		v) v)	
ii iii)		vi) vi)	 dept40 : Dept emp45 : Emp emp50 : Emp dept45 : Dept emp55 : Emp

In a programming language with which you are familiar, write code to implement the class diagram above. Within your code provide a default constructor for each class that sets strings to "Not known" and numeric fields to 0. Your code should also show how the association relationship is implemented in both classes. (15 marks)

Examiner’s Comments

Overall it was a good question that elicited good answers from students.

This question also achieved its aim. However very few students managed to implement the association relationship between classes.

2	This question examines part 4 of the syllabus: Practice	
(a)	Invalid because dept must have at least one employee. Invalid because dept must have at least one employee. Ok for employee not to be associated with a dept. Ok Invalid, because an employee only works for one dept Ok Invalid, because an employee only works for one dept	10 (2 each for i, ii, iv and vi. 1 each for iii and v)

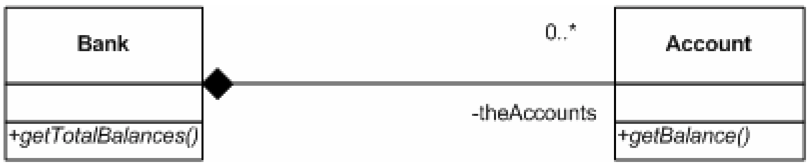
(b)	class Dept{
	        static int noOfEmployees = 0;
	        String deptNo;
	        String deptName;
	      Set employees;
	
	        public Dept(){
	                deptNo = "Not known";
	                deptName = "Not known";
	          employees = new Set();
	        }
	}
	class Emp {
	        String empNo;
	        String empName;
	        int age;
	        float salary;
	        Dept workIn;
	        // or could implement as a Set from Dept
	        public Emp() {
	                empNo = "Not known";
	                empName = "Not known";
	                salary = 0.0f;
	                age = 0;
	                }
	}
	}
		15
		
		7 marks for Dept 8 marks for Emp
		
		1 for correct class and construc-tor.
		
		1 for each attr & defn.

Question 3

Explain and show with an example how a sequence diagram is used to describe the interactions between objects. Show also how the same interaction could be described by a corresponding collaboration diagram. Identify the significant differences between collaboration and sequence diagrams and their individual strengths. (6 marks)

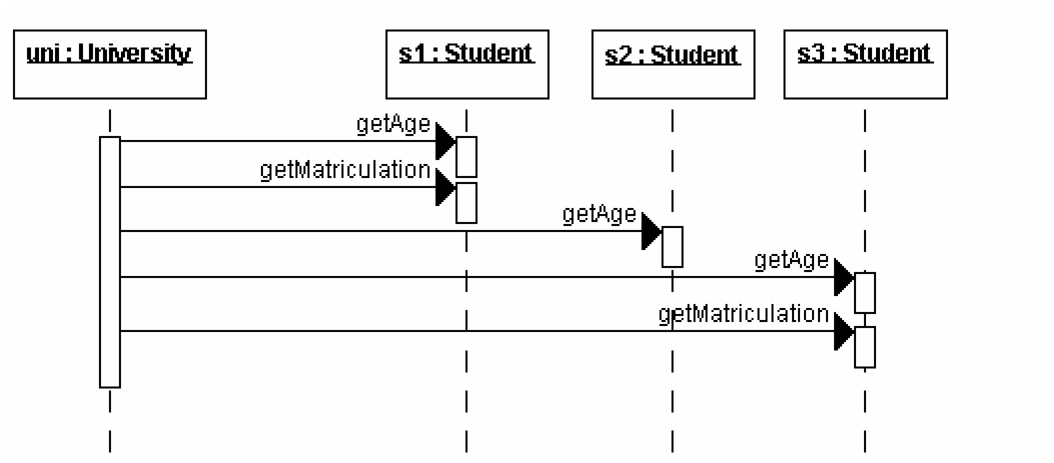
In the following class diagram a Bank is shown with a one-to-many relation with the Account class. The method getBalance delivers the balance for an Account object, while the method getTotalBalances determines the sum of the balances for the Account objects associated with a Bank object.

Present a collaboration diagram for the realisation of the method getTotalBalances. (4 marks)



Show the same interaction presented as a sequence diagram. (4 marks)

In the following sequence diagram a University object is shown sending the message getAge to each of its three Student objects, and for those that are over the age of 21, they are also sent the message to obtain their matriculation number. Present a class diagram showing its attributes and methods that models this scenario.



(8 marks)

A collaboration diagram that does not show any messages is known as an *object diagram*. Explain its purpose. (3 marks)

Examiner's Comments

This was a popular question and generally well answered. Two observable weaknesses were identified. In part (b) many candidates showed too few object instances to reveal the true nature of the interactions. Commonly I found one Bank and one Account were used in the illustration, whereas I might have expected say one Bank and three Account objects. This would have better shown the message flows.

The second problem that was repeated by many candidates was the answers given to part (c). Here, some candidates gave an object instance diagram rather than a class diagram. Second, those offering a class diagram introduced unnecessary complexity. The Student class was expected to have a method to obtain the age of a student. Many solutions then sub-classed Student and redefined the method for those over 21!

The question seeks to demonstrate an understanding of design patterns, their motivation, applicability and consequences.

This answer should consider whether re-introducing an attribute in a subclass with the same name as one already in the superclass is an error. The answer should also recognise the redefinition of the operation oper as normal practice in a specialisation hierarchy.

Question		Mark
3	This question examines Parts 3 and 4 of the syllabus "Design" and "Practice"	
(a)	Singleton Intent: ensure a class has only one instance and provide a common global point of access to it. 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. 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. 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. 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. 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.	2 2 4 2 2 4
(b)	The class diagram reveals that the subclass Bbbb has provided a redefinition for the operation oper. This is in keeping with the	4

	designer seeking to deploy dynamic binding of this method and is typical of specialisation architectures. 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.	5

Question 4

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)

Examiner's Comments
Again, a popular question. Part 4(a)(i) and 4(a)(ii) saw candidates thinking that a class variable and a class method were variables and methods defined in class! In part (b) I was expecting outline code fragments that illustrated the items in 4(a)(i)…(iv). Many candidates offered lots of code but without answering directly the question.

(a) This question seeks to reveal the understanding by the candidates of when it is most appropriate to use these common object-oriented programming artefacts. They need to offer an explanation of what each is normally used for and why.

(b) The following code fragments (Java) illustrate outline coding for the publications the library holds. Each publication may belong to the concrete Book or Journal classes. Both are subclasses of the abstract Publication class. There is also a class variable to represent the name of the library and class operation to accesses.

The Publication class has a constant instance variable to represent the reference number and another normal variable for its title. It also has a class variable for the name of the library all publications hold as well as a class operation to access it. The Book subclass introduces a normal instance variable for its author and another for its isbn number. The Journal subclass introduces one instance variable for its date of publication and another for its editor.

Question		Mark
4	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. 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. 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. A concrete class is used to represent objects that correspond to real objects. It is the default for a class. 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 2 2 2 2

(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();`
	
	3 marks for each concept illustrated
		15

Question 5

a) Give brief definitions of the following:

i) coupling;
ii) cohesion;
iii) information hiding;
iv) abstraction. (8 marks)

b) Explain how abstraction is facilitated by a typical object-oriented programming language. (7 marks)

c) Discuss ways in which object-oriented techniques can influence the level of coupling and cohesion. (10 marks)

Examiner's Comments
Some candidates confused the general notion of abstraction with abstract classes. This occurred in both parts 5(a) (iv) and 5(b).

(a) This question seeks to demonstrate an understanding of these fundamental concepts. Students are expected to offer a definition and explanation of each. A small illustration is also expected.

(b) A typical answer should revolve around the manner in which the language supports the class (possibly the abstract class and/or interface), class inheritance, visibility/redefinition of operations and their implementation.

(c) The answer is expected to focus on the interconnections between objects and their purpose. Good students may state that a common design goal is to have low coupling and high cohesion.

Question		Mark
5	This question examines Part 1 of the syllabus "Foundations"	
(a)	i) The extent to which a method depends on its environment i.e. other methods, classes and objects. ii) The level of uniformity of the method goal. iii) Ensuring that only the information that required by a client is made available. If it is not then it is normally made private or local. iv) The ability to group large quantities of information together in a single entity. It may be viewed in different ways and at different levels of detail depending on circumstances.	2 2 2 2
(b)	A typical answer should revolve around the manner in which the language supports the class (possibly the abstract class and/or interface), class inheritance, visibility/redefinition of operations and their implementation.	7
(c)	In an OOP encapsulation is used to associate data with operation on that data. Normally this involves data hiding i.e. variables are not accessible to methods not located in the same class. This makes classes stand-alone entities and so	

	decreases coupling. Methods perform simple operations on the data in the class. They normally have a single purpose and so increase cohesion. OOP often leads to low coupling and high cohesion.	10

Question 6

a) Explain what is meant by the term *delegation*. (5 marks)

b) A small software company requires that its employees are flexible in the tasks that they are able to perform. For example, an employee may undertake an administrative role in the morning but be a programmer in the afternoon. However it should be possible to determine at any given time what an employee is doing. You are asked to develop software to display on a computer screen the role of each employee in the company at any given time.

i) Explain why inheritance alone is not appropriate when modelling the company and its employees. (5 marks)

ii) Give an initial class diagram for the company and its employees demonstrating the use of delegation. (10 marks)

iii) Using an object-oriented language of your choice outline how your design might be implemented. (5 marks)

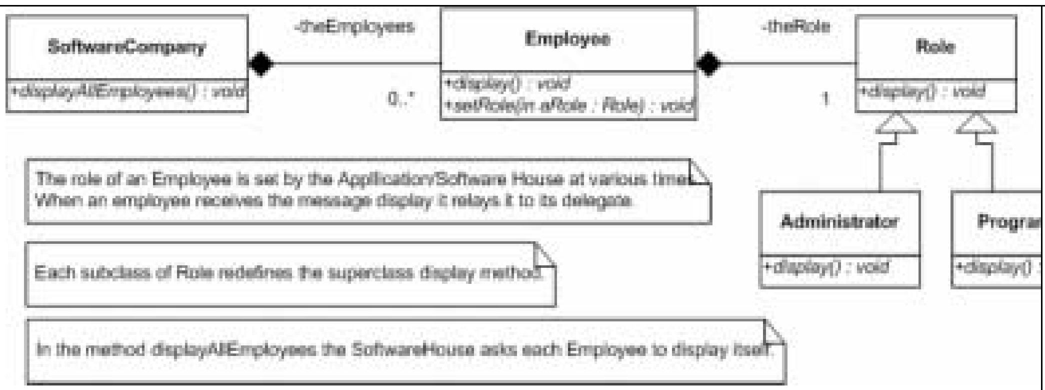
Examiner's Comments

This proved not to be a popular question. It may have appeared too difficult and as a result very few students attempted it.

(a) This question seeks to demonstrate that specialisation is normally a compile-time (static) phenomenon while delegation can be accomplished at run-time (dynamic). Specialisation applies to all objects of the class but delegation may be applied to individual objects.

(b) This question seeks to demonstrate delegation in action with a concrete example.

Question		Mark
6	This question examines Parts 2 , 3 and 4 of the syllabus "Concepts", "Design" and "Practice"	
(a)	Delegation is a technique that can be used to alter the behaviour of an object over time. In delegation an object passes on the responsibility for implementing a method to a delegate. The choice of delegate can be made at runtime and therefore behaviour can be varied dynamically.	5
(b)	i) Each employee adopts different roles at different times. Therefore class scope is not appropriate. We need the run-time flexibility and object scope that delegation offers.	5


SoftwareCompany
+displayAllEmployees() : void
-theEmployees
0..*
Employee
+display() : void
+setRole(in aRole : Role) : void
-theRole
1
Role
+display() : void
Administrator
+display() : void
Progra
+display()
The role of an Employee is set by the Application/Software House at various times. When an employee receives the message display it relays it to its delegate.
Each subclass of Role redefines the superclass display method.
In the method displayAllEmployees the SoftwareHouse asks each Employee to display itself.

10

ii)
4 marks for the architecture
2 marks for the operations
4 marks for the explanation

iii)

```
public class SoftwareCompany  {
    public final void    displayAllEmployees() {
        // Request each Employee to display itself
    }
    private Employee    theEmployees;
    // …
}

public class Employee  {
    public final void    display() {
        // A suitable display of an Employee
              theRole.display();
    }
    public final void    setRole(Role aRole) {
        theRole = aRole;
    }
    private Role    theRole;
    // …
}

public class Role  {
    public void    display() {
        // A suitable display
    }
    // …
}

public class Programmer extends Role  {
    public void    display() {
        // A suitable display
    }
    // …
}

public class Administrator extends Role  {
    public void    display() {
```

5

	```
	        // A suitable display
	    }
	    // …
	}
	```
	2 marks for showing delegation in the Employee display method
	2 marks for setting the delegate
	1 mark for the overall implementation.	