THE BCS PROFESSIONAL EXAMINATION
Diploma

April 2003

EXAMINERS' REPORT

Object Oriented Programming

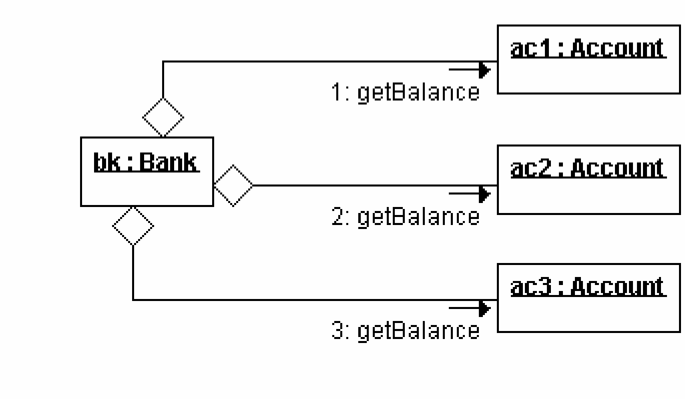
Question 1

a) Collaboration diagrams and sequence diagrams are described as interaction diagrams. Describe what are shown by these two types of diagrams. Identify the individual strengths of collaboration and sequence diagrams.
(6 marks)

Answer Pointers
Interaction diagrams are used to show the flow of messages through a collection of objects. The propagation of these messages is intended to illustrate how some functionality is achieved. A sequence diagram conveys this with a timeline, while a collaboration diagram uses numbered messages. Additionally, the collaboration diagram can reveal the configuration of the objects.

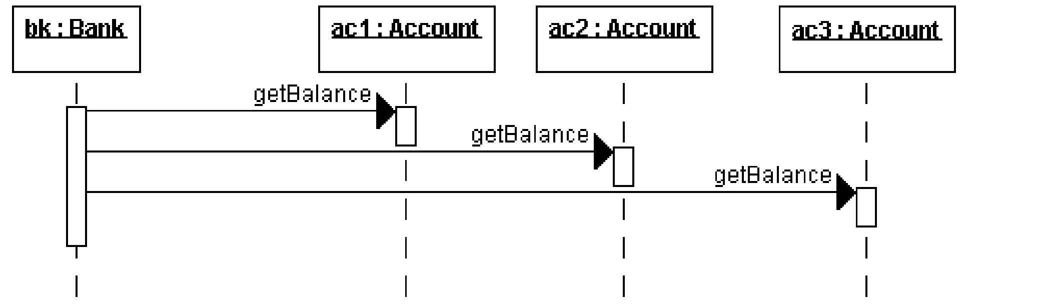
Question		Mark
1	This question examines Part 4 of the syllabus "Methods"	
(a)	Interaction diagrams are used to show the flow of messages through a collection of objects. The propagation of these messages is intended to illustrate how some functionality is achieved. A sequence diagram conveys this with a timeline. A collaboration diagram uses numbered messages. Additionally, the collaboration diagram can reveal the configuration of the objects.	2 2 2

b) In the following collaboration diagram a Bank object is shown sending the message getBalance to each of its three Account objects to obtain a total of the balances. Show the same interaction presented as a sequence diagram.

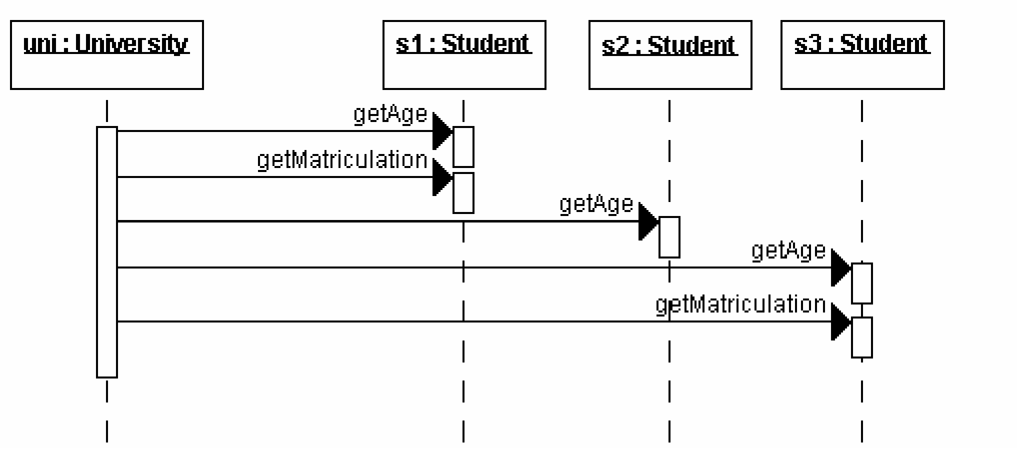


(8 marks)

In parts b) and c) a simple reversal of the diagrams is required. Part b) requires a sequence diagram not unlike that shown in part c); while part c) needs a collaboration diagram as used in part b). Both will need to be presented accurately, showing the objects, the order of messages and, in the case of a collaboration diagram, the configuration of objects.

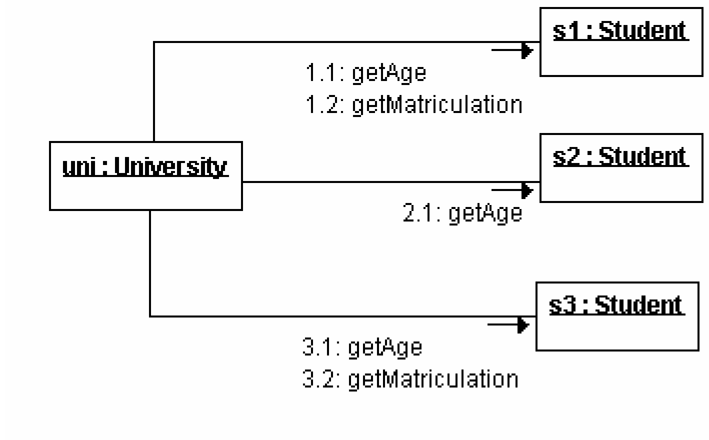
Question		Mark
1	This question examines Part 4 of the syllabus “Methods”	
(b)	 bk : Bank ac1 : Account ac2 : Account ac3 : Account getBalance getBalance getBalance 1 mark for each object instance (4) 1 mark for each message (3) 1 mark for a brief explanation (1)	8

c) 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 the same interaction using a collaboration diagram.



(8 marks)

In parts b) and c) a simple reversal of the diagrams is required. Part b) requires a sequence diagram not unlike that shown in part c); while part c) needs a collaboration diagram as used in part b). Both will need to be presented accurately, showing the objects, the order of messages and, in the case of a collaboration diagram, the configuration of objects.

Question		Mark
1	This question examines Part 4 of the syllabus “Methods”	
c)	 s1 : Student 1.1: getAge 1.2: getMatriculation uni : University s2 : Student 2.1: getAge s3 : Student 3.1: getAge 3.2: getMatriculation 1 mark for each object (4) 1 mark for each group of messages (3) 1 mark for short explanation (1)	8

d) What is a collaboration diagram without any messages known as, and explain its purpose.

(3 marks)

Answer Pointers

A collaboration diagram without any messages is known as an object diagram. The purpose of an object diagram is to reveal the architecture of the interacting objects.

Question		Mark
1	This question examines Part 4 of the syllabus “Methods”	
(d)	A collaboration diagram without any messages is known as an object diagram. The purpose of an object diagram is to reveal the architecture of the interacting objects.	3

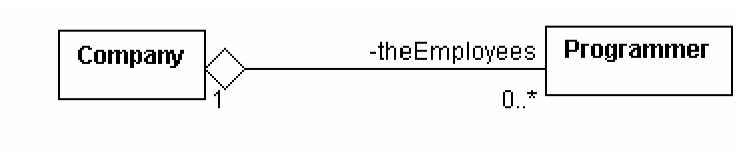
[Total 25 marks]

Examiner’s Guidance Notes

This question was generally well answered, especially parts b) and c) which were of a practical nature.

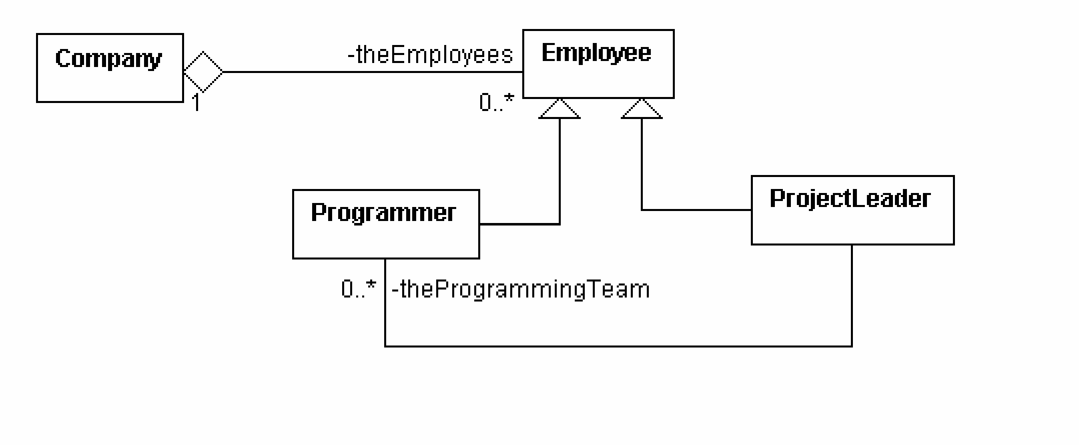
Much poorer responses were received for the descriptive parts a) and d). In part a) candidates should have revealed the strengths of the two diagrams which otherwise convey the same knowledge. Surprisingly, many candidates answered part d) with a class diagram when a completely different (object) diagram was required.

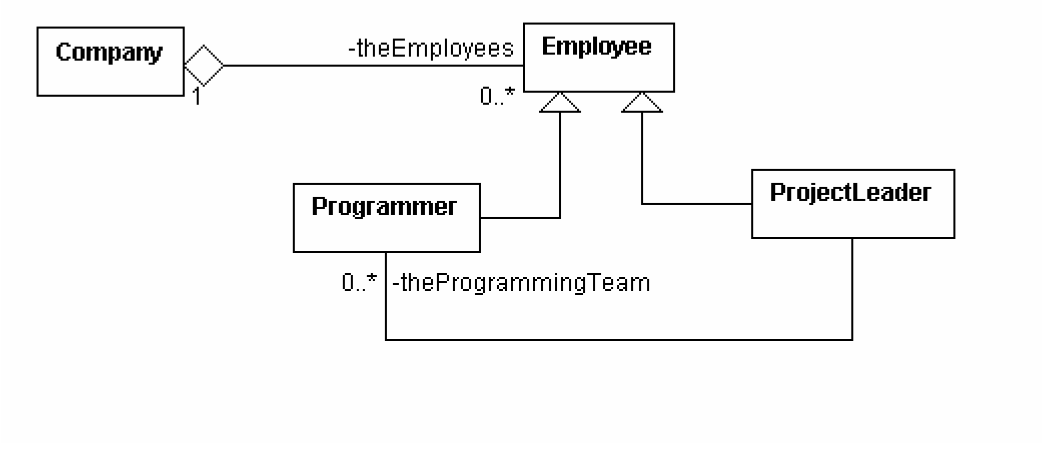
Question 2
The following class diagram presents an organisation with any number of programmer employees.



a) Provide a revised class diagram in which the organisation has both programmers and project leaders as employees. Each project leader supervises any number of programmers. Give an argument for the revisions that you have introduced.
(7 marks)

Answer Pointers
The solution is achieved through specialising an Employee class with two concrete subclasses Programmer and ProjectLeader. The latter has a one-to-many relationship with the programmers in his charge.



Question		Mark
2	This question examines Part 4 of the syllabus “Implementation”	
a)	 Company -theEmployees Employee 1 0..* Programmer ProjectLeader 0..* -theProgrammingTeam 2 marks for the classes 2 marks for the relationships and, where appropriate, role & multiplicity 3 marks for the rationale	7

b) Both the original class diagram and the revised class diagram have one-to-many relationships. Explain how they might be realised in application code.
(5 marks)

Answer Pointers
The answer needs to demonstrate a knowledge of the various containers (collections) and how they are used in handling one-to-many relations that occur frequently in OO designs.

Generally such relationships are implemented using containers such as the C++ STL library or the Java collection classes. These containers are flexible storage classes that can grow to meet the application needs. Further, they are generic and can be used to store any type of object. The containers are also supported by iterators so that code can be written to traverse the collection and access the members.

Question		Mark
2	This question examines Part 4 of the syllabus "Implementation"	
b)	Generally such relationships are implemented using containers such as the C++ STL library or the Java collection classes. These containers are flexible storage classes that can grow to meet the application needs. Further, they are generic and can be used to store any type of object. The containers are also supported by iterators so that code can be written to traverse the collection and access the members.	3 2

c) If the employees were to be arranged in order of employee number, identify what changes would be required in the code. Explain what coding practices might be used to minimise the impact of these changes. What would be the effect on the class that represents the employees? (7 marks)

Answer Pointers
c) Here, we would choose some form of sorted container. The collection objects would normally have some method by which elements can be compared for their sort order. We can reduce the code changes by using iterators and by referring to the collections by an abstract superclass name.

Question		Mark
2	This question examines Part 4 of the syllabus "Implementation"	
c)	Here, we would choose some form of sorted container. The collection objects would normally have some method by which elements can be compared for their sort order. We can reduce the code changes by using iterators and by referring to the collections by an abstract superclass name.	4 3

d) If the application was required to look up the details for an employee using their employee number as a key, describe the revisions that would be necessary to correctly handle this one-to-many relationship. (6 marks)

Answer Pointers
Here, we would choose some form of Map container. The collection objects are arranged as key/value pairs. Here the key would be the employee number while the value would be the employee object itself.

Question		Mark
2	This question examines Part 4 of the syllabus “Implementation”	
d)	Here, we would choose some form of Map container. The collection objects are arranged as key/value pairs.	4
	Here the key would be the employee number while the value would be the employee object itself.	2

[Total 25 marks]

Examiner’s Guidance Notes
There was some indication that this was the last question attempted by candidates. As a result, the outcome was somewhat disappointing with the question poorly answered.

The question was primarily concerned with the usage and value of collections, which I am sure the candidates will have had actual practical experience. Yet this was not evident in the responses, suggesting the candidates were not reflecting on this experience.

I was also surprised by the number of candidates attempting to answer part (a) of this question without recourse to specialisation. There is no other OO way to deal with this but many (strange) attempts were made!

Question 3

a) Describe the usage of the following use case UML extension stereotypes:
i) <<uses>>
ii)<<extends>> **(6 marks)**

b) A local government authority implements a discussion forum where citizens may post messages in order to discuss local issues.

Citizens may post and read messages. In order to ensure that only people from the area under the control of the authority post messages, every user must be registered. When a citizen reads or posts a message the citizen’s identity must be authenticated. This is achieved via a user name and password mechanism.

Messages are grouped together depending on their topic. Before posting or reading a message, a citizen must select the group they are interested in.

The authority has a policy that prohibits the posting of abusive messages. To enforce this, moderators review the groups and delete any messages which they decide contravene the authority’s guidelines. Deletion of a message is very similar to reading a message. A moderator will read a message and then click on a deletion button. Moderators must be authenticated by the system in the same way as citizens.

User accounts are created by an administrator. Administrators also create groups. Prior to creating a user account or a discussion group an administrator must be authenticated in the same manner as any other user of the system.

Draw a use case diagram that describes the system. (12 marks)

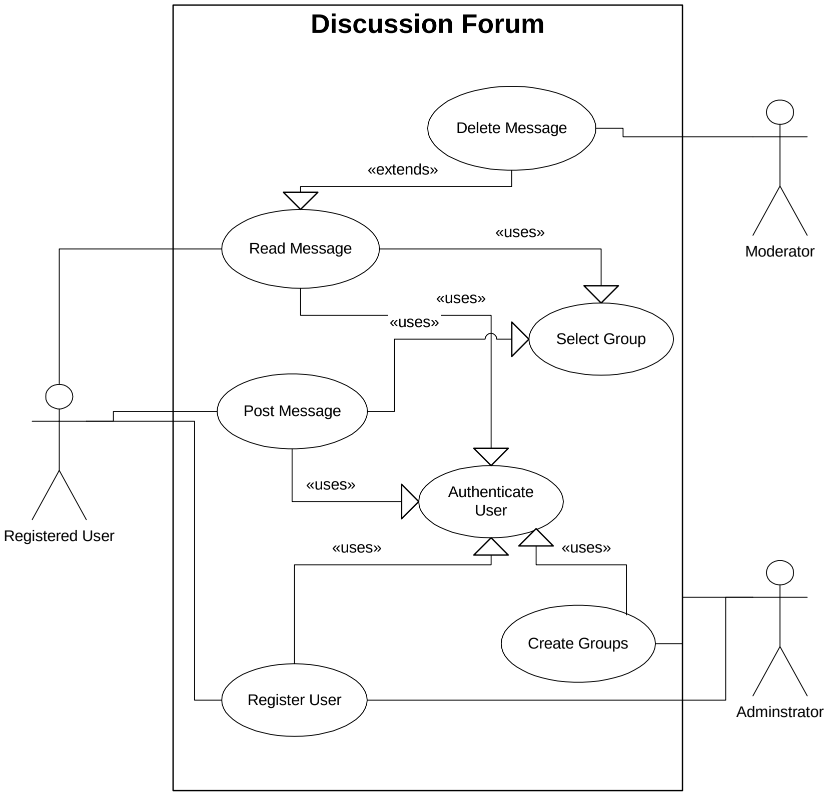
c) Write down a use case description that describes the normal sequence of events that takes place when a moderator deletes a posting. (7 marks)

Answer Pointers

a) i) Used where a use case makes use of another use case. For example where a common action such as user authentication is required

ii) Used where a use case is a specialisation of another use case (i.e. it adds something to the use case). For example deletion is a simple addition to read

b)



c) Moderator enters user name
Moderator enters password
Moderator is authenticated
Moderator selects group
Moderator selects message
Moderator ticks delete
Message is deleted

Examiner’s Guidance Notes

This question on Use case diagrams was as popular as ever. This was the first time that the <<uses>> and <<extends>> stereotypes were required as the answer to a question. Most candidates were able to describe these features but substantially less knew how to apply them in the second part of the question. The scenario descriptions were generally good but quite a few candidates did not understand the scope of the scenario or else tried to use if clauses in their scenario description. Overall the results were quite good for this question.

Question 4

a) Describe a design pattern with which you are familiar. Your answer should contain a description of the motivation for the use of this design pattern.

(8 marks)

b) Explain the circumstances in which you would use the design pattern you described in your answer to a).

(8 marks)

c) Using an object oriented programming language with which you are familiar give an example of the use of the pattern you described in your answer to a).

(9 marks)

Answer Pointers

Assuming that the candidate chose the Decorator pattern described in Gamma et al then the following answer would be appropriate. The answer would vary depending on the choice of pattern.

a) Allows the addition of additional responsibilities to an object dynamically. The Decorator pattern allows these responsibilities to be added at object level not class level. A Decorator conforms to the interface of the object it decorates and can be used in place of that object. Decorators forward messages to an object of the type that they decorate.

b) Used when the programmer wishes to add responsibilities to individual objects dynamically and transparently, also to withdraw responsibilities dynamically. May also be used where inheritance is impractical. This may be when too many subclasses are needed to support the application or when the base class does not allow inheritance.

c) Adapted from Gamma et al.

```
interface VC{
        public void draw();
        public void resize();
}

class VisualComponent implements VC(
        public VisualComponent(){
              ……….
        }

    public void draw(){
          ………….
        }

        public void resize(){
                ………….
        }
};

class Decorator implements VC{
        protected VisualComponent component;

        public void draw(){
                component.draw();
        }

        public void resize(){
                component.resize();
        }
}

class BorderDecorator extends Decorator{
        private width;

        public BorderDecorator(VisualComponent v, int bw){
                component=v;
                width=bw;
        }
        private drawBorder(){
        ………….
        }

        public void draw(){
              super.draw();
              drawBorder();
        }
}
```

Examiner's Guidance Notes

Question 4 was not as popular as 3 or 5. Candidates who answered this question seemed to be aware of a variety of design patterns but were not always able to clearly state how those patterns might be used. In general part a) and b) received the best answers whereas part c) which required outline code was poorly answered. A disappointing result given that the question is largely bookwork. The new syllabus, which is much clearer about which design patterns the candidates are required to study, may improve this area of the subject.

Question 5

a) Define the following terms. For each term show how the concept is realized in an object oriented language with which you are familiar:

i) Class;
ii) Object;
iii) Inheritance;
iv) Overloading;
v) Overriding.

(15 marks)

b) Write an account of the rationale for the development of object oriented programming. Explain how structured programming concepts and the use of abstract data types have contributed to the development of the object oriented paradigm. Your answer should emphasize the utility of the concepts listed in part a) of the question and additionally explain the importance of encapsulation and polymorphism in object oriented languages. (10 marks)

Answer Pointers

a)

i) A template for a group of objects, an object factory.

```
class MyClass{
}
```

ii) Data storage with associated methods. Member of a class

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

iii) Mechanism whereby an existing class can be modified to specify additional data and methods.

```
class YourClass extends MyClass{
}
```

iv) Mechanism whereby two methods can have the same name but accept different arguments. Operations implemented are usually identical.

```
class MyClass{
        private int number;
        public int increment(){
                return number++;
        }
        public int increment(int value){
                return number+value;
        }
}
```

v) Mechanism whereby a method in a subclass can implement extra functionality when compared to a corresponding method in a superclass.

```
class MyClass{
        private int i;
        public void increment{
                i++;
        }
}

class MySubclass extends MyClass{
        private int j;
        public void increment{
                super.increment();
                j++;
        }
}
```

b) This question allows candidates to put OO programming in the context of traditional programming techniques and software engineering principles. Candidates should be aware that programming languages do not add functionality to hardware but that they may prohibit the use of inherent functionality either by design or by accident. The purpose of a programming language is to help programmers construct reliable software easily. To this end structured programming allowed programmers to divide a task into smaller sub tasks and then solve these subtasks. This was aided by a concept of modularity where programmers tried to ensure that the solution for one of the sub tasks did not interfere with the solution of one of the other subtasks. Strong typing was added to help programmers recognise when they assigning data to variables which could not hold it. Most of this development centred on processing and not data storage. Abstract data types allowed programmers to associate data with the operations carried out on that data. Object-orientation takes these threads one stage further. The idea of class and allows the programmer to define the processing associated with a given data structure. Instantiating an object handles the memory management aspects of using these data structures. Encapsulation prevents programmers writing code where operations on one object impact unexpectedly on another. Polymorphism and inheritance and overriding are a controlled relaxation of strong typing which is too limiting. Overloading supports the good practice of associating one name with one operation.

Examiner's Guidance Notes

On the whole, the candidates answered part a) very well. Most candidates now have a good idea of what a class, object and inheritance are, some were weaker on what overloading and overriding is, by either confusing the two terms or not being able to answer these parts at all. Candidates generally performed less well on part b). Most candidates made an attempt at this section, some just reiterated the points from part a), without tailoring the answer to the question. The question made reference to abstract data types, encapsulation and polymorphism, some confused abstract data types with abstract classes and others just said what the terms meant, without any reflection on how these helped with development of object-orientation, or why they were important. A small group of candidates also discussed object-oriented design techniques (class diagrams, use cases etc), which was not relevant.

Question 6
Operations and attributes of a class are given the following visibility scopes: public, protected and private.

a) Explain the meaning of these three scoping terms. (9 marks)

b) Describe how we would use each visibility scope for both an attribute and an operation. Identify and explain where good programming practice is being adopted. (12 marks)

c) Identify a fourth level of scoping that might prove useful. What are the merits and the deficiencies of this scheme? (4 marks)

Answer Pointers
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.

Question		Mark
6	This question examines Part 3 of the syllabus “Data and implementation hiding”	
a)	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. 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. The protected features of a class are private to other classes and public to subclasses.	3 3 3
b)	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	4 4

	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.	4

c) Identify a fourth level of scoping that might prove useful. What are the merits and the deficiencies of this scheme? (4 marks)

Answer Pointers
We might consider a scoping whereby private features are shared between two or more classes. Java refers to this as package visibility and C++ as friend classes. Both break the rules of encapsulation and, perhaps, are best avoided for that reason. However, in a controlled environment there may be a case for deploying it.

Question		Mark
6	This question examines Part 3 of the syllabus “Data and implementation hiding”	
c)	We might consider a scoping whereby private features are shared between two or more classes. Java refers to this as package visibility and C++ as friend classes. Both break the rules of encapsulation and, perhaps, are best avoided for that reason. However, in a controlled environment there may be a case for deploying it.	4

[Total 25 marks]

Examiner’s Guidance Notes:
This was a popular question and generally well answered. The only observable weakness was a tendency by some candidates to answer part b) by repeating the answer they had given to part a) i.e. simply describing the terms. In fact, they were being asked to demonstrate how they would put their understanding of scoping to best software engineering effect.

Part c) probably produced the weakest response with many students unaware of package (Java) or friend (C++) scope and their usage.