Option D — Object-oriented programming

A library in a college loans books to students.

The current book borrowing system, which was developed years ago, is prone to error. You are a member of the new program development team which has decided to develop a system using an object-oriented programming (OOP) approach. The team will be divided into small sub-groups.

The program development team is aware of the following user requirements.

- Certain books are classified as **short term** and can only be borrowed for 5 days. Other books are classified as **long term** and can be borrowed for 30 days.
- Fines are charged at a rate of $1.50 for each day that a short term book is overdue.
- Fines are charged at a rate of 10 cents for each day that a long term book is overdue.
- If the overdue period exceeds 4 days for any short term loan book students are not allowed to borrow any more books and their borrowing rights are suspended; this does not apply for long term book loans.
- Students can borrow a total of 10 books.

10. (a) Outline **two** advantages that the programming team should expect from using an OOP approach. [4]

(b) Explain **two** ethical issues which should be taken into account when working on the project. [4]

(Option D continues on the following page)

(Option D, question 10 continued)

Following some analysis, the team determined that the Loan and Student classes will be used. Part of the Student class is shown below.

```
import java.util.*;
public class Student
{
  private int studentID;
  private String studentName;
  private Loan[] booksBorrowed = new Loan[10];
  private int numBooks = 0;
  public Student(int studentID, String studentName)
  {
    this.studentID=studentID;
    this.studentName=studentName;
  }
  public Loan getLoan (int x)
  {
    return this.booksBorrowed[x];
  }
  public void addLoan(Loan book)
  {
    this.booksBorrowed[numBooks] = book;
    numBooks++;
  }
  public int getStudentID()
  {
    return this.studentID;
  }
  public String getStudentName()
  {
    return this.studentName;
  }
}
```

When creating objects, encapsulation is an important design consideration.

(c) Outline, with direct reference to the `Student` class, how security can be enhanced by using encapsulation. [3]

To use the `Student` class in a program an object has to be created or instantiated.

(d) Construct a statement to create a `Student` object for a student with an ID of 93003 and a name of “Smith”. [2]

(Option D continues on the following page)

(Option D, question 10 continued)

The basic `Loan` class is shown below. Every time a book is borrowed, an instance of the `Loan` class is created. In the `Student` class, the books borrowed by a single student are stored in an array called `booksBorrowed`.

```
import java.util.*;
public class Loan
{
  private int bookID;
  private String bookTitle;
  private Date d;
  static int numBooksLoaned = 0;
  public Loan(int bookID, String bookTitle)
  {
    this.bookID = bookID;
    this.bookTitle = bookTitle;
    this.d = new Date(); //set date borrowed
    numBooksLoaned = numBooksLoaned +1;
  }
  public int getBookID()
  {
    return this.bookID;
  }
  public String getBookTitle()
  {
    return this.bookTitle;
  }
  public Date getDate()
  {
    return this.d;
  }
  public void setBookID(int id)
  {
    this.bookID = id;
  }
  public void setBookTitle(String title)
  {
    this.bookTitle = title;
  }
  public void setDate(Date d)
  {
    this.d = d;
  }
}
```

Note the use of the keyword static in the statement `static int numBooksLoaned = 0` and that this term is not used in the other variable declarations.

(e) Explain the use of the term static for the variable `numBooksLoaned` in the `Loan` class. [3]

(f) Construct the code needed to add a loan with a book ID of 212000 and a book title of "The Stars" to a `Student` object, `ST`. [3]

(Option D continues on the following page)

(Option D continued)

11. In order to store each `Student` object the team has decided to use the `borrowers` array of type `Student`. The team has also decided to use the student's ID as the index into the array. For example, a `Student` object with a student ID of 1111 would have its reference stored in position 1111 of the array.

Consider the code fragment below which would appear in a suitably constructed `main` class.

```
public static void main(String[] args)
{
     Student temp;
     Student[] borrowers = new Student[100000];

     temp = new Student(93001, "Jones");
     temp.addLoan(new Loan(210001,"The Sky"));
     borrowers[93001] = temp;

     temp = new Student(3012, "Zang");
     temp.addLoan(new Loan(210121, "The Animals"));
     borrowers[3012] = temp;

     borrowers[93001].addLoan(new Loan(210002, "The Spooks"));

     temp = new Student(93002, "Nguyen");
     temp.addLoan(new Loan(210011, "The Ocean"));
     borrowers[93002] = temp;

     System.out.println(borrowers[93001].getStudentName());
     System.out.println(borrowers[93001].getLoan(1).getBookTitle());
     System.out.println(borrowers[3012].getLoan(0).getBookTitle());
}
```

(a) Determine the output from the following statements.

(i) `System.out.println(borrowers[93001].getStudentName());` [1]

(ii) `System.out.println(borrowers[93001].getLoan(1).getBookTitle());` [1]

(iii) `System.out.println(borrowers[3012].getLoan(0).getBookTitle());` [1]

The librarian requires a display of the student name, the titles of the books and the dates that they were borrowed, corresponding to a specific student ID.

(b) Construct the method `showDetails()` that could be used in the `main` class, which accepts a student ID as a parameter, and then uses this to access the appropriate `Student` object producing the desired output as indicated above. [7]

(c) State **three** aspects of the user requirements that are not addressed by the current design. [3]

(d) Describe a modification to the `Student` class that would address **one** of the missing requirements. [2]

(Option D continues on the following page)

(Option D continued)

12. The team has decided to use inheritance in its design.

(a) Describe **one** benefit that inheritance brings to OOP. [2]

Two sub-classes will be created: `shortTerm` and `longTerm`.

(b) State **two** additional attributes that each `shortTerm` object should have. [2]

(c) Construct a suitable UML diagram showing the relationships between the `Student` class, the `Loan` superclass and the two sub-classes. **Note:** there is no need to include the attributes or methods of each class. [4]

(d) Outline, using an OOP technique, how the total number of books on loan could be displayed without processing the `borrowers` array. [3]

End of Option D