SLIIT UNI
THE KNOWLEDGE UNIVERSITY

Faculty of Computing
Year 1 Semester 2 (2025)

SE1020 – Object Oriented Programming Lab Sheet 08

Question 1

Develop a Student-Course Registration System where a Student class maintains references to courses they have enrolled in, but the Course class does not store any reference to the student. The Course class represents a university course. It contains two attributes: courseCode and courseTitle, both of type String. The class includes a parameterized constructor to initialize these attributes upon object creation. It also provides a method called displayCourseDetails() which prints the course code and title to the console. The Student class models a student who can enroll in up to three courses. It includes attributes studentID and studentName to store personal information. It maintains a fixed-size array of type Course[] to keep track of enrolled courses (with a size of 3), along with an integer variable to track the number of currently enrolled courses. The class features a default constructor, a parameterized constructor to initialize the student's details, a method enrollCourse(Course course) to add a course into the array, and a displayStudentDetails() method to print student information and all enrolled course details using the displayCourseDetails() method from the Course class. In the main method, create three Course objects with unique codes and titles. Then create a Student object using the parameterized constructor. Use the enrollCourse() method to enroll the student in all three courses. Finally, call displayStudentDetails() to print the student's information along with the details of all the enrolled courses.

A guide to implement the above scenario:

Step 1 - Create the Course Class

- Define a class named Course with two private attributes courseCode (String), courseTitle (String)
- Add a parameterized constructor to initialize both values.
- Add a method displayCourseDetails() to print course code and title.

Example code:

```
public class Course {
    private String courseCode;
    private String courseTitle;

    // Create a constructor to assign values

    public void displayCourseDetails() {
        // Print course code and title
    }
}
```

Step 2: Create the Student Class

- Define a Student class with the studentID (String), studentName (String) and courseCount (int) as attributes.
- Create an array of Course objects as below to hold max 3 courses.
  Course[ ] courses = new Course[3];
- Add default and parameterized constructors
- Add a method enrollCourse(Course c) to add a course to the array. Add course only if courseCount ¡ 3.
- Add a method displayStudentDetails() to print student information Use a loop through the array and call displayCourseDetails() for each course object.

Example code:

```
public class Student {
    private String studentID;
    private String studentName;
    private Course[] courses = new Course[3];
    private int courseCount = 0;

    // Add constructors

    public void enrollCourse(Course c) {
        // Add course to array if space is available
    }

    public void displayStudentDetails() {
        // Print student details and enrolled courses
    }
}
```

Step 3: Implement the Main Class

- Create a class named StudentCourseApp with the main() method
- Create at least 3 Course objects with different details.
- Create 1 Student object using the parameterized constructor.
- Use enrollCourse() to enroll the student in all 3 courses.
- Finally, call displayStudentDetails() to show the output.

Example code:

```
public class StudentCourseApp {
    public static void main(String[] args) {
        // Create 3 courses and 1 student
        // Enroll courses and display details
    }
}
```

Question 2

Create a Library Management System where a Library class maintains references to the books it owns, and each Book object stores a reference to the Library it belongs to. The Book class represents a book with two attributes: isbn and title, both of type String. It also contains a reference to a Library object that the book belongs to. The class includes a parameterized constructor to initialize the ISBN and title. A setLibrary() method is provided to assign the library reference after creation. The displayBookDetails() method prints the book's information and the name of the library (if available). The Library class models a library that can contain up to three books. It includes an attribute libraryName (String) and a Book[] array (size 3) to store references to Book objects. It also includes an integer variable to track how many books have been added. The class provides a parameterized constructor to set the library name. The addBook(Book book) method adds a book to the array and updates the book's reference to the current library. The displayLibraryDetails() method prints the library's name and calls each book's displayBookDetails() method. In the main method, create a Library object with a name. Then create three Book objects with unique ISBNs and titles. Add these books to the library using the addBook() method. Finally, call displayLibraryDetails() to print the library and all book information

A guide to implement the above scenario:

Step 1 - Create the Book Class

- Define a class named Book with the private attributes isbn (String), title (String), and library (Library).
- Add a parameterized constructor to initialize isbn and title.
- Add a method setLibrary(Library lib) to assign the library
- Add a method displayBookDetails() to print the book's details and associated library.

Example code:

```
public class Book {
    private String isbn;
    private String title;
    private Library library;

    // Add the constructor

    // Setter method to assign the library

    public void displayBookDetails() {
        // Print title and ISBN
        if (library != null) {
            // Print library name
        }
    }
}
```

Step 2 - Create the Library Class

- Define a class named Library with private attributes libraryName (String), and bookCount (int)
- Craete an array of Book objects as
  Book[] books = new Book[3];
- Add a constructor to initialize the library name
- Add a method addBook(Book book) to add the book to the array if bookCount ¡ 3
- Call book.setLibrary(this) within addBook() to set the back-reference
- Add a method displayLibraryDetails() to print the name and call displayBookDetails() for each book

Example code:

```
public class Library {
    private String libraryName;
    private Book[] books = new Book[3];
    private int bookCount = 0;

    // Add the constructor

    // Implement getter for LibraryNAme

    public void addBook(Book book) {
        if (bookCount < books.length) {
    //add the book to the array
    //set the library
    //Increment the book count
        }
    }

    public void displayLibraryDetails() {
        // Print the library name
        for (int i = 0; i < bookCount; i++) {
            //Display each book details
        }
    }
}
```

Step 3 - Implement the Main Class

- Create a class named LibraryApp with main() method
- Create a Library object using the constructor
- Create 3 Book objects with different details
- Add books to the library using addBook()
- Finally, call displayLibraryDetails() to print all the data

Example code:

```
public class LibraryApp {
    public static void main(String[] args) {
       // Create a Library object
       // Create 3 Book objects
      // Add the books to the library by calling addBook()
// Display Library details by calling displayLibraryDetails()
    }
}
```

Question 3

Develop a Printing System where a Printer class uses a Document object temporarily to print its content. This represents a dependency relationship, where the Printer depends on the Document only during method execution and does not store it as an attribute. The Document class represents a simple document with two attributes: title and content, both of type String. It provides a parameterized constructor to initialize these attributes. A method called displayDocument() prints the title and content of the document. The Printer class models a device that prints documents. It does not store any long-term reference to the Document object. Instead, it contains a method named printDocument(Document doc) that accepts a Document object as a parameter and uses it inside the method body to print the document details. This demonstrates a weak, short-lived dependency relationship. In the main method, create a Document object and pass it to a Printer object using the printDocument() method. The method should print a message indicating that the document is being printed and then display the document's content.

Step 1 – Create the Document Class

- Define a class Document with private attributes title (String), content (String)
- Add a constructor and displayDocument() method

Example code:

```
public class Document {
    private String title;
    private String content;

    // Add the constructor

    public void displayDocument() {
        //print title and content
    }
// Add a getter method for title
}
```

Step 2 – Create the Printer Class

- Define a class named Printer.
- Add a method printDocument(Document doc) that:
  - Prints a message like "Printing document: [title]"
  - Calls displayDocument() method on the doc object

Example code:

```
public class Printer {
    public void printDocument(Document doc) {
        // Call displayDocument()
    }
}
```

Step 3 – Implement the Main Class

- Create a class named PrinterApp with the main() method.
- Create a Document object using the constructor.
- Create a Printer object.
- Call printDocument() by passing the Document object.

Example code:

```
public class PrinterApp {
    public static void main(String[] args) {
        Document doc = new Document("Lab Report", "This is the
           final lab report for SE1020.");
        Printer printer = new Printer();
        printer.printDocument(doc);
    }
}
```