Experiment#		Student ID	
Date		Student Name	

1. Introductory Session

Aim/Objective: To understand the Object-Oriented Concepts like Class, Object, Inheritance, Encapsulation, Polymorphism and Abstraction.

Description: The student will understand the concept of OOPs.

Pre-Requisites: Computational thinking in OOP.

Tools: Eclipse IDE for Enterprise Java and Web Developers

Pre-Lab:

1) Explain how a Java class helps in structuring and organizing code in software development.

Java classes help in structuring and organizing code by encapsulating data and behaviour, promoting modularity and reusability, enabling abstraction, supporting inheritance and polymorphism and improving code organization.

2) Compare the difference between inheritance and Composition in java

Inheritance is best for establishing a heirarchial relationship and reusing code in a controlled manner, but it can lead to tight coupling and less flexibility.

Composition is preferred for its flexibility, allowing you to build complex behavior by combining simpler, reusable components without creating rigid class heirarchies.

In-Lab:

1) Write a java program that how encapsulation and inheritance are used to design a library system that manages different types of items such as books, magazines, and DVDs, each having both common and unique properties and behaviours.

Procedure/Program:

```
public abstract class LibraryItem {
    private String title;
    private String itemId;
    private boolean isAvailable;
public LibraryItem(String title, String itemId) {
    this.title = title;
    this.itemId = itemId;
    this.isAvailable = true;
} public String getTitle() {
    return title;
} public void setTitle (String title) {
    this.title = title;
} public String getItemId () {
    return itemId;
} public void setItemId (String itemId) {
    this.itemId = itemId;
}
```

```
public boolean isAvailable () [
    return isAvailable;
} public void setAvailable (boolean isAvailable) {
    this.isAvailable = isAvailable;
} public abstract void displayInfo ();
} public class Book extends LibraryItem {
  private string author;
  private string isbn;
public Book (String title, string itemId, String
    author, string isbn) {
  super (title, itemId);
   this.author = author;
   this.isbn = isbn;
} public String getAuthor () {
     return author;
} public void setAuthor (String author) {
    this.author = author;
 } public string getIsbn () {
    return isbn;
} public void setIsbn (String isbn) {
```

```
    this.isbn = isbn;
} public void displayInfo (){
System.out.println ("Book Title: " + getTitle());
System.out.println ("Author: " + author());
System.out.println ("ISBN: " + isbn);
    }
}
```

✓ **Data and Results:**

Book Title: Effective Java

Author: Joshua Bloch

ISBN: 978-0134685991

Item ID: B001

Available: Yes

✓ **Analysis and Inferences:**

The use of encapsulation and inheritance in this library system demonstrates good object-oriented design principles, contributing to a well-organized and scalable codebase.

VIVA-VOCE Questions (In-Lab):

1) Explain the role of Object in java.

Objects are instances of classes that encapsulate data and behavior. They manage and store state, represent behavior and interact with other objects.

2) State the difference between Aggregation and Composition.

Aggregation represents a "has-a" relationship where contained object can exist independently of the container object. Composition represents a "contains-a" relationship. where the contained object cannot exist independently of the container object.

3) How Polymorphism can be achieved in java?

Polymorphism enables a single interface to represent different underlying forms and is achieved through method overriding and method overloading.

4) List the differences between abstract class and an interface.

Abstract class can have both abstract and concreate methods where in interface, by default all methods are abstract until Java 8.

5) Explain the need of an interface in java.

Interface define a contract that classes must follow, promoting a consistent API and enabling polymorphism. They enhance flexibility in design and support design patterns.

Post-Lab:

1) Build an **E-Learning Platform** that offers different types of courses, such as **Programming Courses**, **Design Courses**, and **Business Courses**. Each course has unique attributes and behavior but shares common characteristics like a course name, duration, and the ability to display course details.

   To design this system, use **abstraction** to define a generic Course class and implement **polymorphism** to allow different types of courses to provide their own implementation for course-specific details.

   Design:

   1. **Abstract Class (Course)**:
      - Serves as a base class for all course types.
      - Contains common properties like courseName and duration.
      - Declares an abstract method displayDetails() for specific behavior in subclasses.
   2. **Subclasses**:
      - ProgrammingCourse, DesignCourse, and BusinessCourse inherit from the Course class.
      - Each subclass overrides the displayDetails() method to provide type-specific behavior.
   3. **Client Code**:
      - Demonstrates polymorphism by storing and working with different types of courses in a single collection.

   Procedure/Program:

```
public class Main {

  abstract static class Course {
    protected String courseName;
    protected int duration;

    public Course(String courseName, int duration) {
      this.courseName = courseName;
      this.duration = duration;
```

```
    }

    public abstract void displayDetails();
  }

  static class ProgrammingCourse extends Course {
    private String programmingLanguage;

    public ProgrammingCourse(String courseName, int duration, String
programmingLanguage) {
      super(courseName, duration);
      this.programmingLanguage = programmingLanguage;
    }

    @Override
    public void displayDetails() {
      System.out.println("Programming Course: " + courseName);
      System.out.println("Duration: " + duration + " hours");
      System.out.println("Language: " + programmingLanguage);
      System.out.println();
    }
  }

  static class DesignCourse extends Course {
    private String designTool;

    public DesignCourse(String courseName, int duration, String designTool) {
      super(courseName, duration);
      this.designTool = designTool;
    }

    @Override
    public void displayDetails() {
      System.out.println("Design Course: " + courseName);
      System.out.println("Duration: " + duration + " hours");
      System.out.println("Design Tool: " + designTool);
      System.out.println();
    }
  }
```

```
   static class BusinessCourse extends Course {
      private String businessFocus;

      public BusinessCourse(String courseName, int duration, String
businessFocus) {
         super(courseName, duration);
         this.businessFocus = businessFocus;
      }

      @Override
      public void displayDetails() {
         System.out.println("Business Course: " + courseName);
         System.out.println("Duration: " + duration + " hours");
         System.out.println("Focus: " + businessFocus);
         System.out.println();
      }
   }

   public static void main(String[] args) {
      Course[] courses = {
         new ProgrammingCourse("Java for Beginners", 40, "Java"),
         new DesignCourse("FSAD", 30, "Website Developments"),
         new BusinessCourse("AWS", 25, "Cloud")
      };

      System.out.println("E-Learning Platform Courses:\n");
      for (Course course : courses) {
         course.displayDetails();
      }
   }
}
```

OUTPUT

E-Learning Platform Courses:

Programming Course: Java for Beginners
Duration: 40 hours
Language: Java

Design Course: FSAD
Duration: 30 hours
Design Tool: Website Developments

Business Course: AWS
Duration: 25 hours
Focus: Cloud

Experiment#		Student ID	
Date		Student Name	

- ✓ **Data and Results:**

Data

The platform offers varied courses for diverse learner needs.

Result

Learners gained practical knowledge and skills from each course effectively.

- ✓ **Analysis and Inferences:**

Analysis

Programming, design, and business courses target specific industry requirements.

Inferences

Customized courses improve learning outcomes and cater to market demands.

Evaluator Remark (if Any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.