**Session 2025-2026**

|  |  |
| --- | --- |
| **Vision:**  *To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration* | **Mission:** *To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies****.*** |

**Program Educational Objectives of the program (PEO):** (broad statements that describe the professional and career accomplishments)

|  |  |  |  |
| --- | --- | --- | --- |
| PEO1 | **Preparation** | **P: Preparation** | **Pep-CL abbreviation**  **pronounce as Pep-si-lL easy to recall** |
| PEO2 | **Core Competence** | **E: Environment (Learning Environment)** |
| PEO3 | **Breadth** | **P: Professionalism** |
| PEO4 | **Professionalism** | **C: Core Competence** |
| PEO5 | **Learning Environment** | **L: Breadth (Learning in diverse areas)** |

**Program Outcomes (PO):** (statements that describe what a student should be able to do and know by the end of a program)

**Keywords of POs:**

Engineering knowledge, Problem analysis, Design/development of solutions, Conduct Investigations of Complex Problems, Engineering Tool Usage, The Engineer and The World, Ethics, Individual and Collaborative Team work, Communication, Project Management and Finance, Life-Long Learning

**PSO Keywords:** Cutting edge technologies, Research

“I am an engineer, and I know how to apply engineering knowledge to investigate, analyse and design solutions to complex problems using tools for entire world following all ethics in a collaborative way with proper management skills throughout my life.” *to contribute to the development of cutting-edge technologies and Research*.

**Integrity:** I will adhere to the Laboratory Code of Conduct and ethics in its entirety.

**Name and Signature of Student and Date**

(Signature and Date in Handwritten)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Session** | **2025-26 (EVEN)** | | **Course Name** | **JAVA FSD Lab** | |
| **Semester** | **4th** | | **Course Code** | **23ADS1407** | |
| **Roll No** | **151** | | **Name of Student** | **Parth Rebhankar** | |
|  |  | |  |  |  |
| Practical Number | | **2** | | | |
| Course Outcome | | After completing this program, students will be able to understand and implement the concepts of class, object, and methods in Java, and apply object-oriented programming principles to create and use simple Java programs. | | | |
| Aim | | WAP to demonstrate concept of Class, Object, and  methods in java. | | | |
| Problem Definition | | **Problem Statement 1**  Design and implement a Java program to create a class that represents a Student. The class should contain data members such as student name and marks, and methods to accept input and display student details. Create an object of the class and invoke its methods to demonstrate the use of class, object, and methods in Java.  **Problem Statement (Bank Related) 2**  Develop a Java program to demonstrate the concept of class, object, and methods by creating a class named BankAccount. Problem Statement 1 Design and implement a Java program to create a class that represents a Student. The class should contain data members such as student name and marks, and methods to accept input and display student details. Create an object of the class and invoke its methods to demonstrate the use of class, object, and methods in Java. Problem Statement (Bank Related) 2 : Develop a Java program to demonstrate the concept of class, object, and methods by creating a class named BankAccount. | | | |
| Theory  (100 words) | | This practical illustrates the core concepts of Object-Oriented Programming (OOP) in Java, including class, object, and methods, through real-world examples such as Student and BankAccount. A class serves as a blueprint that specifies data members and associated methods. In the Student example, the class includes attributes like student name and marks, along with methods to input and display student information. Likewise, the BankAccount class represents a bank account with properties such as account holder name, account number, and balance, and provides methods to carry out deposit, withdrawal, and display functions. Objects of these classes are instantiated to access and execute the methods, allowing interaction with the stored data. This practical aids in understanding how Java programs can be organized using classes and objects to model real-life entities, thereby enhancing code modularity, reusability, and clarity. | | | |
| Procedure and Execution  (100 Words) | | Algorithm:  Algorithm:  FOR PROBLEM STATEMENT 1:  **1. Start the program.**  **2. Define a class named Student.**  **3. Declare data members (variables) such as name and marks.**  **4. Define a method getData() to accept student details.**  **5. Define a method displayData() to display the entered details.**  **6. Create the main() method.**  **7. Inside the main() method, create an object of the Student class.**  **8. Call the getData() method using the object.**  **9. Call the displayData() method using the object.**  **10. Stop the program.**  FOR PROBLEM STATEMENT 2:  **1. Start the program.**  **2. Define a class named BankAccount.**  **3. Declare data members such as accountHolderName, accountNumber, and balance.**  **4. Create a method getAccountDetails() to accept account information. 5. Create a method deposit() to add money to the account balance.**  **6. Create a method withdraw() to deduct money from the account balance.**  **7. Create a method displayDetails() to display the account information and current balance.**  **8. Define the main() method.**  **9. Create an object of the BankAccount class inside the main() method.**  **10. Call the methods using the object to perform banking operations. 11. Stop the program.** | | | |
| Code:  Problem statement 1 :  class Student {  String name;  int marks;  void acceptDetails(String n, int m) {  name = n;  marks = m;  }  void displayDetails() {  System.out.println("Student Name: " + name);  System.out.println("Marks: " + marks);  }  }  public class StudentDemo {  public static void main(String[] args) {  Student s = new Student();  s.acceptDetails("Parth", 85);  s.displayDetails();  }  }  Problem Statement 2 :  class BankAccount {  String accountHolderName;  int accountNumber;  double balance;  void setAccountDetails(String name, int accNo, double bal) {  accountHolderName = name;  accountNumber = accNo;  balance = bal;  }  void deposit(double amount) {  balance += amount;  System.out.println("Amount Deposited: " + amount);  }  void withdraw(double amount) {  if (amount <= balance) {  balance -= amount;  System.out.println("Amount Withdrawn: " + amount);  } else {  System.out.println("Insufficient Balance");  }  }  void displayAccountDetails() {  System.out.println("Account Holder Name: " + accountHolderName);  System.out.println("Account Number: " + accountNumber);  System.out.println("Balance: " + balance);  }  }  public class BankDemo {  public static void main(String[] args) {  BankAccount acc = new BankAccount();  acc.setAccountDetails("Parth", 123456, 5000);  acc.deposit(2000);  acc.withdraw(1500);  acc.displayAccountDetails();  }  } | | | |
| Output: **For Problem Statement 1 – Student Class Program** When the program is executed, the user is prompted to enter student details such as name and marks. These inputs are stored in the data members of the Student class through an input method. Once the details are entered, the display method is invoked using the student object, and the stored information is displayed on the screen. The output confirms that the object correctly stores the data and that the class methods function as expected. This demonstrates how a class acts as a blueprint, how an object holds real-world information, and how methods are used to access and present that data. The accurate display of student details verifies the proper interaction between the class, object, and methods. **For Problem Statement 2 – BankAccount Program** When the BankAccount program runs, an object is created with the account holder’s name, account number, and initial balance. The deposit method then increases the account balance when a deposit amount is provided, and the updated balance is reflected in the output. The withdraw method checks whether sufficient funds are available before deducting the amount. If adequate balance exists, the withdrawal is completed and the new balance is shown; otherwise, an insufficient balance message is displayed. Finally, the display method outputs all account details along with the final balance. The output confirms that the object maintains its own data and that methods can modify and display this data correctly. This clearly demonstrates the practical application of class, object, and methods in Java. | | | |
| Output Analysis | | **For Problem Statement 1 – Student Class Program** When the program is executed, the user is prompted to enter student details such as name and marks. These inputs are stored in the data members of the Student class through an input method. Once the details are entered, the display method is invoked using the student object, and the stored information is displayed on the screen. The output confirms that the object correctly stores the data and that the class methods function as expected. This demonstrates how a class acts as a blueprint, how an object holds real-world information, and how methods are used to access and present that data. The accurate display of student details verifies the proper interaction between the class, object, and methods. **For Problem Statement 2 – BankAccount Program** When the BankAccount program runs, an object is created with the account holder’s name, account number, and initial balance. The deposit method then increases the account balance when a deposit amount is provided, and the updated balance is reflected in the output. The withdraw method checks whether sufficient funds are available before deducting the amount. If adequate balance exists, the withdrawal is completed and the new balance is shown; otherwise, an insufficient balance message is displayed. Finally, the display method outputs all account details along with the final balance. The output confirms that the object maintains its own data and that methods can modify and display this data correctly. This clearly demonstrates the practical application of class, object, and methods in Java. | | | |
| Link of student Github profile where lab assignment has been uploaded | |  | | | |
| Conclusion | | The Student class program demonstrates that the accurate input and display of student information illustrate how a class defines the data structure and how an object stores and handles that data through method invocations. In the BankAccount program, the deposit and withdrawal processes confirm that methods can update object data while enforcing logical conditions such as balance verification. The presentation of updated details in both programs shows that objects preserve their state and respond correctly to method execution. Overall, the results validate the effective implementation of object-oriented programming concepts and emphasize how classes and objects are utilized to represent real-world entities in Java applications. | | | |
| Plag Report (Similarity index < 12%) | |  | | | |
| Date | | 28-01-2026 | | | |

**Paste here Faculty signed observation page.**