NIRF-2023
Engineering Rank Band (151-200)
Pharmacy Rank - 88
Innovation Rank Band (51-100)

www.kiet.edu
Delhi-NCR, Ghaziabad

KIET GROUP OF INSTITUTIONS
Connecting Life with Learning

Accredited by
NAAC with Grade "A+"

Department of Information Technology
B.Tech. – II Year / IV Sem,
Assignment No. 1, (2024-25)
OOPS with JAVA (BCS403)

Instructions

1. All questions are compulsory.
2. Hard deadline is **30 April 2025**, Assignments will not be accepted after the deadline.
3. The assignment must be submitted in HARD COPY only ***(Either register or on A4 size plain white sheets)***
4. On the first page of the assignment write your Name, Semester, Section, and Roll number.
5. **Do maintain academic integrity.**

Submission deadline: 30 April 2025

Q. No.	Question	BL/KC
1	**SmartBank** is a company that plans to build a banking app using Java. Before starting, the developers need to understand the Java basics: • Why Java is platform-independent • What JVM, JRE, and the compilation process are **Question 1:** Explain **how Java achieves platform independence**. Describe the **role of JVM and JRE** in running a Java application. Sketch a simple structure of a **Java source file** for SmartBank's `Account` class.	3/P
2	Case Study 2: Core Structures for Bank Customers In SmartBank's app, they need to store customer data: name, balance, and account number. They decide to create a `Customer` class using basic programming structures. **Question 2:** ☞ Write a Java class `Customer` with: • Private variables: `name` (String), `balance` (double), and `accountNumber` (int) • A **constructor** to initialize the variables • **Getter** and **setter** methods for all variables • Use **comments** to describe each method. • Also, demonstrate the usage of **static** and **final** keywords in the class.	3/P

3	Case Study 3: Object-Oriented Approach for SmartBank SmartBank needs a hierarchy: • A `BankAccount` class • A `SavingsAccount` subclass that adds an `interestRate` • Methods to **override** `withdraw` behavior in the `SavingsAccount` **Question 3: Create the following:** • A **superclass** `BankAccount` with methods `deposit` and `withdraw`. • A **subclass** `SavingsAccount` that **overrides** the `withdraw` method. • Demonstrate **method overloading** in `BankAccount` by creating two versions of `deposit`. Highlight concepts of **inheritance, overriding, and overloading** with simple comments.	3/P
4	Case Study 4: Working with Interfaces and Abstract Classes SmartBank plans to introduce different types of accounts like loan accounts, current accounts, etc. To maintain a uniform structure, they decide to use **Interfaces** and **Abstract Classes**. **Question 4:** Define an **interface** `AccountOperations` with methods: `deposit()`, `withdraw()`, and `checkBalance()`. Create an **abstract class** `Account` that implements the interface and defines a common property `accountHolderName`. Create a concrete class `LoanAccount` that extends `Account` and provides actual implementations for all methods.	3/P