**Anudip Foundation – AJP Lab 3**

**Question 1 — Bank Account Management**

Create a class **BankAccount** with the following private fields  
accountNumber (String)  
accountHolderName (String)  
balance (double)  
**Requirements**

* Provide public getter and setter methods.
* Allow deposit and withdrawal using methods.
* Ensure balance cannot be negative.
* Prevent withdrawal if the balance is insufficient.

**PROGRAM:**

**//Lab 3 Question 1 //**

class BankAccount {

    private String accountNumber;

    private String accountHolderName;

    private double balance;

    BankAccount(String accountNumber, String accountHolderName, double openingBalance) {

        this.accountNumber     = accountNumber;

        this.accountHolderName = accountHolderName;

        setBalance(openingBalance);

    }

    /\* getters \*/

    String getAccountNumber()     { return accountNumber; }

    String getAccountHolderName() { return accountHolderName; }

    double getBalance()           { return balance; }

    /\* setters \*/

    void setAccountNumber(String accountNumber)     { this.accountNumber = accountNumber; }

    void setAccountHolderName(String accountHolderName) { this.accountHolderName = accountHolderName; }

    private void setBalance(double balance) {

        if (balance < 0) throw new IllegalArgumentException("Opening balance cannot be negative");

        this.balance = balance;

    }

    /\* behaviour \*/

    void deposit(double amount) {

        if (amount <= 0) throw new IllegalArgumentException("Deposit must be positive");

        balance += amount;

    }

    boolean withdraw(double amount) {

        if (amount <= 0) { System.out.println("Withdrawal must be positive"); return false; }

        if (amount > balance) { System.out.println("Insufficient balance"); return false; }

        balance -= amount;

        return true;

    }

    public static void main(String[] args) {

        BankAccount ba = new BankAccount("ACC123", "Dhash", 1000);

        ba.deposit(250);

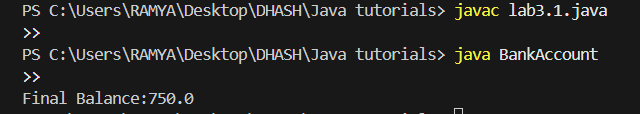
        ba.withdraw(500);

        System.out.println("Final Balance:" + ba.getBalance());

    }

}

**OUTPUT:**



Question 2 — Student Report Card

Create a class Student with the following private fields  
name (String)  
rollNumber (int)  
marks (float)  
Requirements

* Use getter and setter methods to access fields.
* Restrict marks between 0 and 100.
* Add a method to display student details.

**PROGRAM:**

import java.util.Scanner;

/\* ── Lab 3 Question 2 ── Student class \*/

class Student {

    private String name;

    private int rollNumber;

    private float marks;

    Student(String name, int rollNumber, float marks) {

        this.name = name;

        this.rollNumber = rollNumber;

        setMarks(marks);

    }

    /\* Getters \*/

    String getName() { return name; }

    int getRollNumber() { return rollNumber; }

    float getMarks() { return marks; }

    /\* Setters \*/

    void setName(String name) { this.name = name; }

    void setRollNumber(int rollNumber) { this.rollNumber = rollNumber; }

    void setMarks(float marks) {

        if (marks < 0 || marks > 100) {

            throw new IllegalArgumentException("Marks must be between 0 and 100");

        }

        this.marks = marks;

    }

    void display() {

        System.out.println("\nStudent Report Card ");

        System.out.println("Name       : " + name);

        System.out.println("Roll Number: " + rollNumber);

        System.out.println("Marks      : " + marks);

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter student name: ");

        String name = sc.nextLine();

        System.out.print("Enter roll number: ");

        int roll = sc.nextInt();

        System.out.print("Enter marks (0 to 100): ");

        float marks = sc.nextFloat();

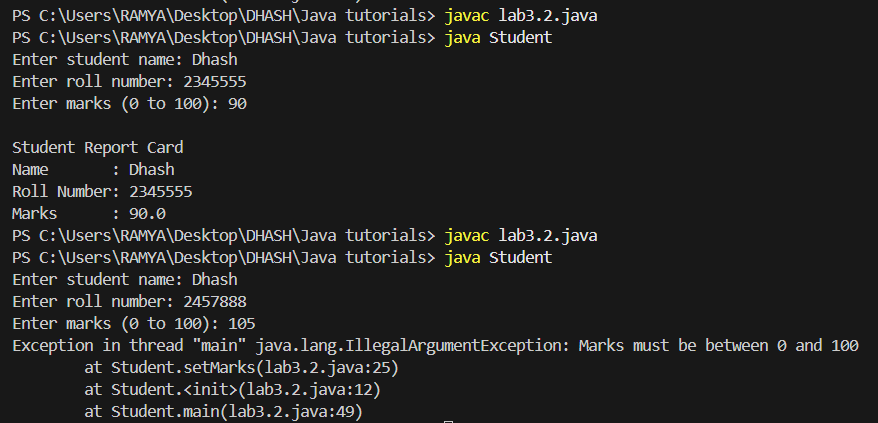
        Student s = new Student(name, roll, marks);

        s.display();

    }

}

**OUTPUT:**

****

 Question 3 — Product Catalog

Create a class Product with private fields  
productId (int)  
productName (String)  
price (double)  
Requirements

* Implement getters and setters with validation.
* Price must be greater than 0.
* Add a method to display product details.

**PROGRAM:**

import java.util.Scanner;

/\* ── Lab 3 Question 3 ── Product class with user input \*/

class Product {

    private int productId;

    private String productName;

    private double price;

    Product(int productId, String productName, double price) {

        this.productId = productId;

        this.productName = productName;

        setPrice(price);

    }

    /\* Getters \*/

    int getProductId() { return productId; }

    String getProductName() { return productName; }

    double getPrice() { return price; }

    /\* Setters \*/

    void setProductId(int productId) { this.productId = productId; }

    void setProductName(String productName) { this.productName = productName; }

    void setPrice(double price) {

        if (price <= 0) {

            throw new IllegalArgumentException("Price must be greater than 0");

        }

        this.price = price;

    }

    void display() {

        System.out.println("\n--- Product Details ---");

        System.out.println("Product ID   : " + productId);

        System.out.println("Product Name : " + productName);

        System.out.println("Price        : " + price);

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter product ID: ");

        int id = sc.nextInt();

        sc.nextLine();  // clear buffer

        System.out.print("Enter product name: ");

        String name = sc.nextLine();

        System.out.print("Enter product price (> 0): ");

        double price = sc.nextDouble();

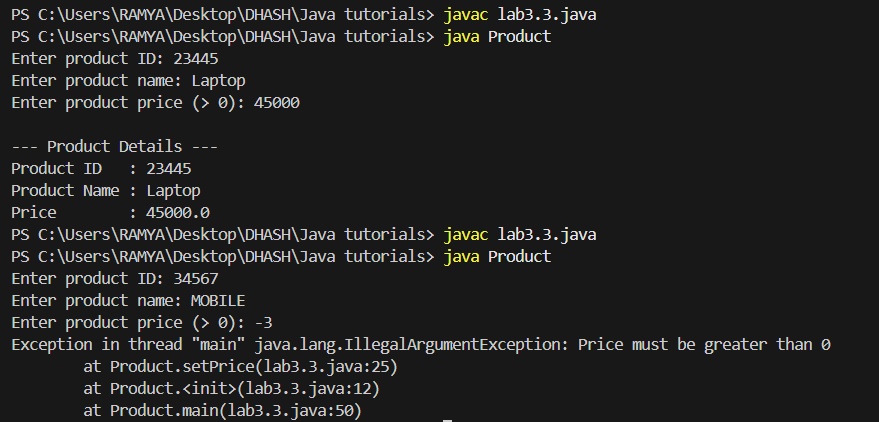
        Product p = new Product(id, name, price);

        p.display();

    }

}

**OUTPUT:**

****