Object Oriented Programming Lab File

Name: Satvik Raj
SAP Id: 500119624
Batch: 1 (CCVT)
Submitted to: Satyam Tiwari

Experiment 4: Classes (Constructors, Access Modifiers, Method Overloading, Static & Non-Static Members, this)

1. Create a Book class with attributes title and author.
   - Assign default values (e.g., "Untitled", "Unknown Author").
   - Accept user-defined values for title and author.
   - Add another constructor that also includes an integer parameter for publicationYear.

Objective: Create multiple Book objects using the different constructors and display their details.

```
public class question1Book {
    public static void main(String[] args) {
        book book1 = new book();
        book book2 = new book("Book", "John Doe", 2023);

        System.out.println("\nDetails of first book:");
        book1.displayBook();

        System.out.println("\nDetails of second book:");
        book2.displayBook();
    }
}

class book{
    private String title;
    private String author;
    private int year;
    public book() {
        this("Untitled", "Unkown Author", -1);
    }
    public book(String title, String author) {
        this(title, author, -1);
    }
    public book(String title, String author, int year) {
        this.title = title;
        this.author = author;
        this.year = year;
    }
    public void displayBook() {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        if (year != -1) {
            System.out.println("Published in Year: " + year);
        }
    }
}
```

```
Details of first book:
Title: Untitled
Author: Unkown Author

Details of second book:
Title: Book
Author: John Doe
Published in Year: 2023
```

2. Declare a Customer class with a private variable balance.
   - Implement a public addBalance(double amount) method to add funds.
   - Implement a protected deductBalance(double amount) method to reduce funds.
   - Define a default-access (package-private) method showBalance() to display the current balance.
   - Overload the addBalance(...) method to also accept an int amount for smaller deposits.

Objective: Create an instance of the Customer class Showcase method overloading for different deposit parameters.

```
public class question2Customer{
    public static void main(String[] args) {
        customer c1 = new customer();
        c1.addBalance(50.0);
        c1.deductBalance(30.0);
        c1.showBalance();
    }
}
class customer{
    private double balance;
    public void showBalance() {
        System.out.println("Current Balance: $" + String.format("%.2f", balance));
    }
    public void addBalance(double amount) {
        balance += amount;
        showDetails();
    }
    public void deductBalance(double amount) {
        if (balance < amount) {
            System.out.println("Insufficient funds for withdrawal.");
            return;
        }
        balance -= amount;
        showDetails();
    }
    private void showDetails() {
        System.out.println("\nCustomer Details");
        System.out.printf("\tBalance: %.2f", balance);
    }
}
```

```
Customer Details
        Balance: 50.00

Customer Details
        Balance: 20.00
Current Balance: $20.00
```

3. Create a Club class with a static variable clubName and a non-static variable memberName.

- Provide a static method displayClubName() to print the club's name.
- Create multiple Club objects and assign different member names to each.

Objective: Show how the static variable clubName is shared across all instances, while memberName remains unique to each object. Prove this by displaying each member's name alongside the shared club name.

```
public class question3Club{
    public static void main(String[] args) {
        Club mem1 = new Club("Myers");
        Club mem2 = new Club("Alex");
        Club mem3 = new Club("Reed");
        mem1.displayClubMember();
        mem1.displayClubName();
        mem2.displayClubMember();
        mem2.displayClubName();
        mem3.displayClubMember();
        mem3.displayClubName();

    }
}

class Club{
    private static final String CLUBNAME = "Global Club";
    String memberName;

    public Club(String memberName){
        this.memberName = memberName;
    }

    public void displayClubName() {
        System.out.println("Club Name: " + CLUBNAME);
    }

    public void displayClubMember() {
        System.out.println("Member Name: " + this.memberName);
    }
}
```

```
Member Name: Myers
Club Name: Global Club
Member Name: Alex
Club Name: Global Club
Member Name: Reed
Club Name: Global Club
```

4. Define a Car class with attributes make, model, and year.

- Create a parameterized constructor that uses the this keyword to distinguish constructor parameters from the class fields (e.g., this.make = make;).
- Instantiate a Car object with specific values (e.g., "Tesla", "Model 3", 2025) and display its details, ensuring the correct assignment of attributes using this.

```
public class question4Car {
    public static void main(String[] args) {
        car car = new car("Toyota", "Camry", 2019);
        car.displayCar();
    }
}

class car{
    String make;
    String model;
    int year;

    public car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }
    public void displayCar() {
        System.out.println("Make: " + make);
        System.out.println("Model: " + model);
        System.out.println("Year of manufacture: " + year);
    }
}
```

5. A company wants to manage a list of products with details such as product ID, name, price, and category. They also want to keep track of the total number of products and calculate the overall stock value.

- Product Class:
  - Instance variables: productId, productName, category.
  - A private variable price.
  - A default constructor setting some default values.
  - A parameterized constructor that initializes product details using this to differentiate parameters from class fields.
  - A public method getPrice() to access the private price.
  - A public method displayProductInfo() to show all details.
- Static Members:
  - A static variable totalProducts to count how many Product objects are created.
  - A static method displayTotalProducts() to print the total count.
- Method to Calculate Stock Value:
  - A method (e.g., calculateStockValue()) to multiply the price by a given quantity (demonstrate method overloading by adding an optional parameter for a discount rate).
- In the main method:
  - Create multiple Product objects using both the default and parameterized constructors.
  - Call the static method to display the total number of products.
  - Display each product's details, including the price and the calculated stock value for a given quantity.

```
public class question5Product {
    public static void main(String[] args) {
        product laptop = new product(1, "Laptop", "Electronics", 50000);
        laptop.displayProductInfo();
        System.out.println("Total number of products: " +
product.getTotalProduct());
        System.out.println("Stock value: " + laptop.calculateStockValue(10));
        System.out.println("Stock value after discount: " +
laptop.calculateStockValue(10, 0.1f));
    }
}

class product {
    int productId;
    String productName;
    String productCategory;
    private int price;
    private static int totalProduct = 0;

    public product(){
        this(0, "No Name", "No Category", 0);
    }
    public product(int productId, String productName, String productCategory, int
price) {
        this.productId = productId;
        this.productName = productName;
        this.productCategory = productCategory;
        this.price = price;
        totalProduct++;
    }
    public int getPrice() {
        return price;
    }
    public void displayProductInfo(){
        System.out.printf("Product ID: %d\nProduct Name: %s\nProduct Category:
%s\nProduct Price: %d\n",
                          productId, productName, productCategory, price);
    }
```

```
    public static int getTotalProduct() {
        return totalProduct;
    }
    public float calculateStockValue(int quantity) {
        return price * quantity;
    }
    public float calculateStockValue(int quantity, float discount) {
        return (price * quantity)*(1-discount);
    }
}
```

```
Product ID: 1
Product Name: Laptop
Product Category: Electronics
Product Price: 50000
Total number of products: 1
Stock value: 500000.0
Stock value after discount: 450000.0
```