Lab worksheet 5: Object Oriented Concepts

1. You are designing a class named **Constants** to store mathematical and scientific constants. Implement the class with the following requirements:

   Declare a final double constant named **PI** and initialize it with the value 3.14159.

   Declare a final int constant named **SPEED_OF_LIGHT** and initialize it with the value 299792458 (speed of light in meters per second).

   Implement a constructor that takes a double parameter and initializes a final double constant named **GRAVITY** with the provided value.

```java
public class Constants {
   // First approach: Initializing final field when declared
   final double PI = 3.14159;

   // Second approach: Initializing final field within a
constructor
   final int SPEED_OF_LIGHT = 299792458;
   final double GRAVITY;

   // Constructor to initialize GRAVITY with the provided
value
   public Constants(double gravityValue) {
       this.GRAVITY = gravityValue;
   }
}
```

2. Imagine you are developing a **library management system** in Java, and you want to create classes for handling books in different genres. You decide to organize these classes into packages to maintain a clean and structured codebase.

   Create a package named **library.books** that will contain classes related to books.

   Inside the **library.books** package, create a class named **FictionBook** that represents a fictional book. Include attributes such as **title**, **author**, and **genre**. Implement a method **displayInfo()** to display information about the fiction book.

   Create another package named **library.utils** that will contain utility classes.

   Inside the **library.utils** package, create a class named **BookUtils** that provides a utility method for processing books. Implement a static method named **printBookDetails** that takes a **FictionBook** object as a parameter and prints its details.

Lab worksheet 5: Object Oriented Concepts

```java
// File: FictionBook.java
package library.books;

public class FictionBook {
   private String title;
   private String author;
   private String genre;

   public FictionBook(String title, String author, String
genre) {
       this.title = title;
       this.author = author;
       this.genre = genre;
   }

   public void displayInfo() {
       System.out.println("Title: " + title);
       System.out.println("Author: " + author);
       System.out.println("Genre: " + genre);
   }
}
```

```java
// File: BookUtils.java
package library.utils;

import library.books.FictionBook;

public class BookUtils {
   public static void printBookDetails(FictionBook book) {
       System.out.println("Book Details:");
       book.displayInfo();
   }
}
```

```java
// File: LibraryManagementSystem.java
import library.books.FictionBook;
import library.utils.BookUtils;

public class LibraryManagementSystem {
   public static void main(String[] args) {
       // Creating a FictionBook object
       FictionBook fictionBook = new FictionBook("The Great
Gatsby", "F. Scott Fitzgerald", "Classic");
```

```
        // Using the utility method to print book details
        BookUtils.printBookDetails(fictionBook);
    }
}
```

3. You are tasked with creating a simple class called **BankAccount** that represents a user's bank account. Implement the class with the following requirements:

   Create a private variable **balance** to store the account balance.

   Implement a constructor that initializes the **balance** to 0.

   Create public methods **deposit** and **withdraw** to modify the account **balance**.

   Ensure that the **withdraw** method cannot make the **balance** negative. If an attempt is made to **withdraw** more than the current **balance**, print an error message and do not perform the **withdrawal**.

   Implement a public method **getBalance** to retrieve the current **balance**.

   In a separate class named **BankManagementSystem**, demonstrate the usage of the **BankAccount** class by performing the following steps:

   Create an instance of **BankAccount**.
   Deposit an amount into the account.
   Withdraw an amount from the account.
   Display the current **balance**.

   Ensure that the **BankAccount** class uses appropriate access modifiers for encapsulation.

```java
// BankAccount.java
public class BankAccount {
    private double balance;

    public BankAccount() {
        this.balance = 0;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: $" + amount);
    }

    public void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Error: Insufficient funds for
withdrawal.");
        } else {
            balance -= amount;
            System.out.println("Withdrawn: $" + amount);
        }
    }

    public double getBalance() {
        return balance;
    }
}
```

```java
// BankManagementSystem.java
public class BankManagementSystem {
    public static void main(String[] args) {
        // Creating an instance of BankAccount
        BankAccount account = new BankAccount();

        // Depositing and withdrawing funds
        account.deposit(1000);
        account.withdraw(500);
        account.withdraw(800); // This should print an error
message.

        // Displaying the current balance
        System.out.println("Current Balance: $" +
account.getBalance());
    }
}
```

4. You are developing a utility class named **MathOperations** to perform common mathematical operations. Implement the class with the following requirements:

   Declare a static variable named **pi** of type double and initialize it with the value 3.14159.

   Implement a static method named **calculateCircleArea** that takes the **radius** as a parameter and returns the **area of a circle** using the formula: **area = pi * radius * radius.**

Lab worksheet 5: Object Oriented Concepts

Implement a static method named **calculateSquareArea** that takes the **side length** as a parameter and returns the **area of a square** using the **formula: area = side * side.**

Ensure that the **pi** variable is accessible only within the **MathOperations** class.

Demonstrate the usage of the **MathOperations** class in a separate class named **GeometryCalculator** by performing the following steps:

Print the value of the static variable **pi** from the **MathOperations** class.
Calculate and print the **area of a circle** with a **radius** of 5 using the **calculateCircleArea** method.
Calculate and print the **area of a square** with a **side length** of 4 using the **calculateSquareArea** method.

```
// MathOperations.java
public class MathOperations {
   // Static variable accessible only within the
MathOperations class
   private static double pi = 3.14159;

   // Static method to calculate the area of a circle
   public static double calculateCircleArea(double radius) {
       return pi * radius * radius;
   }

   // Static method to calculate the area of a square
   public static double calculateSquareArea(double side) {
       return side * side;
   }
}
```

```
// GeometryCalculator.java
public class GeometryCalculator {
   public static void main(String[] args) {
       // Accessing the static variable pi from the
MathOperations class
       System.out.println("Value of pi: " +
MathOperations.pi);

       // Calculating and printing the area of a circle
       double circleArea =
MathOperations.calculateCircleArea(5);
```

```
        System.out.println("Area of a circle with radius 5: " +
circleArea);

        // Calculating and printing the area of a square
        double squareArea =
MathOperations.calculateSquareArea(4);
        System.out.println("Area of a square with side length
4: " + squareArea);
    }
}
```

5. You are tasked with implementing a program that calculates the area of a rectangle and a square. Utilize the concept of scope in Java to create a solution. Implement the following:

   Create a class named **AreaCalculator**.

   Declare a static method named **calculateRectangleArea** that takes two parameters (**length** and **width**) and calculates the **area of a rectangle**.

   Declare a static method named **calculateSquareArea** that takes one parameter (**sideLength**) and calculates the **area of a square**.

   In the main method:
   - Declare a variable **rectangleLength** and assign it the value 5.
   - Declare a variable **rectangleWidth** and assign it the value 8.
   - Declare a variable **squareSideLength** and assign it the value 4.
   - Call the **calculateRectangleArea** method and print the result.
   - Call the **calculateSquareArea** method and print the result.

```
public class AreaCalculator {
    // Method to calculate the area of a rectangle
    static double calculateRectangleArea(double length, double
width) {
        return length * width;
    }

    // Method to calculate the area of a square
    static double calculateSquareArea(double sideLength) {
        return sideLength * sideLength;
    }

    public static void main(String[] args) {
        // Method scope variables
        double rectangleLength = 5;
        double rectangleWidth = 8;
        double squareSideLength = 4;

        // Calculate and display the area of a rectangle
        double rectangleArea =
calculateRectangleArea(rectangleLength, rectangleWidth);
        System.out.println("Area of Rectangle: " +
rectangleArea);

        // Calculate and display the area of a square
        double squareArea =
calculateSquareArea(squareSideLength);
        System.out.println("Area of Square: " + squareArea);

    }
}
```