RegNo: 23MCA **Name**: Vinayak Kumar Singh

Java Programming Lab (PMCA502P)

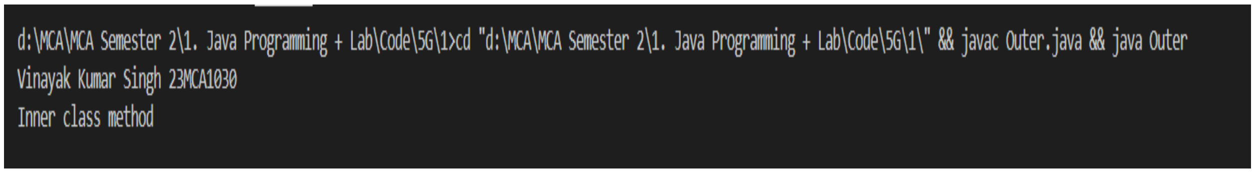
Ex 5.g Inner Class /Nested Class

1.Write a Java program that demonstrates the use of outer and inner classes. Create an outer class named Outer and an inner class named Inner. Inside the inner class, define a method named displayInner() that prints "Inner class method". In the main class, create an instance of the outer class and call the inner class method.

Code:

```java
public class Outer {
    // Inner class with access to outer class members
    public class Inner {
        // Method to print a message from the inner class
        public void displayInner() {
            System.out.println("Vinayak Kumar Singh 23MCA1030");
            System.out.println("Inner class method");
        }
    }

    public static void main(String[] args) {
        // Create an instance of the outer class
        Outer outerObject = new Outer();
        // Create an instance of the inner class using the outer object
        Outer.Inner innerObject = outerObject.new Inner();
        // Call the inner class method
        innerObject.displayInner();
    }
}
```

Output:



```
d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\1>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\1\" && javac Outer.java && java Outer
Vinayak Kumar Singh 23MCA1030
Inner class method
```

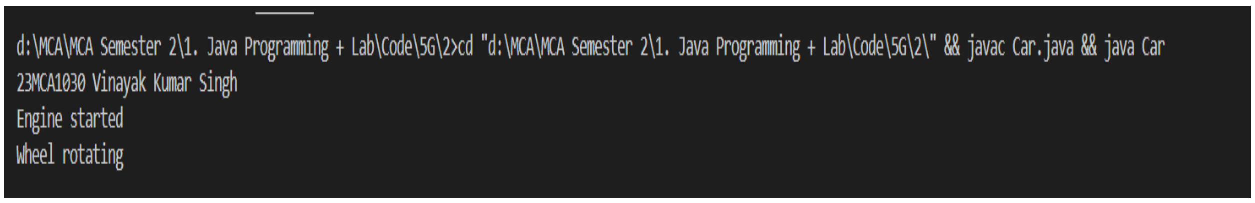
2: Encapsulation with Inner Class

Create a class named Car with an outer class named Engine and an inner class named Wheel. The Car class should have methods to start the engine and rotate the wheels.In the Engine class, define a method named startEngine() that prints "Engine started".In the Wheel class, define a method named rotateWheel() that prints "Wheel rotating".In the main class, create an instance of the Car class and call both the startEngine() and rotateWheel() methods.

Code:

```java
public class Car {
    public class Engine {
        // Method to start engine
        public void startEngine() {
            System.out.println("Engine started");
        }
    }
    // Inner class
    public class Wheel {
        // Method to rotate wheel
        public void rotateWheel() {
            System.out.println("Wheel rotating");
        }
    }
    // Method to demonstrate starting engine and rotating wheel
    public void drive() {
        // Created instance of Engine class
        Engine carEngine = new Engine();
        // Create instance of Wheel class
        Wheel carWheel = new Wheel();
        // Starting engine
        carEngine.startEngine();
        // Rotate the wheel
        carWheel.rotateWheel();
    }
    public static void main(String[] args) {
        // Create an instance of Car class
        Car myCar = new Car();
        // Called drive method to start engine and rotate wheel
        System.out.println("23MCA1030 Vinayak Kumar Singh");
        myCar.drive();
    }
}
```

Output:



```
d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\2>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\2\" && javac Car.java && java Car
23MCA1030 Vinayak Kumar Singh
Engine started
Wheel rotating
```

3: Accessing Outer Class Members from Inner Class

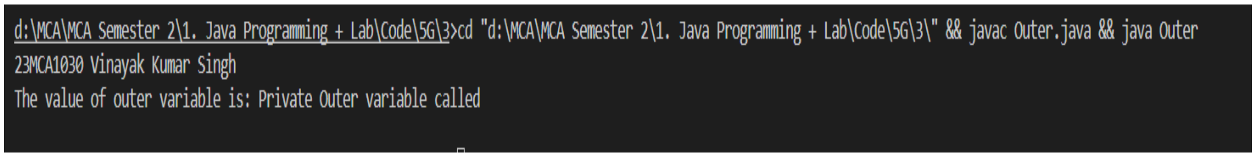
Create an outer class named Outer with a private member variable outerVariable. Inside the outer class, create an inner class named Inner with a method named displayOuterVariable() that prints the value of outerVariable.

In the main class, create an instance of the outer class and call the method displayOuterVariable() from the inner class.

Code:

```
public class Outer {
    private String outerVariable = "Private Outer variable called";
    public class Inner {
        public void displayOuterVariable() {
            // Accessing outerVariable using outer object reference
            System.out.println("The value of outer variable is: " +
Outer.this.outerVariable);
        }
    }
    public static void main(String[] args) {
        // Created instance of outer class
        Outer outerObject = new Outer();
        // Created instance of inner class associated with the outer object
        Inner innerObject = outerObject.new Inner();
        // Called inner class method to access and print outer variable
        System.out.println("23MCA1030 Vinayak Kumar Singh");
        innerObject.displayOuterVariable();
    }
}
```

Output:



```
d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\3>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\3\" && javac Outer.java && java Outer
23MCA1030 Vinayak Kumar Singh
The value of outer variable is: Private Outer variable called
```

4: Static Inner Class

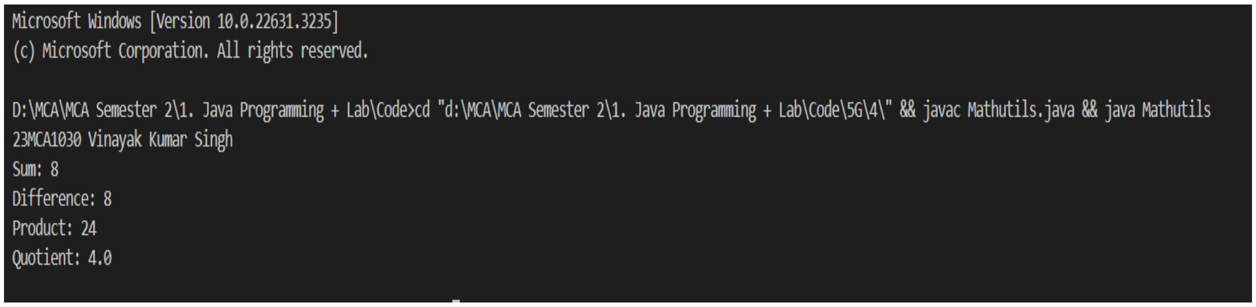
Create a class named MathUtils with an outer class named Calculator and a static inner class named Operations. Inside the Operations class, define static methods for addition, subtraction, multiplication, and division. In the main class, create an instance of the Calculator. Operations class and call each of the static methods.

Code:

```
public class Mathutils  {
   // Static inner class for performing basic arithmetic operations
   public static class Operations {
      // Static methods for arithmetic operations
      public static int add(int x, int y) {
         return x + y;
      }
      public static int subtract(int x, int y) {
         return x - y;
      }
      public static int multiply(int x, int y) {
         return x * y;
      }
      public static double divide(double x, double y) {
         if (y == 0) {
            throw new IllegalArgumentException("Division by zero is not
allowed.");
```

```java
            }
            return x / y;
        }
    }
    public static void main(String[] args) {
        // Call static methods directly using the outer class name
        int sum = Mathutils.Operations.add(5, 3);
        int difference = Mathutils.Operations.subtract(10, 2);
        int product = Mathutils.Operations.multiply(4, 6);
        double quotient = Mathutils.Operations.divide(12.0, 3.0);
        System.out.println("23MCA1030 Vinayak Kumar Singh");
        System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);
        System.out.println("Product: " + product);
        System.out.println("Quotient: " + quotient);
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.22631.3235]
(c) Microsoft Corporation. All rights reserved.

D:\MCA\MCA Semester 2\1. Java Programming + Lab\Code>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\4\" && javac Mathutils.java && java Mathutils
23MCA1030 Vinayak Kumar Singh
Sum: 8
Difference: 8
Product: 24
Quotient: 4.0
```

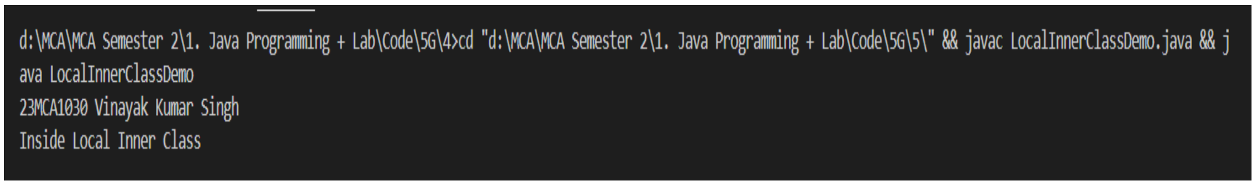
5: Local Inner Class

Write a program that demonstrates the use of a local inner class. Inside the main method, define a local inner class named LocalInner with a method named display() that prints "Inside Local Inner Class".

Code:

```java
public class LocalInnerClassDemo {
    public static void main(String[] args) {
        // Local inner class definition
        class LocalInner {
            public void display() {
                System.out.println("Inside Local Inner Class");
            }
        }
        // Create an instance of local inner class
        LocalInner innerObject = new LocalInner();
        // Calling method of local inner class
        System.out.println("23MCA1030 Vinayak Kumar Singh");
        innerObject.display();
    }
}
```

Output:



```
d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\4>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\5\" && javac LocalInnerClassDemo.java && j
ava LocalInnerClassDemo
23MCA1030 Vinayak Kumar Singh
Inside Local Inner Class
```

6. Implement a Shopping Cart

You are tasked with creating a simple shopping cart system in Java. The shopping cart should have classes representing the cart itself and the items that can be added to it. Each item should have attributes such as name, price, and quantity.

Requirements:

Create an outer class named ShoppingCart.
Inside the ShoppingCart class, define an inner class named CartItem.
The CartItem class should have attributes for the item name, price, and quantity.
Implement methods in the ShoppingCart class to add items, remove items, and calculate the total price of all items in the cart.
Write a main class to test your shopping cart system. Create an instance of ShoppingCart and add several items to it. Test the methods to add, remove, and calculate the total price.

Detailed Input:

- You can manually create instances of items with names, prices, and quantities for testing purposes.

Detailed Output:

- After each operation (adding, removing items, calculating total price), print the current state of the shopping cart (i.e., the list of items and the total price).

Sample Input and Output:

```
public class ShoppingCartTest {
   public static void main(String[] args) {
      // Create a shopping cart
      ShoppingCart cart = new ShoppingCart();
```

```
        // Add items to the cart
        cart.addItem("T-shirt", 15.99, 2);
        cart.addItem("Jeans", 29.99, 1);
        cart.addItem("Sneakers", 49.99, 1);

        // Display items in the cart
        System.out.println("Items in the cart:");
        cart.displayItems();

        // Calculate total price
        double totalPrice = cart.calculateTotalPrice();
        System.out.println("\nTotal price of items in the cart: $" + totalPrice);

        // Remove an item from the cart
        cart.removeItem("Jeans");

        // Display items in the cart after removal
        System.out.println("\nItems in the cart after removal:");
        cart.displayItems();

        // Calculate total price again after removal
        totalPrice = cart.calculateTotalPrice();
        System.out.println("\nTotal price of items in the cart after removal: $" +
totalPrice);
    }
}
```

Expected Output:

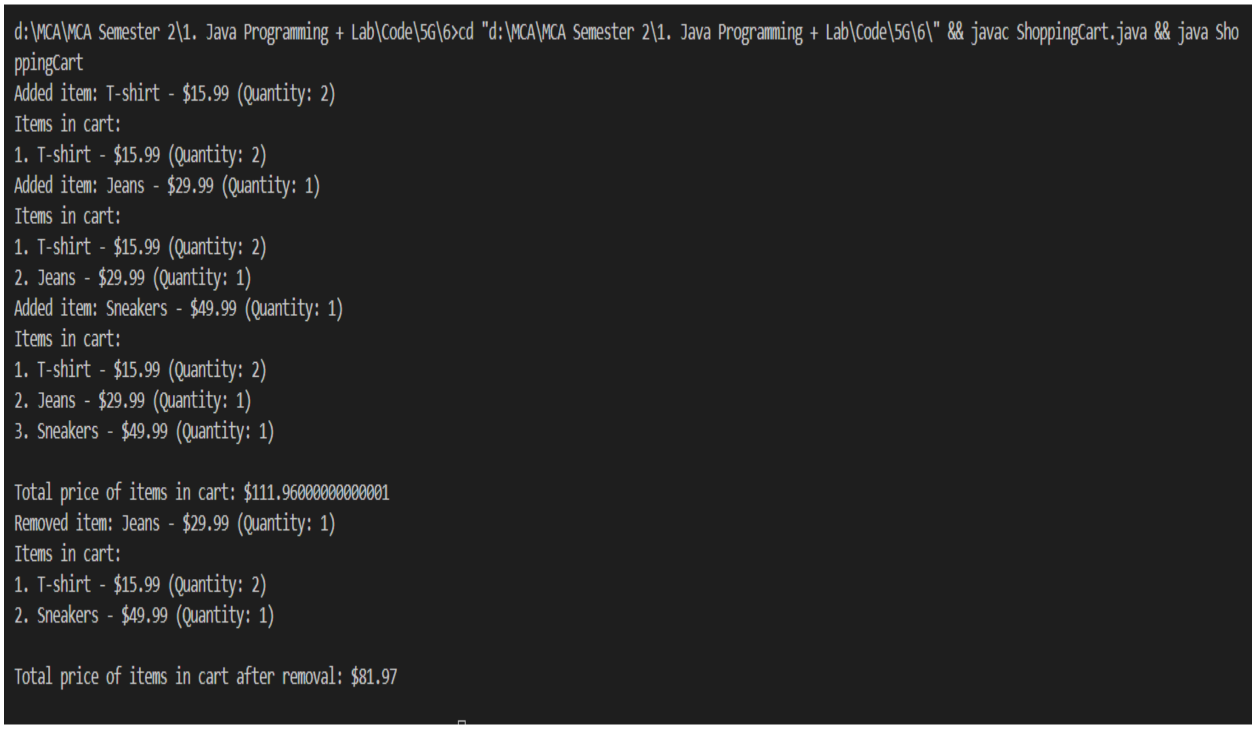
```
Items in the cart:
1. T-shirt - $15.99 (Quantity: 2)
2. Jeans - $29.99 (Quantity: 1)
3. Sneakers - $49.99 (Quantity: 1)

Total price of items in the cart: $111.96
```

```
Items in the cart after removal:
1. T-shirt - $15.99 (Quantity: 2)
2. Sneakers - $49.99 (Quantity: 1)

Total price of items in the cart after removal: $81.97
```

Code:

```java
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
public class ShoppingCart {
    // Inner class representing items in cart
    public class CartItem {
        private String itemName;
        private double price;
        private int quantity;
        public CartItem(String itemName, double price, int quantity) {
            this.itemName = itemName;
            this.price = price;
            this.quantity = quantity;
        }
        public String getItemName() {
            return itemName;
        }
        public double getPrice() {
            return price;
        }
        public int getQuantity() {
            return quantity;
        }
        @Override
        public String toString() {
            return itemName + " - $" + price + " (Quantity: " + quantity + ")";
        }
    }
    private List<CartItem> items;
    public ShoppingCart() {
        this.items = new ArrayList<>();
    }
    // Method to add items to the cart
    public void addItem(String itemName, double price, int quantity) {
        CartItem newItem = new CartItem(itemName, price, quantity);
        items.add(newItem);
        System.out.println("Added item: " + newItem);
        displayItems();
    }
    // Method to remove an item from the cart
    public void removeItem(String itemName) {
        Iterator<CartItem> iterator = items.iterator();
        while (iterator.hasNext()) {
            CartItem currentItem = iterator.next();
            if (currentItem.getItemName().equals(itemName)) {
                iterator.remove();
                System.out.println("Removed item: " + currentItem);
                displayItems();
                return;
            }
        }
        System.out.println("Item not found: " + itemName);
    }
    // Method to calculate total price of items in cart
    public double calculateTotalPrice() {
        double totalPrice = 0;
        for (CartItem item : items) {
            totalPrice += item.getPrice() * item.getQuantity();
        }
        return totalPrice;
    }
    // Method to display items in cart
    public void displayItems() {
        System.out.println("Items in cart:");
        int index = 1;
        for (CartItem item : items) {
            System.out.println(index + ". " + item);
            index++;
        }
    }
    public static void main(String[] args) {
        // Create a shopping cart
        ShoppingCart cart = new ShoppingCart();
        // Add items to cart
        cart.addItem("T-shirt", 15.99, 2);
        cart.addItem("Jeans", 29.99, 1);
        cart.addItem("Sneakers", 49.99, 1);
        // Display items in cart
        System.out.println("\nTotal price of items in cart: $" +
cart.calculateTotalPrice());
        // Remove an item from cart
        cart.removeItem("Jeans");
        // Display items in cart after removal
        System.out.println("\nTotal price of items in cart after removal: $" +
cart.calculateTotalPrice());
    }
}
```

Output:



```
d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\6>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\6\" && javac ShoppingCart.java && java Sho
ppingCart
Added item: T-shirt - $15.99 (Quantity: 2)
Items in cart:
1. T-shirt - $15.99 (Quantity: 2)
Added item: Jeans - $29.99 (Quantity: 1)
Items in cart:
1. T-shirt - $15.99 (Quantity: 2)
2. Jeans - $29.99 (Quantity: 1)
Added item: Sneakers - $49.99 (Quantity: 1)
Items in cart:
1. T-shirt - $15.99 (Quantity: 2)
2. Jeans - $29.99 (Quantity: 1)
3. Sneakers - $49.99 (Quantity: 1)

Total price of items in cart: $111.96000000000001
Removed item: Jeans - $29.99 (Quantity: 1)
Items in cart:
1. T-shirt - $15.99 (Quantity: 2)
2. Sneakers - $49.99 (Quantity: 1)

Total price of items in cart after removal: $81.97
```

7: Implement a Library System

You are tasked with implementing a library system in Java. The library system should have classes representing both the library itself and the books it contains. Each book should have attributes such as title, author, and genre.

Instructions:

Create a Library class that contains an inner class named Book.
The Book class should have attributes for title, author, and genre.

Implement methods in the Library class to add books, remove books, and display all books in the library.
Write a main class to test your library system. Create an instance of Library and add several books to it. Test the methods to add, remove, and display books.

Detailed Input:

- For testing purposes, you can manually create instances of books with titles, authors, and genres.

Detailed Output:

- After each operation (adding, removing, displaying books), print the current state of the library (i.e., the list of books).

Sample Input and Output:

```
public class LibrarySystemTest {
    public static void main(String[] args) {
        // Create a library
        Library library = new Library();

        // Add books to the library
        library.addBook("The Great Gatsby", "F. Scott Fitzgerald", "Fiction");
        library.addBook("To Kill a Mockingbird", "Harper Lee", "Fiction");
        library.addBook("1984", "George Orwell", "Dystopian Fiction");

        // Display all books in the library
        System.out.println("Books in the library:");
        library.displayBooks();

        // Remove a book from the library
        library.removeBook("To Kill a Mockingbird");

        // Display all books in the library after removal
        System.out.println("\nBooks in the library after removal:");
        library.displayBooks();
    }
}
```

Expected Output:

```
Books in the library:
1. Title: The Great Gatsby, Author: F. Scott Fitzgerald, Genre: Fiction
2. Title: To Kill a Mockingbird, Author: Harper Lee, Genre: Fiction
3. Title: 1984, Author: George Orwell, Genre: Dystopian Fiction

Books in the library after removal:
1. Title: The Great Gatsby, Author: F. Scott Fitzgerald, Genre: Fiction
2. Title: 1984, Author: George Orwell, Genre: Dystopian Fiction
```

Code:

Library.java

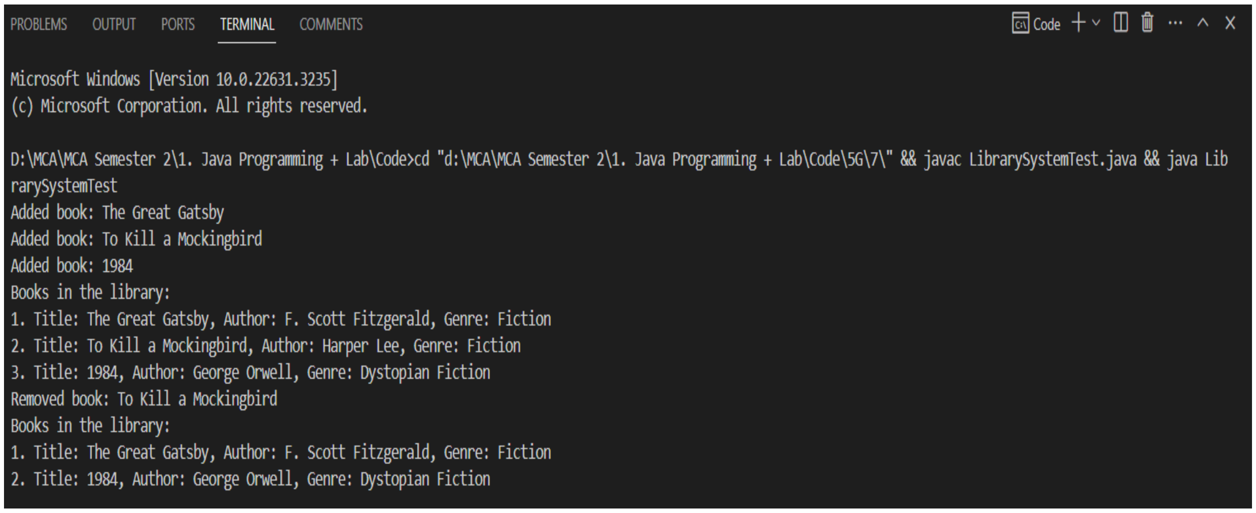
```java
import java.util.ArrayList;
import java.util.List;

public class Library {

    // Inner class representing a book in the library
    public class Book {
        private String title;
        private String author;
        private String genre;

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

        public String getTitle() {
            return title;
        }

        public String getAuthor() {
            return author;
        }

        public String getGenre() {
            return genre;
        }

        @Override
        public String toString() {
            return "Title: " + title + ", Author: " + author + ", Genre: " + genre;
        }
    }

    private List<Book> books = new ArrayList<>();

    public void addBook(String title, String author, String genre) {
        Book newBook = new Book(title, author, genre);
        books.add(newBook);
        System.out.println("Added book: " + newBook.getTitle());
    }

    public void removeBook(String title) {
        boolean removed = books.removeIf(book -> book.getTitle().equals(title));
        if (removed) {
            System.out.println("Removed book: " + title);
        } else {
            System.out.println("Book not found: " + title);
        }
    }

    public void displayBooks() {
        if (books.isEmpty()) {
            System.out.println("There are no books in the library.");
        } else {
            System.out.println("Books in the library:");
            for (int i = 0; i < books.size(); i++) {
                System.out.println((i + 1) + ". " + books.get(i));
            }
        }
    }
}
```

Code:

LibrarySystemTest.java

```
public class LibrarySystemTest {

    public static void main(String[] args) {
        Library library = new Library();

        library.addBook("The Great Gatsby", "F. Scott Fitzgerald", "Fiction");
        library.addBook("To Kill a Mockingbird", "Harper Lee", "Fiction");
        library.addBook("1984", "George Orwell", "Dystopian Fiction");

        library.displayBooks();

        library.removeBook("To Kill a Mockingbird");

        library.displayBooks();
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.22631.3235]
(c) Microsoft Corporation. All rights reserved.

D:\MCA\MCA Semester 2\1. Java Programming + Lab\Code>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Code\5G\7\" && javac LibrarySystemTest.java && java Lib
rarySystemTest
Added book: The Great Gatsby
Added book: To Kill a Mockingbird
Added book: 1984
Books in the library:
1. Title: The Great Gatsby, Author: F. Scott Fitzgerald, Genre: Fiction
2. Title: To Kill a Mockingbird, Author: Harper Lee, Genre: Fiction
3. Title: 1984, Author: George Orwell, Genre: Dystopian Fiction
Removed book: To Kill a Mockingbird
Books in the library:
1. Title: The Great Gatsby, Author: F. Scott Fitzgerald, Genre: Fiction
2. Title: 1984, Author: George Orwell, Genre: Dystopian Fiction
```