Assignment 1

Student Name: Sarthak Rana **UID:22BCS16222**
Branch: BE-CSE **Section/Group:642/B**
Semester:6th **Date of Performance:**
Subject Name: Project Based Learning in Java with Lab **Subject Code: 22CSH-359**

Q:1 Develop a Java program showcasing the concept of inheritance. Create a base class and a derived class with appropriate methods and fields.

Ans:
```
// Base class
class Animal {
    String name;
    int age;

    // Constructor
    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Method to display animal details
    public void displayInfo() {
        System.out.println("Animal Name: " + name);
        System.out.println("Age: " + age);
    }

    // Method for sound (to be overridden)
    public void makeSound() {
        System.out.println("Animal makes a sound.");
    }
}

// Derived class (Child class)
class Dog extends Animal {
    String breed;
    // Constructor
    public Dog(String name, int age, String breed) {
        super(name, age); // Calling parent class constructor
        this.breed = breed;
    }

    // Overriding the makeSound() method
    @Override
    public void makeSound() {
        System.out.println(name + " barks: Woof! Woof!");
    }

    // Additional method specific to Dog
    public void showBreed() {
        System.out.println("Breed: " + breed);
    }
}

// Main class
public class InheritanceDemo {
    public static void main(String[] args) {
        // Creating an object of Dog class
        Dog myDog = new Dog("Buddy", 3, "Golden Retriever");

        // Accessing inherited and overridden methods
        myDog.displayInfo();
        myDog.makeSound();
        myDog.showBreed();
    }
}
```

Q:2 Implement a Java program that uses method overloading to perform different mathematical operations.

Ans: // Class containing overloaded methods for mathematical operations

```
class MathOperations {

    // Method to add two integers
    public int add(int a, int b) {
        return a + b;
    }

    // Method to add three integers
    public int add(int a, int b, int c) {
        return a + b + c;
    }

    // Method to add two floating-point numbers
    public double add(double a, double b) {
        return a + b;
    }

    // Method to multiply two integers
    public int multiply(int a, int b) {
        return a * b;
    }

    // Method to multiply three integers
    public int multiply(int a, int b, int c) {
        return a * b * c;
    }

    // Method to multiply two floating-point numbers
    public double multiply(double a, double b) {
        return a * b;
    }
}

// Main class
public class MethodOverloadingDemo {
    public static void main(String[] args) {
        MathOperations mathOps = new MathOperations();

        // Testing overloaded addition methods
        System.out.println("Addition of 5 and 10: " + mathOps.add(5, 10));
        System.out.println("Addition of 5, 10 and 15: " + mathOps.add(5, 10, 15));
        System.out.println("Addition of 5.5 and 2.3: " + mathOps.add(5.5, 2.3));

        // Testing overloaded multiplication methods
        System.out.println("Multiplication of 4 and 6: " + mathOps.multiply(4, 6));
        System.out.println("Multiplication of 2, 3 and 4: " + mathOps.multiply(2, 3, 4));
        System.out.println("Multiplication of 3.5 and 2.0: " + mathOps.multiply(3.5, 2.0));
    }
}
```

Q:3 Define an interface in Java and create a class that implements it, demonstrating the concept of abstraction.

Ans: // Defining an interface

```
interface Vehicle {
    void start();  // Abstract method (no implementation)
    void stop();   // Abstract method (no implementation)
}

// Implementing the interface in a class
class Car implements Vehicle {
    private String model;

    // Constructor
    public Car(String model) {
        this.model = model;
    }

    // Implementing the start method
    @Override
    public void start() {
        System.out.println(model + " is starting with a key.");
    }

    // Implementing the stop method
    @Override
    public void stop() {
        System.out.println(model + " is stopping.");
    }
}

// Main class
public class InterfaceDemo {
    public static void main(String[] args) {
        Vehicle myCar = new Car("Toyota Corolla"); // Upcasting
        myCar.start(); // Calls implemented method
        myCar.stop();  // Calls implemented method
    }
}
```

Q:4 Create a custom exception class in Java. Write a program that throws this custom exception in a specific scenario.

Ans: // Custom exception class

```
class InsufficientBalanceException extends Exception {
    public InsufficientBalanceException(String message) {
        super(message); // Passing message to Exception class
    }
}

// BankAccount class
class BankAccount {
    private double balance;

    // Constructor
    public BankAccount(double balance) {
        this.balance = balance;
    }

    // Method to withdraw money
    public void withdraw(double amount) throws InsufficientBalanceException {
        if (amount > balance) {
            throw new InsufficientBalanceException("Insufficient balance! Available balance: " +
balance);
        }
        balance -= amount;
        System.out.println("Withdrawal successful! New balance: " + balance);
    }
}

// Main class
public class CustomExceptionDemo {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(5000); // Initial balance
        try {
            account.withdraw(6000); // Trying to withdraw more than available balance
        } catch (InsufficientBalanceException e) {
            System.out.println("Exception Caught: " + e.getMessage());
        }
    }
}
```

Q:5 Explain the difference between the throw and throws keywords in Java. Provide examples illustrating their usage.

Ans: class ThrowExample {

```
    public static void validateAge(int age) {
        if (age < 18) {
            throw new IllegalArgumentException("Age must be 18 or above.");
        }
        System.out.println("Valid age!");
    }

    public static void main(String[] args) {
        validateAge(16); // This will throw an exception
    }
}
```

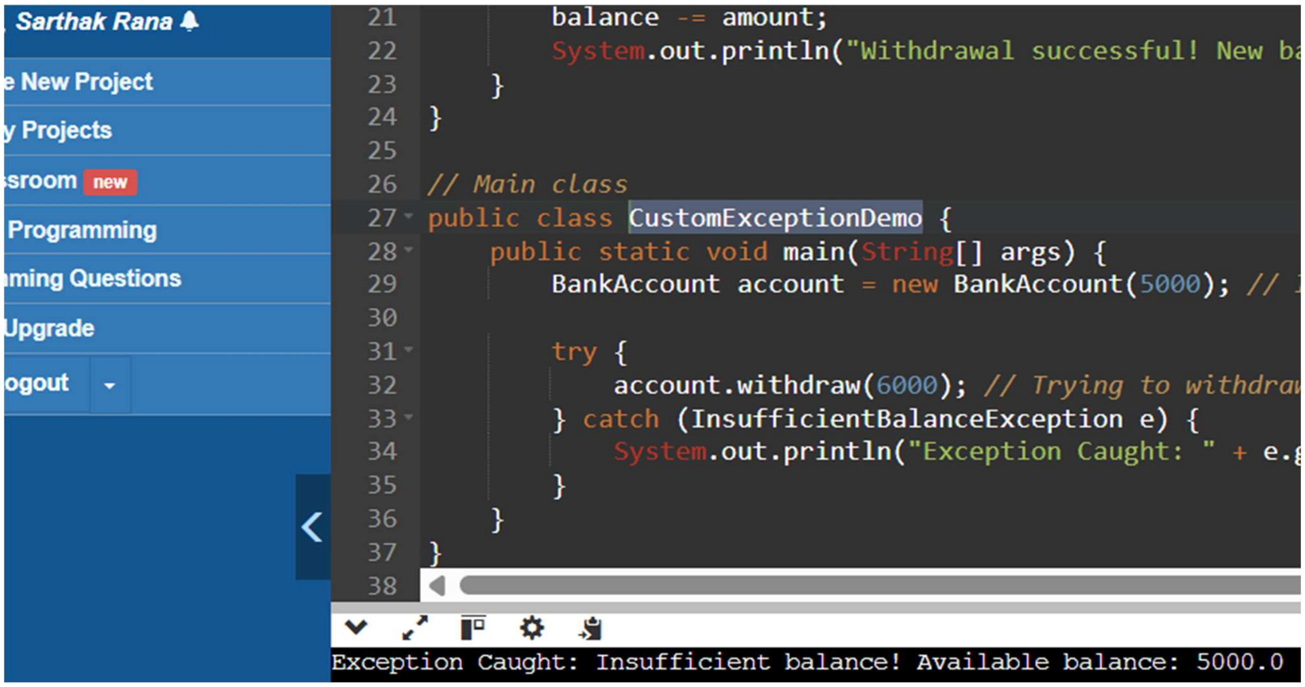
II.

```
import java.io.IOException;

class ThrowsExample {
    // Declaring that this method may throw an IOException
    public static void readFile() throws IOException {
        throw new IOException("File not found!"); // Throwing an exception
    }

    public static void main(String[] args) {
        try {
            readFile(); // Calling the method that throws an exception
        } catch (IOException e) {
            System.out.println("Exception Caught: " + e.getMessage());
        }
    }
}
```

OUTOUT:

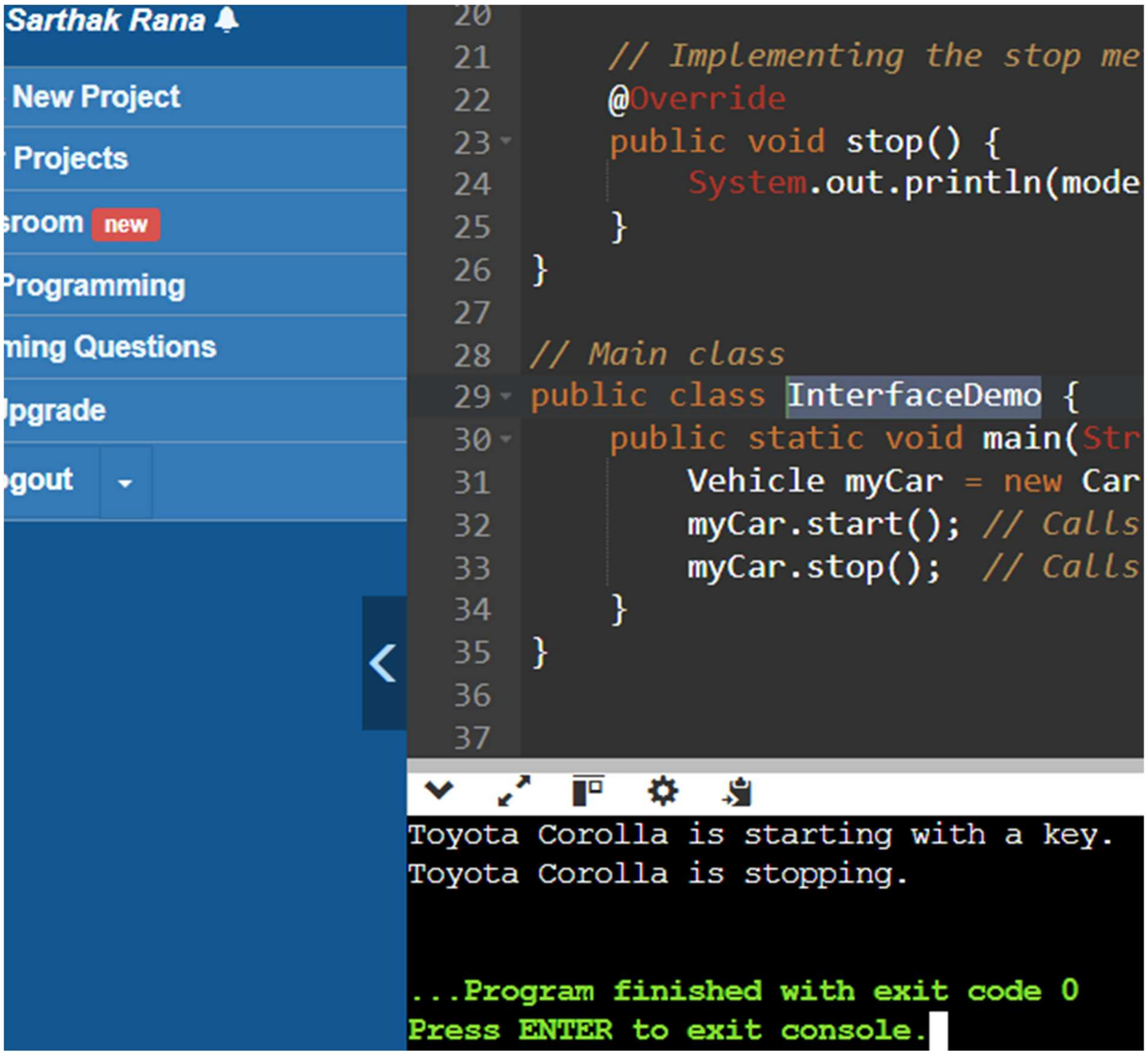


```
            balance -= amount;
            System.out.println("Withdrawal successful! New ba
        }
    }

// Main class
public class CustomExceptionDemo {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(5000); // I

        try {
            account.withdraw(6000); // Trying to withdraw
        } catch (InsufficientBalanceException e) {
            System.out.println("Exception Caught: " + e.g
        }
    }
}
```

```
Exception Caught: Insufficient balance! Available balance: 5000.0
```

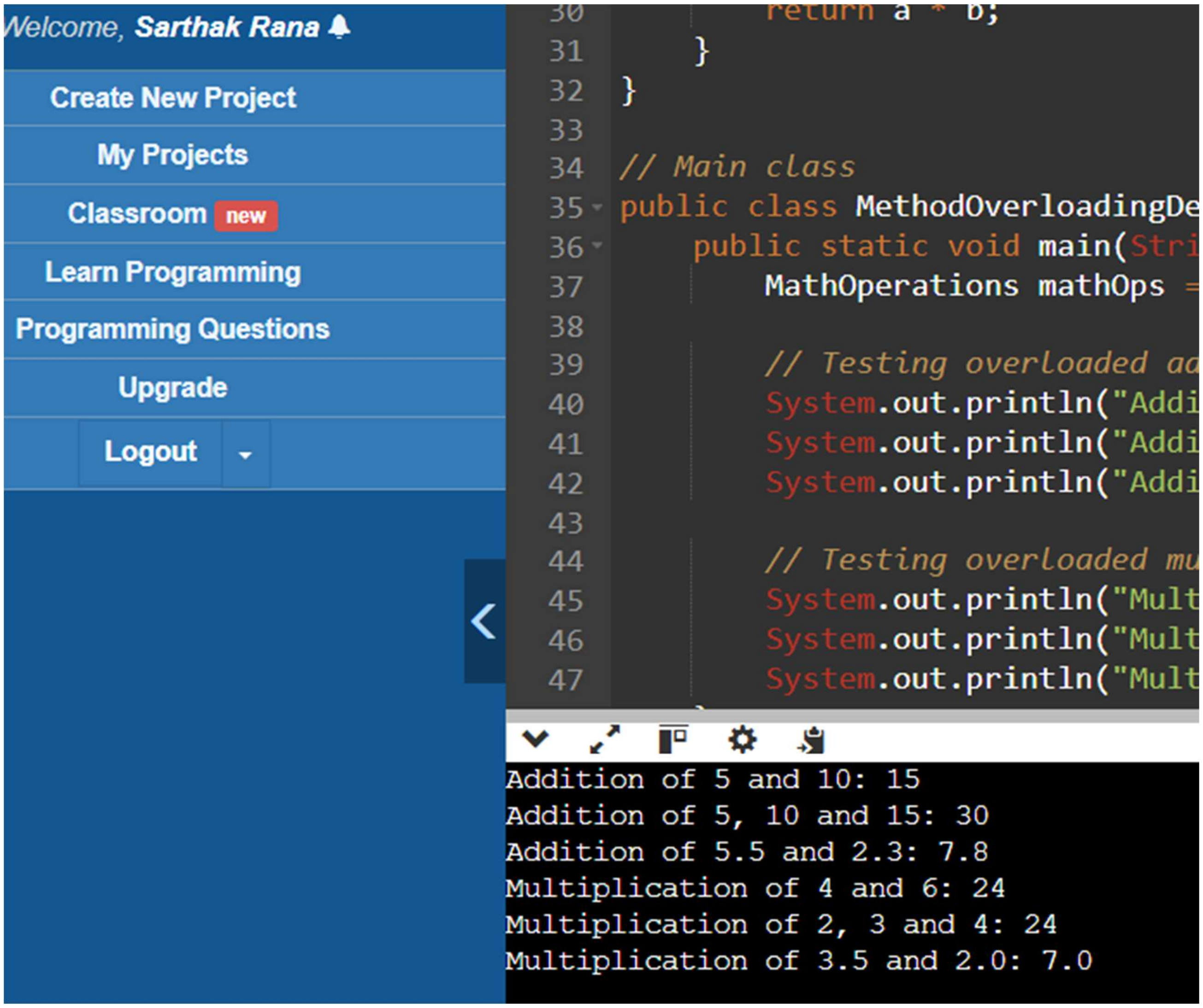


```
    // Implementing the stop me
    @Override
    public void stop() {
        System.out.println(mode
    }
}

// Main class
public class InterfaceDemo {
    public static void main(Str
        Vehicle myCar = new Car
        myCar.start(); // Calls
        myCar.stop();  // Calls
    }
}
```

```
Toyota Corolla is starting with a key.
Toyota Corolla is stopping.


...Program finished with exit code 0
Press ENTER to exit console.
```



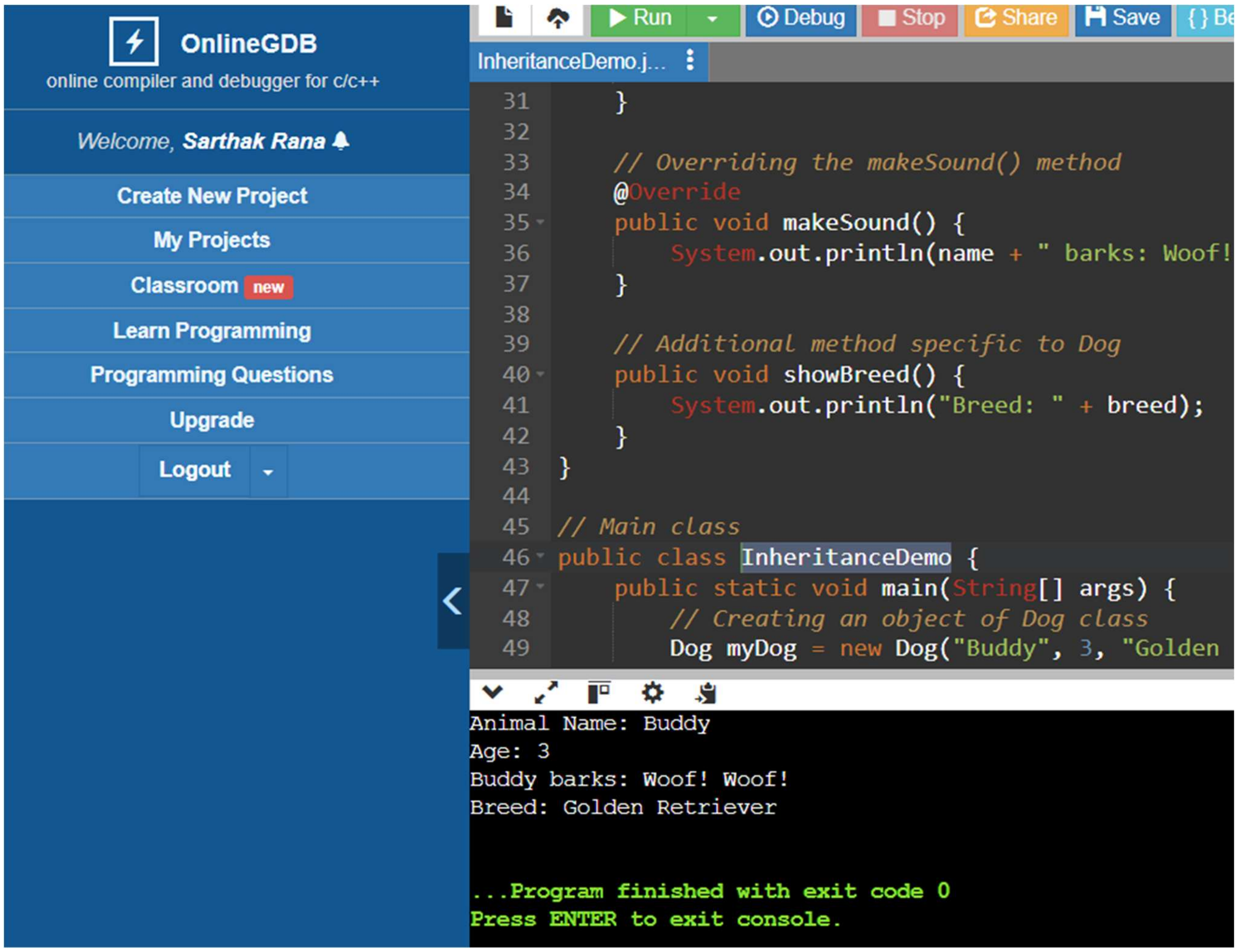
```
            return a * b;
        }
}

// Main class
public class MethodOverloadingDe
    public static void main(Stri
        MathOperations mathOps =

        // Testing overloaded ad
        System.out.println("Addi
        System.out.println("Addi
        System.out.println("Addi

        // Testing overloaded mu
        System.out.println("Mult
        System.out.println("Mult
        System.out.println("Mult
```

```
Addition of 5 and 10: 15
Addition of 5, 10 and 15: 30
Addition of 5.5 and 2.3: 7.8
Multiplication of 4 and 6: 24
Multiplication of 2, 3 and 4: 24
Multiplication of 3.5 and 2.0: 7.0
```



```
    }

    // Overriding the makeSound() method
    @Override
    public void makeSound() {
        System.out.println(name + " barks: Woof!
    }

    // Additional method specific to Dog
    public void showBreed() {
        System.out.println("Breed: " + breed);
    }
}

// Main class
public class InheritanceDemo {
    public static void main(String[] args) {
        // Creating an object of Dog class
        Dog myDog = new Dog("Buddy", 3, "Golden
```

```
Animal Name: Buddy
Age: 3
Buddy barks: Woof! Woof!
Breed: Golden Retriever


...Program finished with exit code 0
Press ENTER to exit console.
```