Southern University Bangladesh

Department of Computer Science and Engineering

Faculty of Science and Engineering

Object Oriented Programming Lab

Lab Report: 05

Course code: CSE 0613-108

Submitted by:

Name : Sharmistha Chowdhury

ID: 666-61-60

Submitted to :

Mohammed Arif Hasan Chowdhury, Assistant Professor

Dept. of Computer Science and Engineering

Date: 13/05/2024

Problem 01: Demonstrate Method overloading By Changing the Number of Parameters.

Objective:

The objective of the program is to demonstrate method overloading in Java by defining a Product class with two multiply methods that calculate and print the product of two or three integers.

Algorithm:

- **Step 1**: Import the necessary Java libraries. In this case, java.io.* is imported, but it's not used in the program.
- **Step 2**: Define a class named Product with two methods named multiply.
  - 2.1. The first multiply method takes two integer parameters a and b. It calculates the product of a and b and returns the result.
  - 2.2. The second multiply method takes three integer parameters a, b, and c. It calculates the product of a, b, and c and returns the result.
- **Step 3**: Define the main class Program1 with a main method.
- **Step 4**: Inside the main method, create an instance of the Product class named ob.
- **Step 5**: Call the multiply method of the Product instance ob with two arguments (1 and 2), store the result in prod1, and print the result.
- **Step 6**: Call the multiply method of the Product instance ob with three arguments (1, 2, and 3), store the result in prod2, and print the result.
- **Step 7**: The program ends after printing the results.

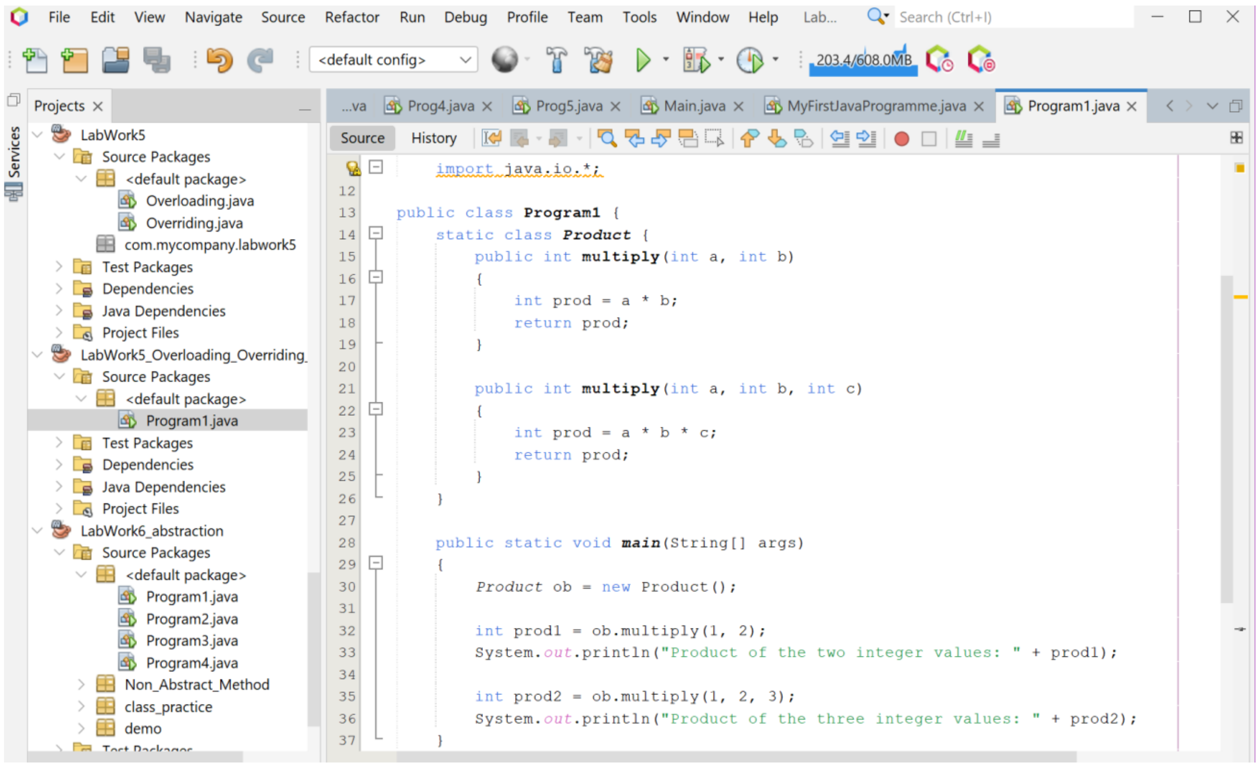
Code:

```
import java.io.*;

public class Program1 {
    static class Product {
        public int multiply(int a, int b)
        {
            int prod = a * b;
            return prod;
        }

        public int multiply(int a, int b, int c)
        {
            int prod = a * b * c;
            return prod;
        }
    }

    public static void main(String[] args)
    {
        Product ob = new Product();

        int prod1 = ob.multiply(1, 2);
        System.out.println("Product of the two integer values: " + prod1);

        int prod2 = ob.multiply(1, 2, 3);
        System.out.println("Product of the three integer values: " + prod2);
    }
}
```

Input/ Expected Output:

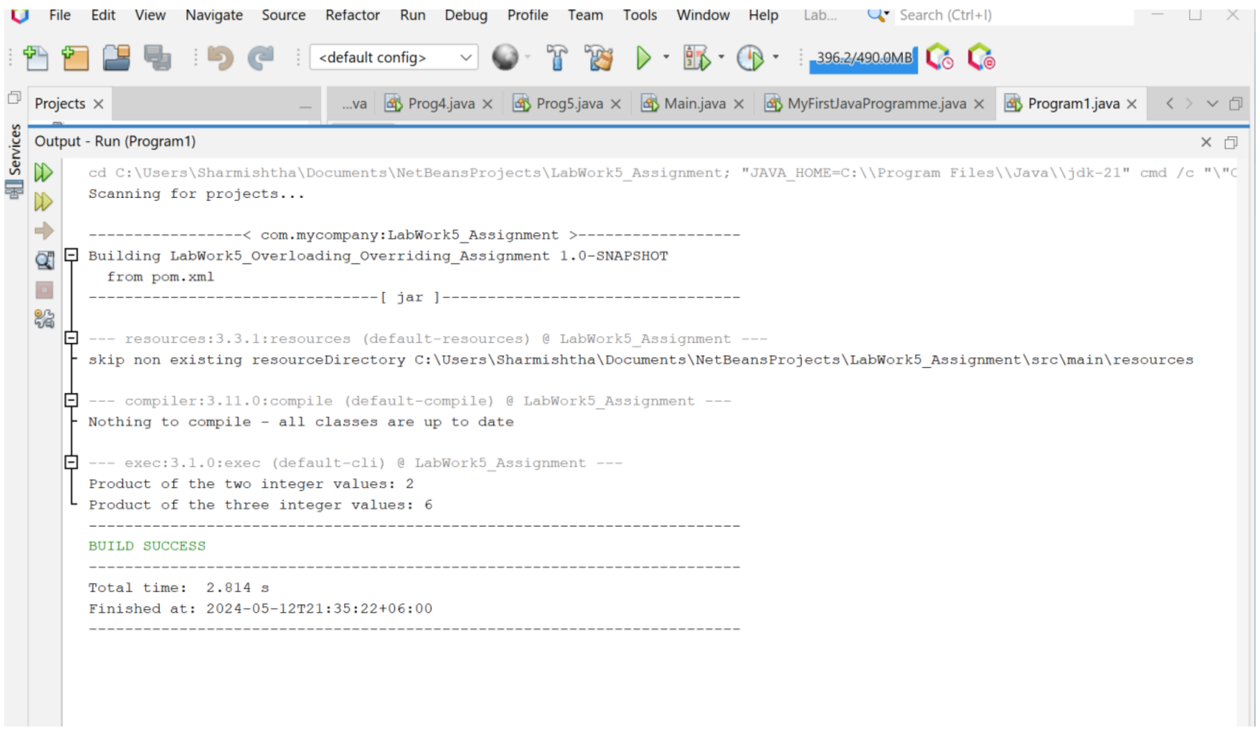
```
Product of the two integer values: 2
Product of the three integer values: 6
```

Screenshot of code edition window:



```java
    import java.io.*;

public class Program1 {
    static class Product {
        public int multiply(int a, int b)
        {
            int prod = a * b;
            return prod;
        }

        public int multiply(int a, int b, int c)
        {
            int prod = a * b * c;
            return prod;
        }
    }

    public static void main(String[] args)
    {
        Product ob = new Product();

        int prod1 = ob.multiply(1, 2);
        System.out.println("Product of the two integer values: " + prod1);

        int prod2 = ob.multiply(1, 2, 3);
        System.out.println("Product of the three integer values: " + prod2);
    }
```

Screenshot of Output screen/Run screen window:



```
cd C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork5_Assignment; "JAVA_HOME=C:\\Program Files\\Java\\jdk-21" cmd /c "\"C
Scanning for projects...

-----------------< com.mycompany:LabWork5_Assignment >-----------------
Building LabWork5_Overloading_Overriding_Assignment 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ LabWork5_Assignment ---
skip non existing resourceDirectory C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork5_Assignment\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ LabWork5_Assignment ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ LabWork5_Assignment ---
Product of the two integer values: 2
Product of the three integer values: 6
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.814 s
Finished at: 2024-05-12T21:35:22+06:00
------------------------------------------------------------------------
```

Explanation:

The program is a simple Java application that demonstrates the concept of method overloading, a feature in Java that allows a class to have more than one method having the same name, if their argument lists are different. It defines a Product class with two multiply methods, one for calculating the product of two integers and another for three integers. The Program1 class contains the main method where an instance of the Product class is created and both multiply methods are invoked with different sets of numbers.

Discussion

This program effectively models a banking system with different types of accounts (Savings and Current), each having unique properties and behaviors, and performs operations like calculating tax and interest. It demonstrates the concept of inheritance in Java, where subclasses inherit properties and behaviors from a superclass, and can also provide their own unique behaviors.

Problem 02: Demonstrate Method overloading Changing Data Types of the Parameters.

Objective:

The objective of the program is to demonstrate method overloading in Java by defining a Product class with two Prod methods that calculate and print the product of three integers or doubles.

Algorithm:

1. **Step 1**: Import the necessary Java libraries. In this case, java.io.* is imported, but it's not used in the program.
2. **Step 2**: Define a class named Product with two methods named Prod.
    - 2.1. The first Prod method takes three integer parameters a, b, and c. It calculates the product of a, b, and c and returns the result.
    - 2.2. The second Prod method takes three double parameters a, b, and c. It calculates the product of a, b, and c and returns the result.
3. **Step 3**: Define the main class Program2 with a main method.
4. **Step 4**: Inside the main method, create an instance of the Product class named obj.
5. **Step 5**: Call the Prod method of the Product instance obj with three integer arguments (1, 2, and 3), store the result in prod1, and print the result.
6. **Step 6**: Call the Prod method of the Product instance obj with three double arguments (1.0, 2.0, and 3.0), store the result in prod2, and print the result.
7. **Step 7**: The program ends after printing the results.

Code:

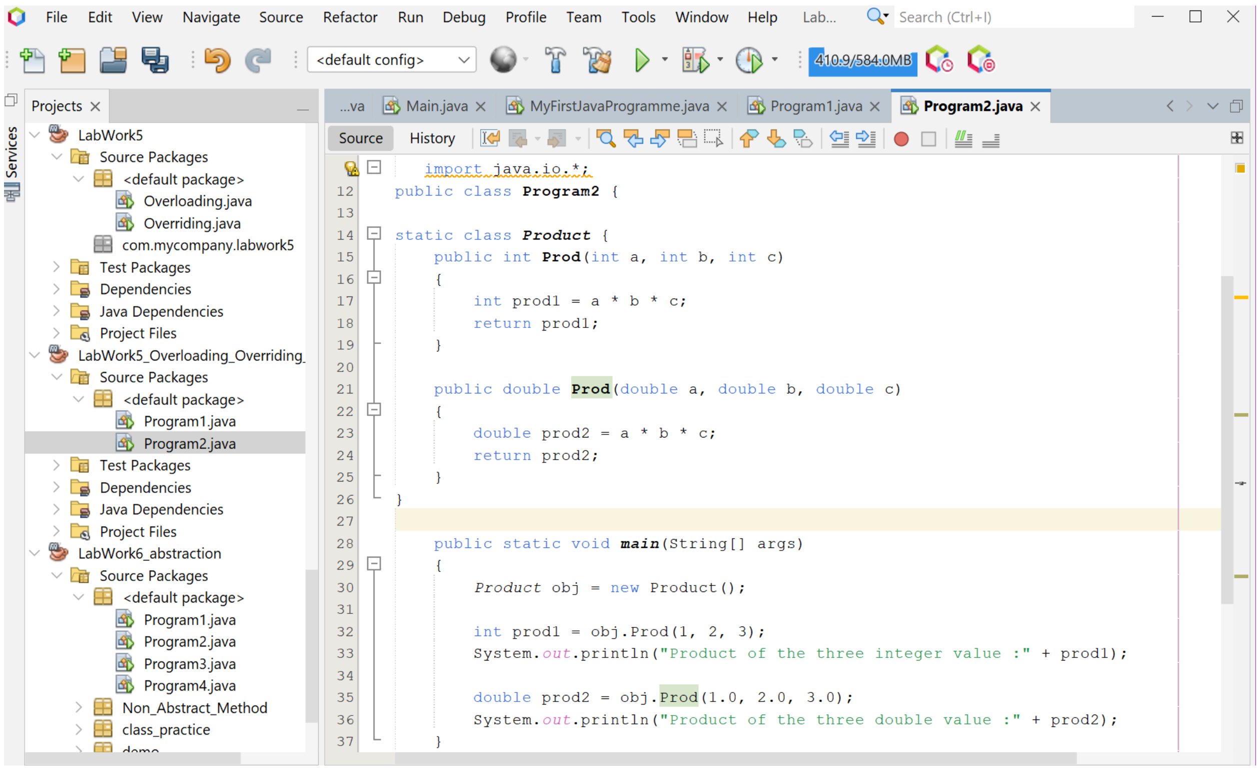
```
import java.io.*;

public class Program2 {


static class Product {

  public int Prod(int a, int b, int c)

  {

    int prod1 = a * b * c;

    return prod1;

  }
```

```
    public double Prod(double a, double b, double c)
    {
        double prod2 = a * b * c;
        return prod2;
    }
}


    public static void main(String[] args)
    {
        Product obj = new Product();

        int prod1 = obj.Prod(1, 2, 3);
        System.out.println("Product of the three integer value :" + prod1);

        double prod2 = obj.Prod(1.0, 2.0, 3.0);
        System.out.println("Product of the three double value :" + prod2);
    }
}
```

Input/ Expected Output:

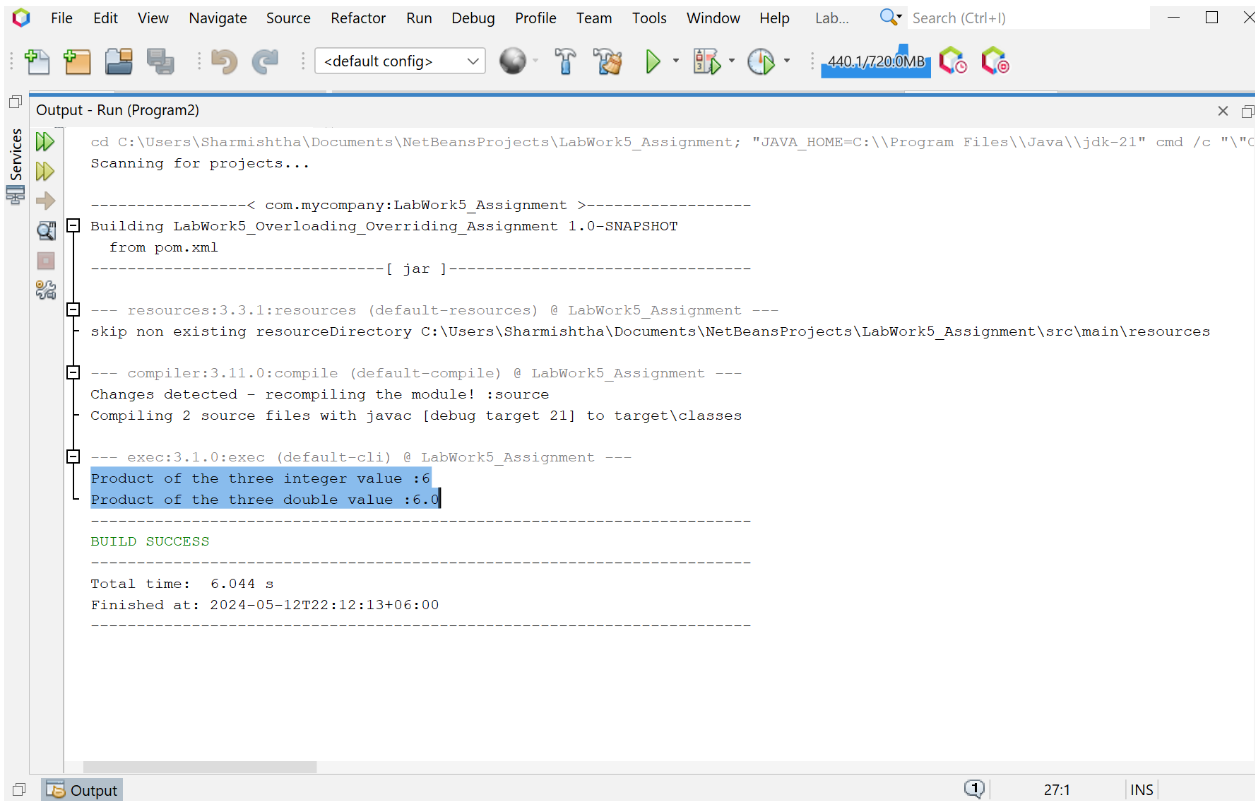
```
Product of the three integer value :6
Product of the three double value :6.0
```

Screenshot of code edition window:



```java
    import java.io.*;
public class Program2 {

static class Product {
    public int Prod(int a, int b, int c)
    {
        int prod1 = a * b * c;
        return prod1;
    }

    public double Prod(double a, double b, double c)
    {
        double prod2 = a * b * c;
        return prod2;
    }
}

    public static void main(String[] args)
    {
        Product obj = new Product();

        int prod1 = obj.Prod(1, 2, 3);
        System.out.println("Product of the three integer value :" + prod1);

        double prod2 = obj.Prod(1.0, 2.0, 3.0);
        System.out.println("Product of the three double value :" + prod2);
    }
```

Screenshot of Output screen/Run screen window:



```
cd C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork5_Assignment; "JAVA_HOME=C:\\Program Files\\Java\\jdk-21" cmd /c "\"C
Scanning for projects...

-----------------< com.mycompany:LabWork5_Assignment >------------------
Building LabWork5_Overloading_Overriding_Assignment 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ LabWork5_Assignment ---
skip non existing resourceDirectory C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork5_Assignment\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ LabWork5_Assignment ---
Changes detected - recompiling the module! :source
Compiling 2 source files with javac [debug target 21] to target\classes

--- exec:3.1.0:exec (default-cli) @ LabWork5_Assignment ---
Product of the three integer value :6
Product of the three double value :6.0
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  6.044 s
Finished at: 2024-05-12T22:12:13+06:00
------------------------------------------------------------------------
```

Explanation :

The program is a Java application that demonstrates the concept of method overloading, which allows a class to have more than one method having the same name, if their argument lists are different. It defines a Product class with two Prod methods, one for calculating the product of three integers and another for three doubles. The Program2 class contains the main method where an instance of the Product class is created and both Prod methods are invoked with the numbers 1, 2, and 3 for both integer and double types.

Discussion:

Method overloading is a key aspect of Java, allowing for flexibility and increased readability in code. In this program, the Prod methods in the Product class are overloaded to handle different types of arguments, demonstrating how the correct method is chosen at compile time based on the type of arguments. This is a practical example of compile-time polymorphism. However, it’s important to note that method overloading should be used judiciously as it can lead to confusion if the methods perform vastly different tasks. In this case, all Prod methods are logically related as they all perform a multiplication operation, making it a good use of method overloading

Problem 03: Demonstrate method overriding using inheritance

Objective:

The program's goal is to demonstrate the use of Java inheritance and method overriding by performing addition operations.

Algorithm:

Step 1: Define a class named Grandpa with a method show() that prints a message indicating it's inside the show() method of the Grandpa class.
Step 2: Define a subclass named Dad that extends Grandpa. Override the show() method in Dad to print a message indicating it's inside the show() method of the Dad class.
Step 3: Define another subclass named Me that extends Dad. Override the show() method in Me to print a message indicating it's inside the show() method of the Me class.
Step 4: Define the main class Program3 with a main method.
Step 5: Inside the main method, create an instance of each class (Grandpa, Dad, and Me).
Step 6: Call the show() method on each instance.
Step 7: The program ends after printing the messages from each show() method.

Code:

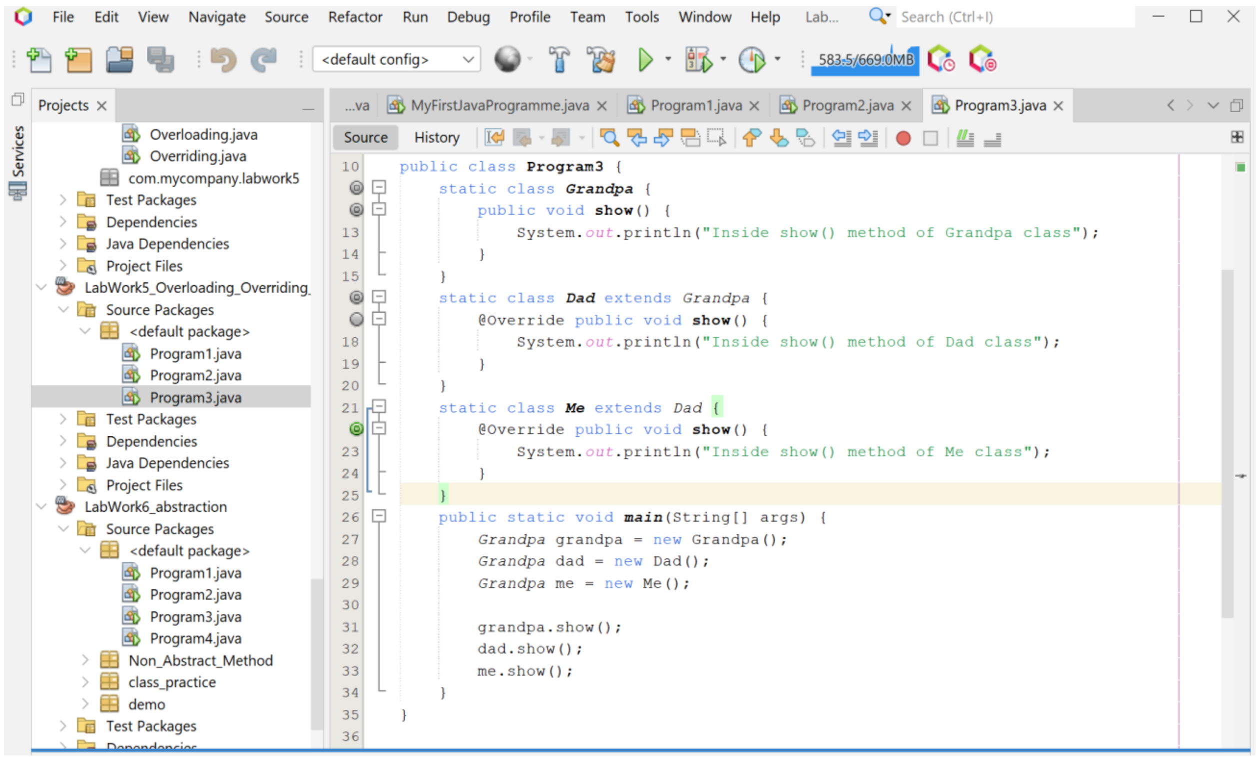
```
public class Program3 {
  static class Grandpa {
    public void show() {
      System.out.println("Inside show() method of Grandpa class");
    }
  }
  static class Dad extends Grandpa {
    @Override public void show() {
      System.out.println("Inside show() method of Dad class");
    }
  }
  static class Me extends Dad {
    @Override public void show() {
      System.out.println("Inside show() method of Me class");
    }
  }
```

```
    public static void main(String[] args) {
        Grandpa grandpa = new Grandpa();
        Grandpa dad = new Dad();
        Grandpa me = new Me();

        grandpa.show();
        dad.show();
        me.show();
    }
}
```

Input/ Expected Output:

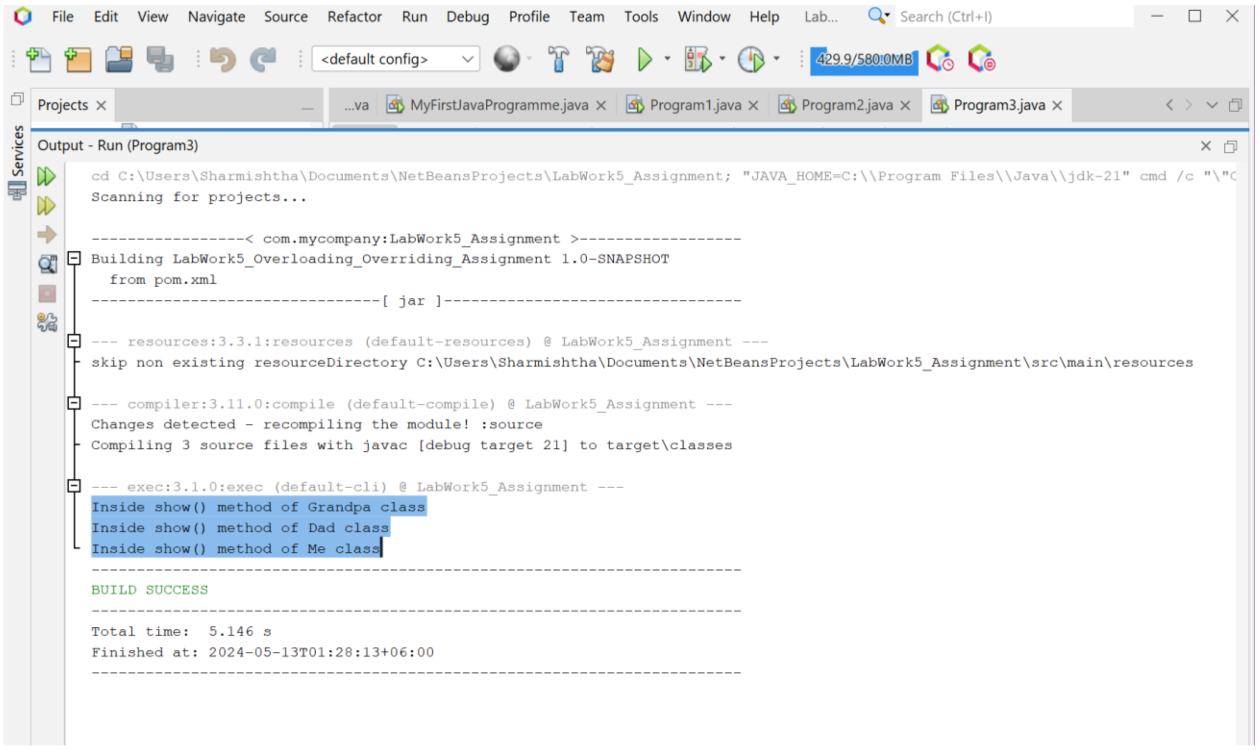
```
Inside show() method of Grandpa class
Inside show() method of Dad class
Inside show() method of Me class
```

Screenshot of code edition window:



```java
public class Program3 {
    static class Grandpa {
        public void show() {
            System.out.println("Inside show() method of Grandpa class");
        }
    }
    static class Dad extends Grandpa {
        @Override public void show() {
            System.out.println("Inside show() method of Dad class");
        }
    }
    static class Me extends Dad {
        @Override public void show() {
            System.out.println("Inside show() method of Me class");
        }
    }
    public static void main(String[] args) {
        Grandpa grandpa = new Grandpa();
        Grandpa dad = new Dad();
        Grandpa me = new Me();

        grandpa.show();
        dad.show();
        me.show();
    }
}
```

Screenshot of Output screen/Run screen window:



```
cd C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork5_Assignment; "JAVA_HOME=C:\\Program Files\\Java\\jdk-21" cmd /c "\"C
Scanning for projects...

----------------< com.mycompany:LabWork5_Assignment >-----------------
Building LabWork5_Overloading_Overriding_Assignment 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ LabWork5_Assignment ---
skip non existing resourceDirectory C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork5_Assignment\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ LabWork5_Assignment ---
Changes detected - recompiling the module! :source
Compiling 3 source files with javac [debug target 21] to target\classes

--- exec:3.1.0:exec (default-cli) @ LabWork5_Assignment ---
Inside show() method of Grandpa class
Inside show() method of Dad class
Inside show() method of Me class
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  5.146 s
Finished at: 2024-05-13T01:28:13+06:00
------------------------------------------------------------------------
```

Explanation :

The Program3 class in the Java application demonstrates the concept of method overriding and inheritance. It defines a hierarchy of classes (Grandpa, Dad, and Me) where each class overrides the show method to print a unique message. The main method in Program3 creates an instance of each class and invokes the show method on each instance. The output of the program is the messages printed by the show method of each class, demonstrating how Java determines which method to execute based on the actual object type.

Discussion:

Method overriding is a key aspect of Java, allowing for flexibility and increased readability in code. In this program, the show methods in the Grandpa, Dad, and Me classes are overridden to handle different messages, demonstrating how the correct method is chosen at runtime based on the actual object type. This is a practical example of runtime polymorphism. However, it’s important to note that method overriding should be used judiciously as it can lead to confusion if the methods perform vastly different tasks. In this case, all show methods are logically related as they all perform a similar operation (printing a message), making it a good use of method overriding.