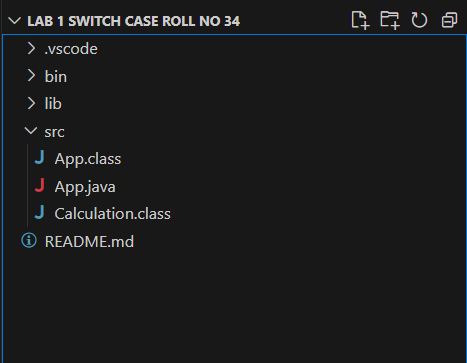
1. Write a JAVA program to calculate: Add, Subtract , Multiply and Divide using Switch Case.



***INPUT:***

import java.util.Scanner;

public class App {

public static void main(String[] args) throws Exception {

Scanner sc = new Scanner(System.in);

System.out.println("Enter first digit:");

int a = sc.nextInt();

System.out.println("Enter second digit:");

int b = sc.nextInt();

System.out.println("Enter 1 for Add:");

System.out.println("Enter 2 for Sub:");

System.out.println("Enter 3 for Mul:");

System.out.println("Enter 4 for Div:");

int option = sc.nextInt();

Calculation obj = new Calculation();

switch (option) {

case 1:

int c = obj.Add(a, b);

System.out.println(c);

break;

case 2:

int d = obj.Sub(a, b);

System.out.println(d);

break;

case 3:

int e = obj.Mul(a, b);

System.out.println(e);

break;

case 4:

int f = obj.Div(a, b);

System.out.println(f);

break;

}

sc.close(); }

}

class Calculation {

public int Add(int a, int b)

{

return a + b;

}

public int Sub(int a, int b)

{

return a - b;

}

public int Mul(int a, int b)

{

return a \* b;

}

public int Div(int a, int b)

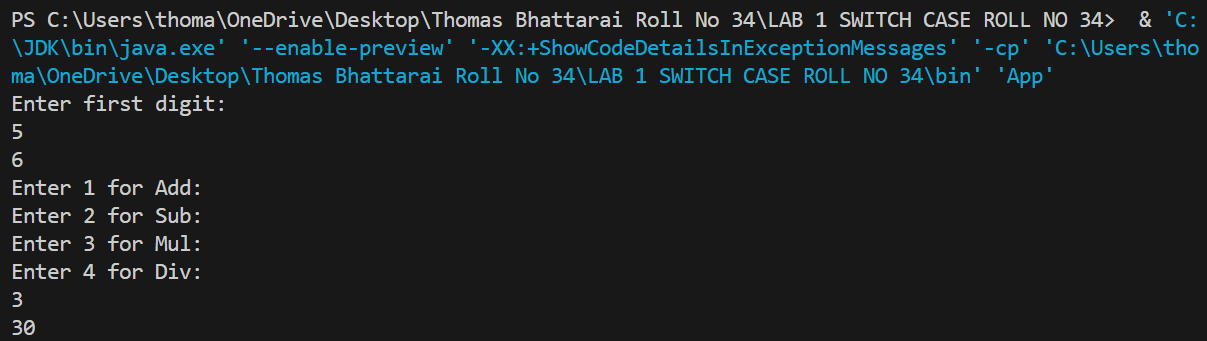
{

return a / b;

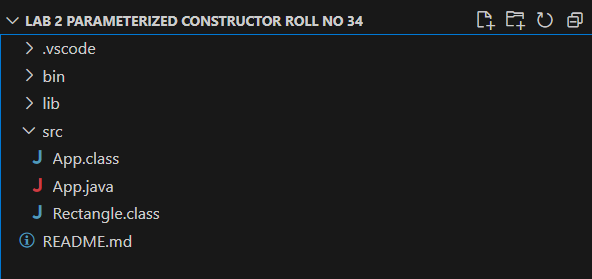
}

}

***Output:***



2. Write a JAVA program to calculate Add two digit using parameterized constructor.



***INPUT:***

import java.util.Scanner;

public class App {

public static void main(String[] args) throws Exception {

Scanner sc=new Scanner(System.in);

System.out.println("Enter first digit :");

int a=sc.nextInt();

System.out.println("Enter second digit :");

int b=sc.nextInt();

Rectangle rect=new Rectangle(a,b);

rect.Display();

sc.close();

}

}

class Rectangle

{

int first=0;

int second=0;

public Rectangle(int a,int b)

{

first=a;

second=b;

}

public void Display()

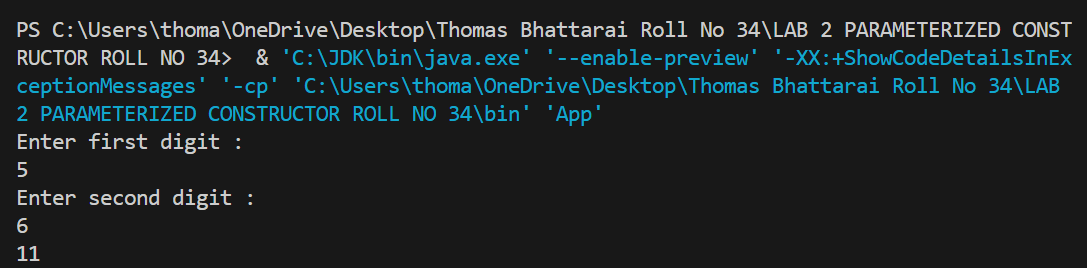
{

System.out.println(first+second);

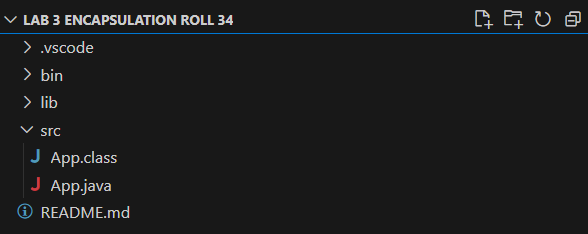
}

}

***Output:***



3. Write a Java program to achieve encapsulation using private access modifier.



***INPUT:***

public class App {

private int length;

private int breadth;

public App(int l, int b)

{

this.length = l;

this.breadth = b;

}

public void Area() {

System.out.println(length \* breadth);

}

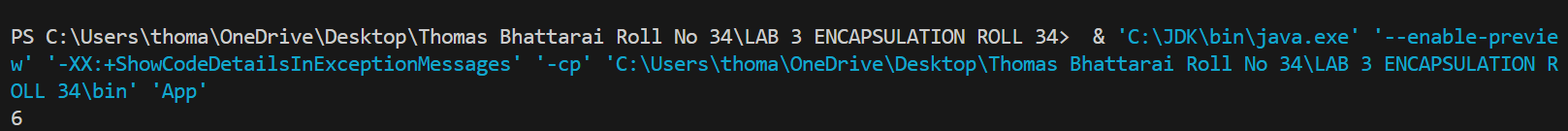
public static void main(String[] args) {

App ap = new App(2, 3);

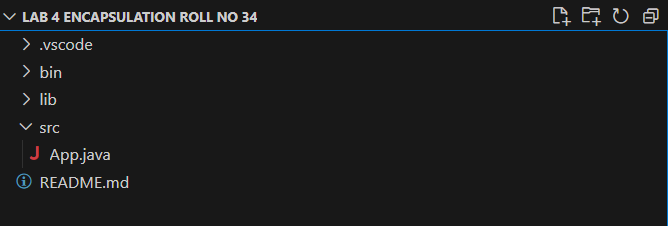
ap.Area();

}

}

***OUTPUT :***

4.Write a program to write encapsulation using getter and setter.



***INPUT:***

public class App {

public static void main(String[] args) throws Exception {

Rectangle rect=new Rectangle();

rect.setLength(5);

rect.setBreadth(5);

System.out.println(rect.getLength()\*rect.getBreadth());

}

}

class Rectangle

{

private int length;

private int breadth;

public int getLength(){

return length;

}

public void setLength(int length) {

this.length = length;

}

public int getBreadth() {

return breadth;

}

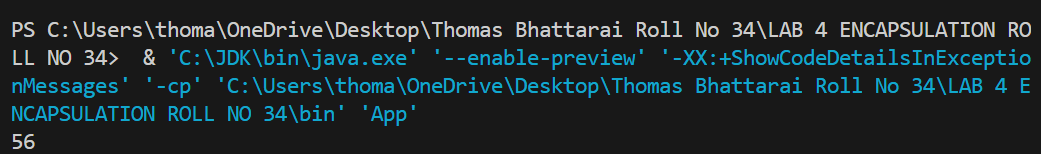
public void setBreadth(int breadth) {

this.breadth = breadth;

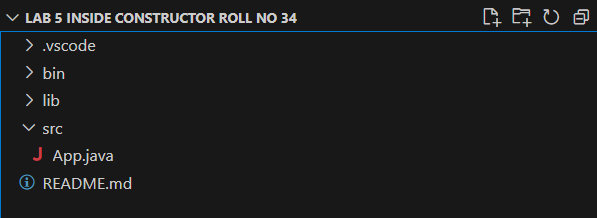
}

}

***OUTPUT:***



**5.Write a Java Program to demonstrate “this” keyword.**



***INPUT:***

public class App {

int num = 10;

public App() {

System.out.println("Inside constructor");

}

public App(int num) {

this();

this.num = num;

}

void display() {

this.show();

System.out.println(" num: "+ this.num);

}

void show() {

System.out.println("Inside show method");

}

public static void main(String[] args){

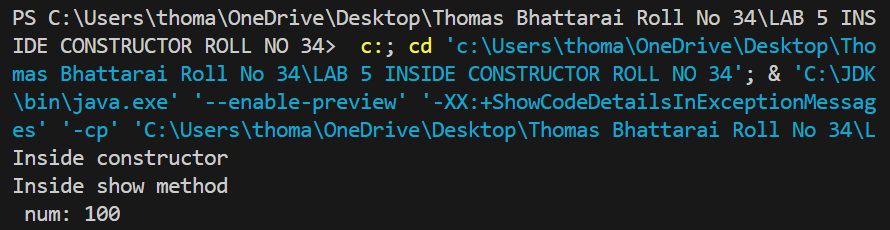
App obj = new App(100);

obj.display();

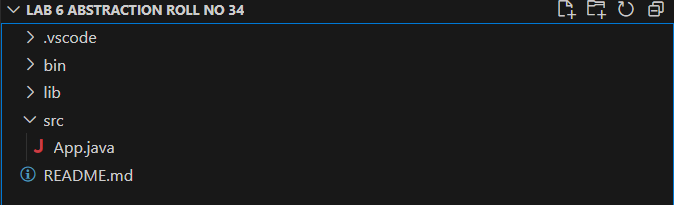
}

}

***OUTPUT:***



**6.Write a Java Program to achieve abstraction using abstract class.**



***INPUT:***

public class AbstractionExample {

public static void main(String[] args) throws Exception {

Shape shape = new Circle();

shape.draw(); // Output: Drawing Circle

}

}

abstract class Shape {

abstract void draw(); // Abstract method

}

class Circle extends Shape {

void draw() {

System.out.println("Drawing Circle");

}

}

class Rectangle extends Shape {

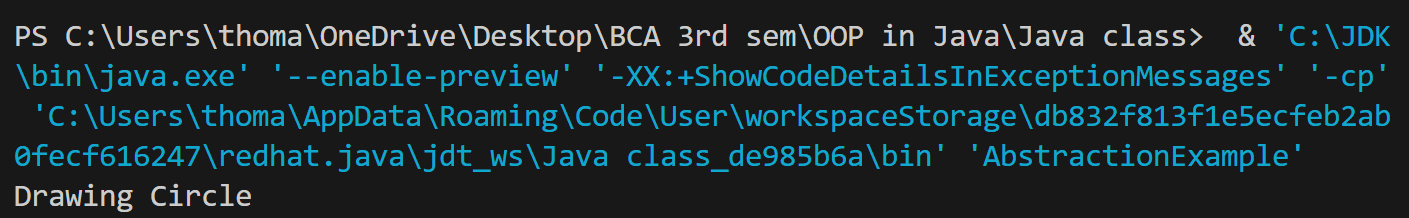
void draw() {

System.out.println("Drawing Rectangle");

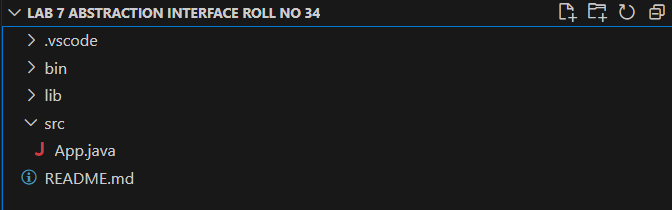
}

}

***OUTPUT:***

******

**7.Write a JAVA program to achieve abastraction using Interface or achieve 100% abstraction.**

****

**INPUT:**

public class App {

public static void main(String[] args) throws Exception {

Circle circle = new Circle();

circle.draw();

}

}

interface Drawable {

void draw();

}

class Circle implements Drawable {

public void draw() {

System.out.println("Drawing Circle");

}

}

class Rectangle implements Drawable {

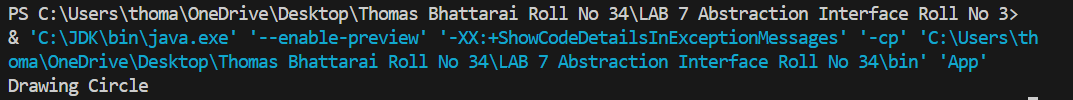
public void draw() {

System.out.println("Drawing Rectangle");

}

}

**OUTPUT:**

****