                 USN : 2GI19CS175                                       Student Name : Venkatesh G Dhongadi

**Title of the Experiment ( Abstract Classes )**

**Experiment  No.**  \_\_\_\_06\_\_\_\_\_\_                                                     **Date :** \_\_24/12/20\_\_\_\_\_

**Problem Statement:**

Design an abstract class Car to have carName, chassiNum, modelName as member

variables and add two abstract methods, startCar and operateSteering . Inherit MarutiCar and

BmwCar from Car class and override the two abstract methods in their own unique way.

Design a driver class to have driver name, gender and age as data members and add a method

driveCar with abstract class reference variable as argument and invoke the two basic

operations namely, startCar and operateStearing and demonstrate run-time polymorphism.

**Objectives of the Experiment:**

1. Learn declaration and initialization of variables and implementation of Run-time Polymorphism in Java.

2. Understand the use of Run-time Polymorphism in a real-life application.

3. Learn the usage of Looping constructs and control statements.

4. Learn to Display the result in a readable/proper format.

**Problem Source Code:**

**package** termwork\_6;

**publicclass** termwork\_6 {

**publicstaticvoid** main(String[] args) {

MarutiCar c=**new** MarutiCar(22,"Maruti","Maruti 800",10001023);

BMWCar c1=**new** BMWCar(125,"BMW-AS 6","BMW",1233422);

Driver d=**new** Driver("Ajay","Male",24);

d.driveCar(c);

d.driveCar(c1);

}

}

**abstractclass** Car {

String carName, model;

**long** chassNumber;

**public** Car(String carName, String model, **long** chassNumber) {

**this**.carName = carName;

**this**.model = model;

**this**.chassNumber = chassNumber;

}

**publicabstractvoid** startCar();

**publicabstractvoid** operateStearing();

}

**class** MarutiCar **extends** Car {

**float** mileage;

**public** MarutiCar(**float** mileage, String carName, String model, **long** chassNumber) {

**super**(carName, model, chassNumber);

**this**.mileage = mileage;

}

**publicvoid** startCar(){ System.***out***.println("Maruti Car Started -Key Start......"); }

**publicvoid** operateStearing() { System.***out***.println("Its manualstearing.... Its very Hard..."); }

}

**class** BMWCar **extends** Car{

**float** horsePower;

**public** BMWCar(**float** horsePower, String carName, String model, **long** chassNumber) {

**super**(carName, model, chassNumber);

**this**.horsePower = horsePower;

}

**publicvoid** startCar(){ System.***out***.println("BMW Car started -Touch start...."); }

**publicvoid** operateStearing(){ System.***out***.println("Its powerstearing and is very Smooth..."); }

}

**class** Driver {

String name, gender;

**int** age;

**public** Driver(String name, String gender, **int** age) {

**this**.name = name;

**this**.gender = gender;

**this**.age = age;

}

**publicvoid** driveCar(Car c){

System.***out***.println(name + " is driving "+c.carName+" car ");

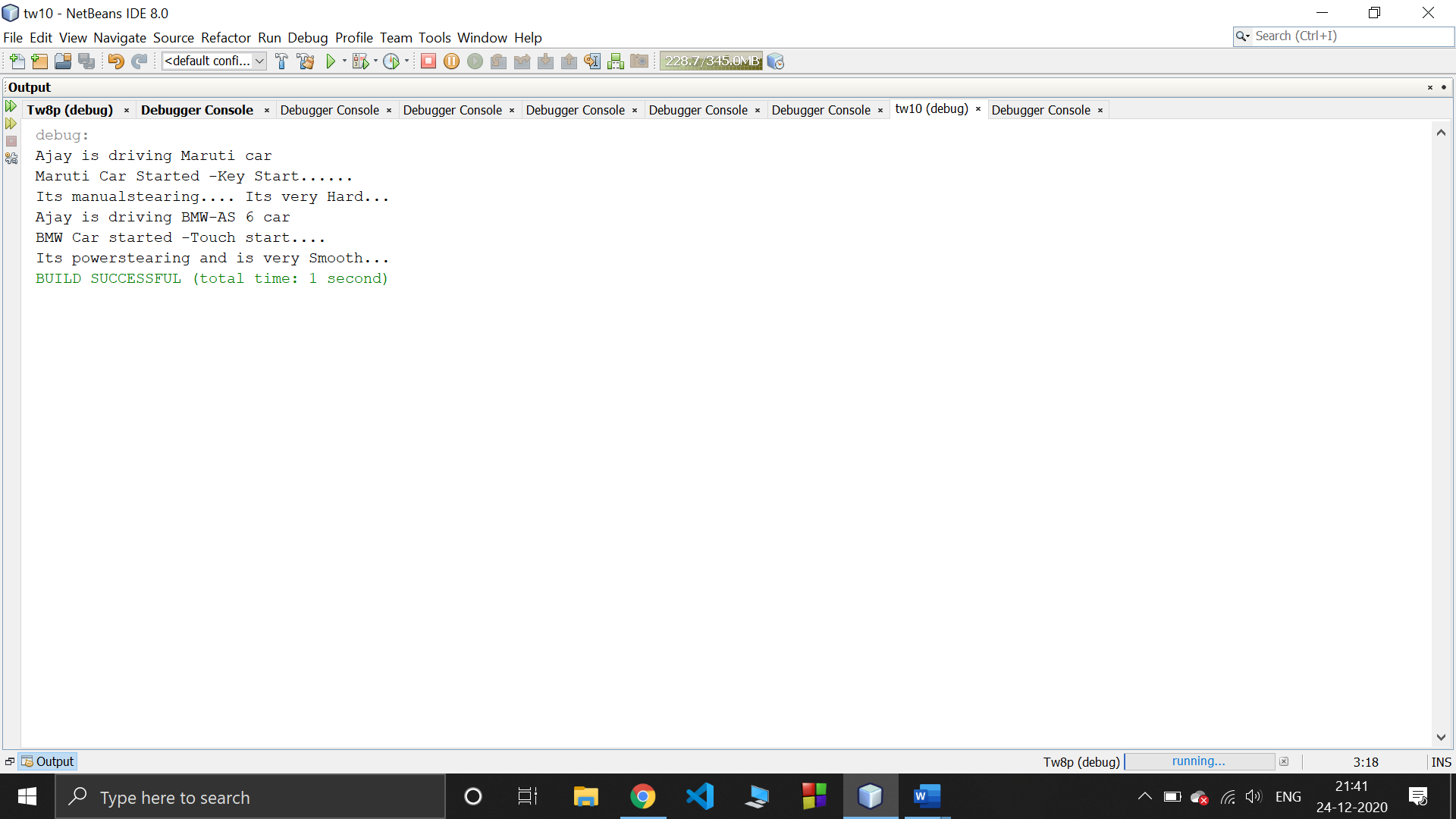
c.startCar();

c.operateStearing();

}

}

**Output:**

****

**Outcomes of the Experiment:**

1. Able to Demonstrate the use of Run-time Polymorphism in solving real-life problems.

2. Identify appropriate variables and their types

3. Identify appropriate looping constructs (for)

4. Check if one loop will suffice or use nesting

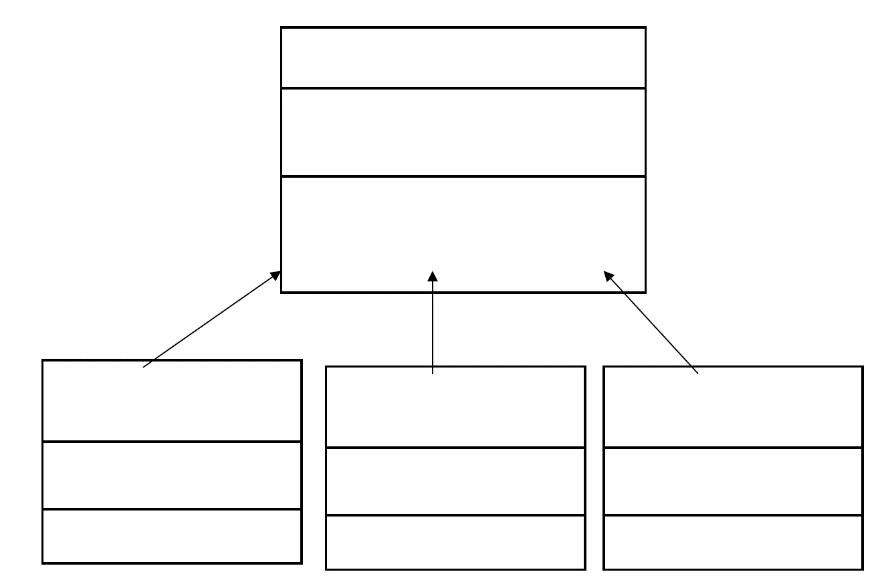
5. Identify the control statements needed to meet the problem requirements.

**Conclusion:**

From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in Eclipse IDE by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of built-in class System and its method println to display the result. The program was executed for two sets of input and result obtained were verified to be correct and recorded.

**Practice Problem Statement:**

Implement the following inheritance hierarchy.



**Problem Source Code:**

**package** termwork\_6\_pp1;

**import** java.util.Scanner;

**abstractclass** Shape{

**double** area;

**double** perimeter;

String type;

**abstractvoid** computeArea();

**abstractvoid** computePerimeter();

}

**class** Rectangle **extends** Shape{

**double** length, width;

Rectangle(){

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("Enter the length and width: ");

length = in.nextDouble();

width = in.nextDouble();

}

**void** computeArea() {

area = length \* width;

System.***out***.println("Area of the rectangle is: " + area);

}

**void** computePerimeter() {

perimeter = 2 \* (length + width);

System.***out***.println("Perimeter of the rectangle is: " + perimeter);

}

}

**class** Circle **extends** Shape{

**double** radius;

Circle(){

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("Enter the radius: ");

radius = in.nextDouble();

}

**void** computeArea() {

area = Math.***PI*** \* radius \* radius;

System.***out***.println("Area of the circle is: " + area);

}

**void** computePerimeter() {

perimeter = 2 \* Math.***PI*** \* radius;

System.***out***.println("Perimeter of the circle is: " + perimeter);

}

}

**class** Triangle **extends** Shape{

**double** a, b, c, s;

Triangle(){

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("Enter the 3 sides: ");

a = in.nextDouble();

b = in.nextDouble();

c = in.nextDouble();

}

**void** computeArea() {

s = (a + b + c) / 2.0;

area = Math.*sqrt*(s \* (s-a) \* (s-b) \* (s-c));

System.***out***.println("Area of the triangle is: " + area);

}

**void** computePerimeter() {

perimeter = a + b + c;

System.***out***.println("Perimeter of the triangle is: " + perimeter);

}

}

**publicclass** Tw6\_pp1 {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

Rectangle r = **new** Rectangle();

r.computeArea();

r.computePerimeter();

Circle c = **new** Circle();

c.computeArea();

c.computePerimeter();

Triangle t = **new** Triangle();

t.computeArea();

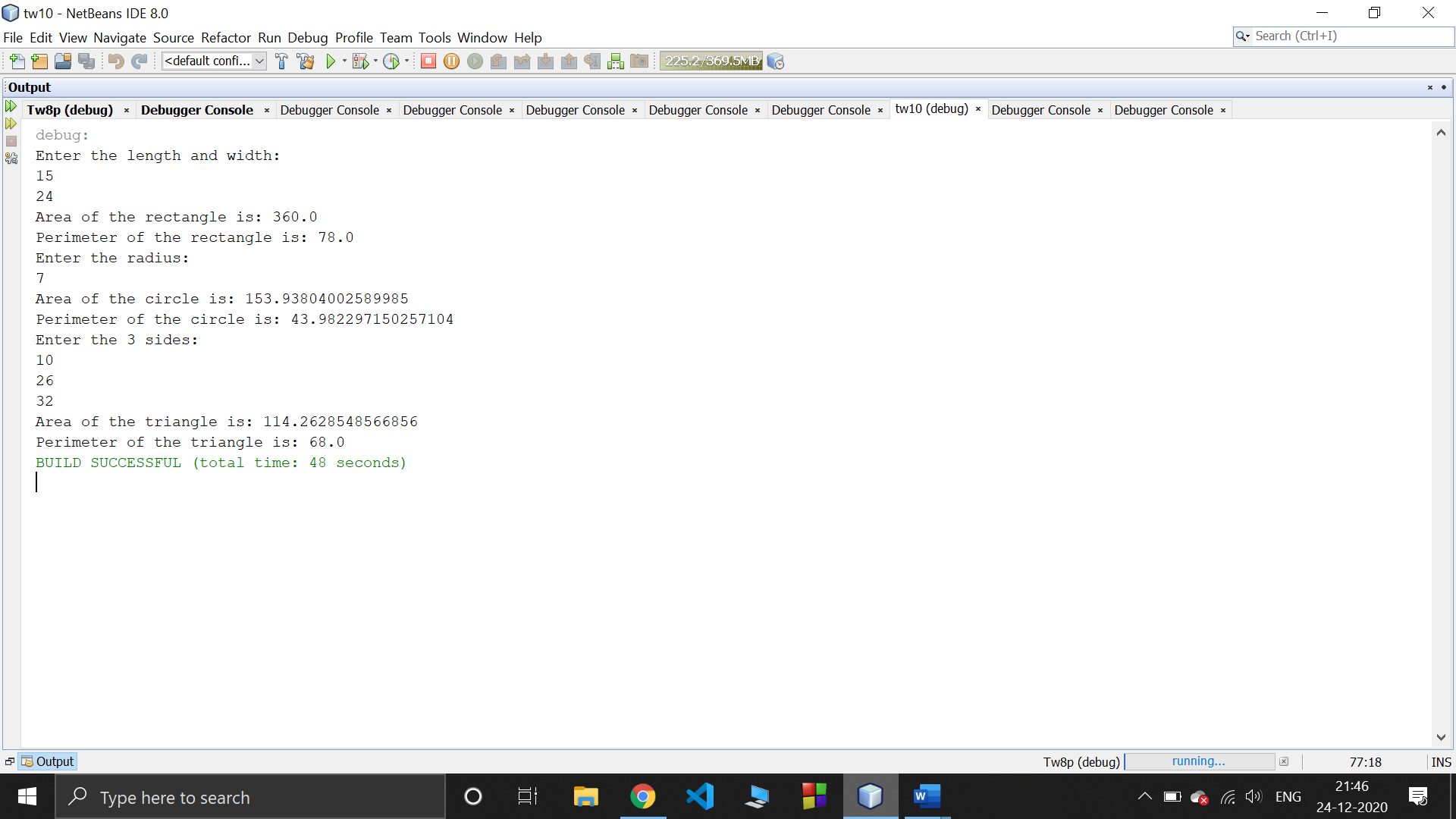
t.computePerimeter();

}

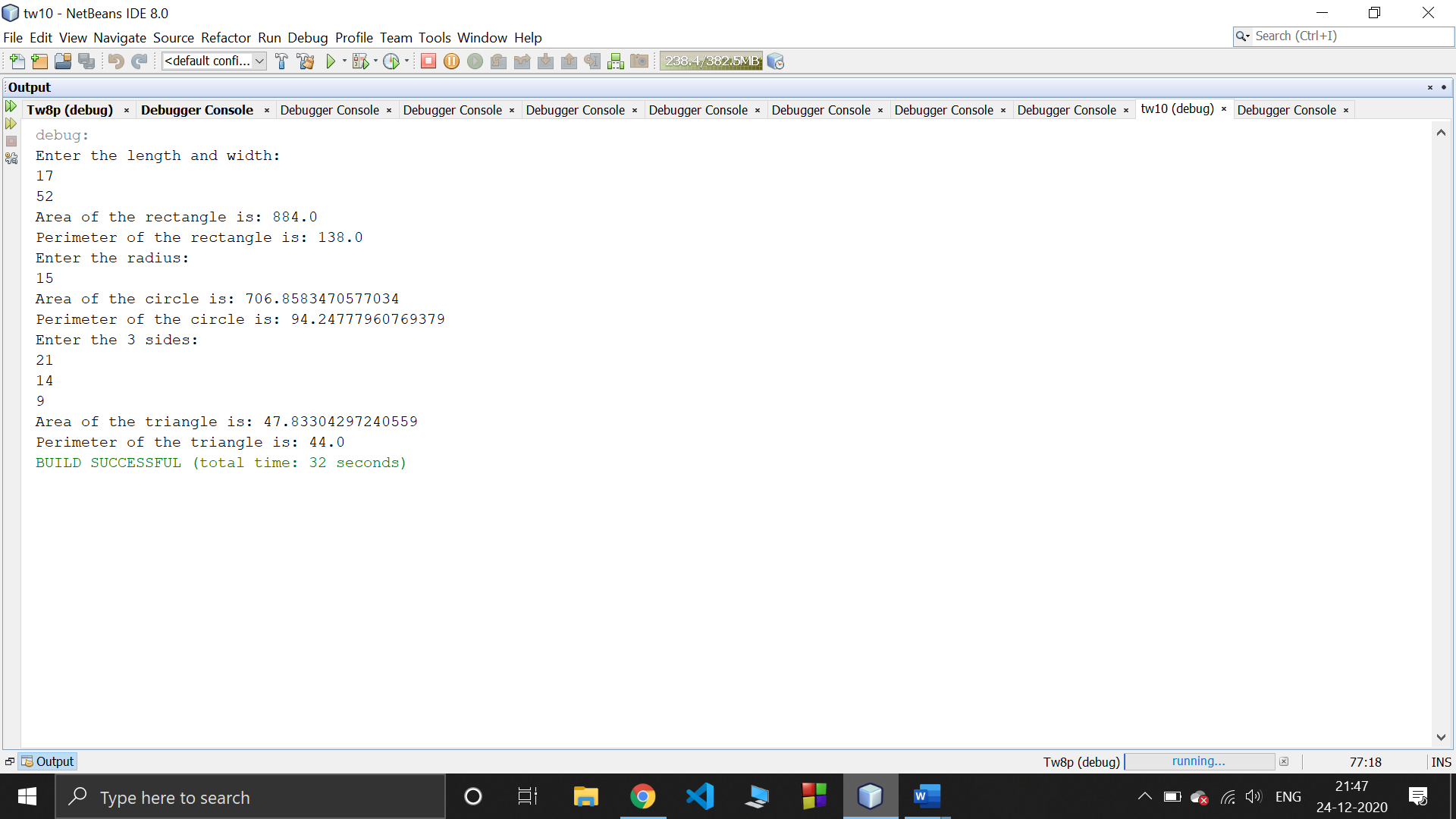
}

**Output:**

**Case 1:**

****

**Case2:**

****