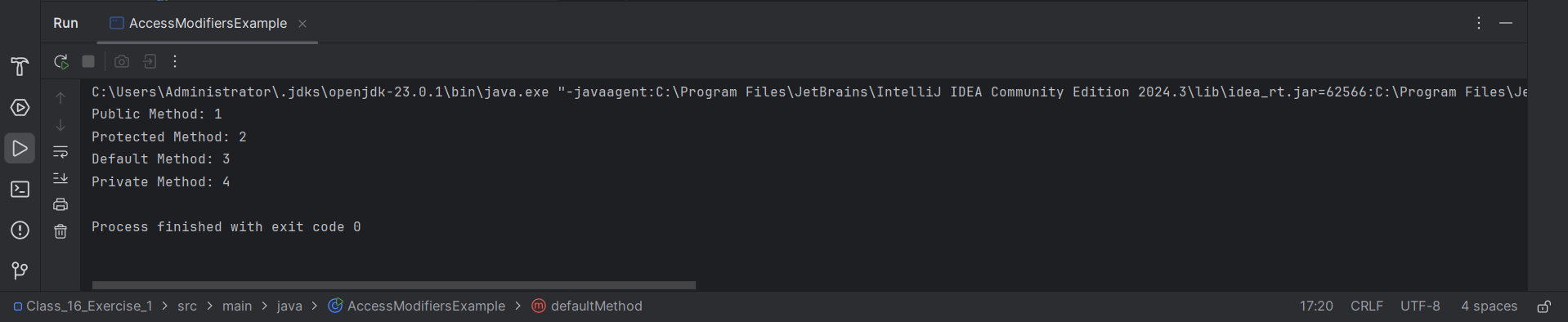
**Exercise 1**

**1.Access Modifiers**

**Code:**

class AccessModifiersExample {  
  
 public int publicVar = 1;  
 protected int protectedVar = 2;  
 int defaultVar = 3;  
 private int privateVar = 4;  
  
 public void publicMethod() {  
 System.*out*.println("Public Method: " + publicVar);  
 }  
  
 protected void protectedMethod() {  
 System.*out*.println("Protected Method: " + protectedVar);  
 }  
  
 void defaultMethod() {  
 System.*out*.println("Default Method: " + defaultVar);  
 }  
  
 private void privateMethod() {  
 System.*out*.println("Private Method: " + privateVar);  
 }  
  
 public void accessPrivate() {  
 privateMethod(); // Accessing private method within the class  
 }  
  
 public static void main(String[] args) {  
 AccessModifiersExample obj = new AccessModifiersExample();  
 obj.publicMethod();  
 obj.protectedMethod();  
 obj.defaultMethod();  
 obj.accessPrivate();  
 }  
}

**Output :**

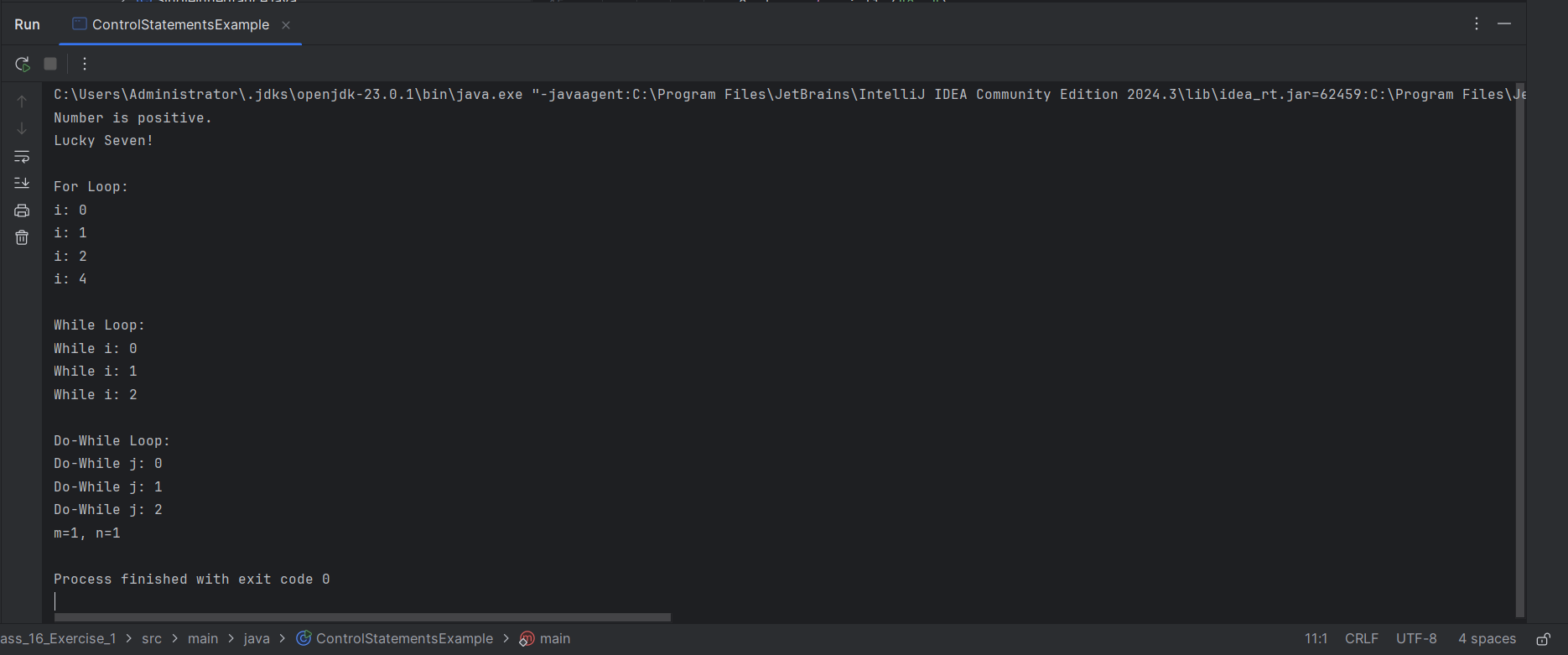
****

**2. Control Statements**

**Code:**

public class ControlStatementsExample {  
 public static void main(String[] args) {  
 int number = 7;  
  
 // If-else  
 if (number > 0) {  
 System.*out*.println("Number is positive.");  
 } else {  
 System.*out*.println("Number is non-positive.");  
 }  
  
 // Switch  
 switch (number) {  
 case 1:  
 System.*out*.println("One");  
 break;  
 case 7:  
 System.*out*.println("Lucky Seven!");  
 break;  
 default:  
 System.*out*.println("Not Matched");  
 }  
  
 // For loop  
 System.*out*.println("\nFor Loop:");  
 for (int i = 0; i < 5; i++) {  
 if (i == 3) continue;  
 System.*out*.println("i: " + i);  
 }  
  
 // While loop  
 System.*out*.println("\nWhile Loop:");  
 int i = 0;  
 while (i < 3) {  
 System.*out*.println("While i: " + i);  
 i++;  
 }  
  
 // Do-While  
 System.*out*.println("\nDo-While Loop:");  
 int j = 0;  
 do {  
 System.*out*.println("Do-While j: " + j);  
 j++;  
 } while (j < 3);  
  
 // Jump statement  
 outer: for (int m = 1; m <= 3; m++) {  
 for (int n = 1; n <= 3; n++) {  
 if (n == 2) {  
 break outer; // labeled break  
 }  
 System.*out*.println("m=" + m + ", n=" + n);  
 }  
 }  
 }  
}

**Output:**



**3.Data Operations**

**Code:**

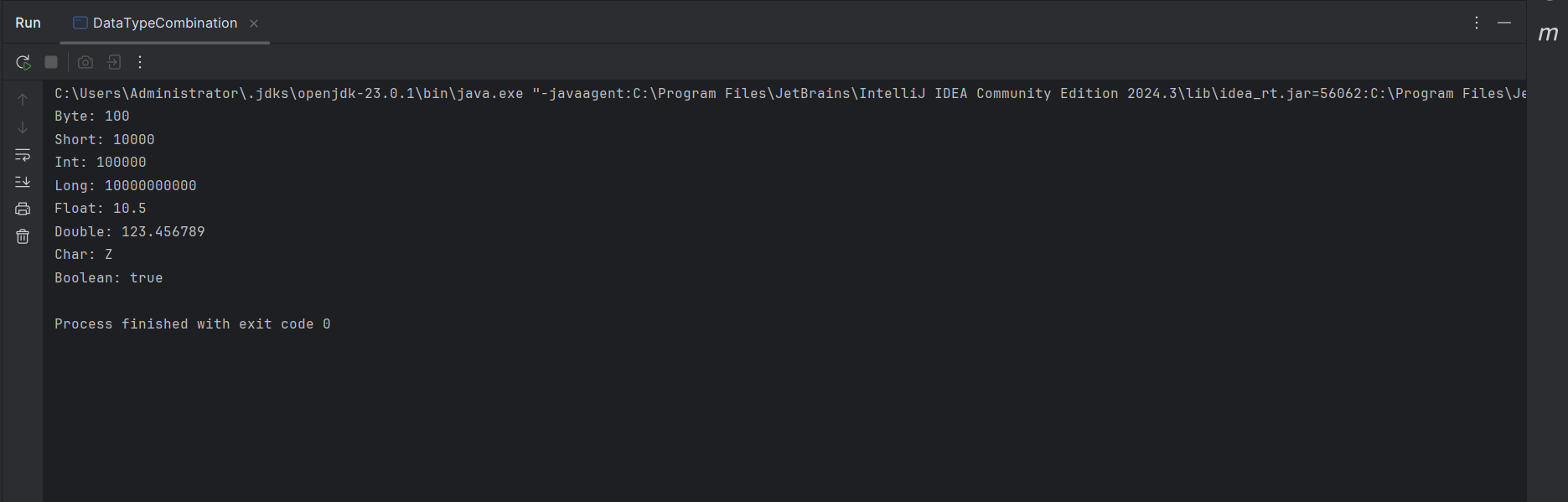
public class DataOperations {  
 public static void main(String[] args) {  
 int a = 10, b = 5;  
 float x = 3.5f, y = 1.5f;  
 double d1 = 4.567, d2 = 2.123;  
 char c1 = 'A', c2 = 'B';  
 boolean flag = true;  
  
 System.*out*.println("Arithmetic Operations:");  
 System.*out*.println("a + b = " + (a + b));  
 System.*out*.println("x - y = " + (x - y));  
 System.*out*.println("d1 \* d2 = " + (d1 \* d2));  
 System.*out*.println("a / b = " + (a / b));  
 System.*out*.println("a % b = " + (a % b));  
  
 System.*out*.println("\nRelational Operations:");  
 System.*out*.println("a > b = " + (a > b));  
 System.*out*.println("x <= y = " + (x <= y));  
  
 System.*out*.println("\nLogical Operations:");  
 System.*out*.println("flag && (a > b): " + (flag && (a > b)));  
 System.*out*.println("flag || (a < b): " + (flag || (a < b)));  
 System.*out*.println("!flag = " + (!flag));  
  
 System.*out*.println("\nAssignment and Compound Operations:");  
 a += 2; // a = a + 2  
 System.*out*.println("a += 2 => " + a);  
 x \*= 2; // x = x \* 2  
 System.*out*.println("x \*= 2 => " + x);  
  
 System.*out*.println("\nCharacter Operations:");  
 System.*out*.println("c1: " + c1 + ", ASCII: " + (int)c1);  
 }  
}

**4. Data Type combination**

**Aim :**

public class DataTypeCombination {  
 public static void main(String[] args) {  
 byte b = 100;  
 short s = 10000;  
 int i = 100000;  
 long l = 10000000000L;  
  
 float f = 10.5f;  
 double d = 123.456789;  
  
 char ch = 'Z';  
 boolean isJavaFun = true;  
  
 System.*out*.println("Byte: " + b);  
 System.*out*.println("Short: " + s);  
 System.*out*.println("Int: " + i);  
 System.*out*.println("Long: " + l);  
 System.*out*.println("Float: " + f);  
 System.*out*.println("Double: " + d);  
 System.*out*.println("Char: " + ch);  
 System.*out*.println("Boolean: " + isJavaFun);  
 }  
}

**Output:**

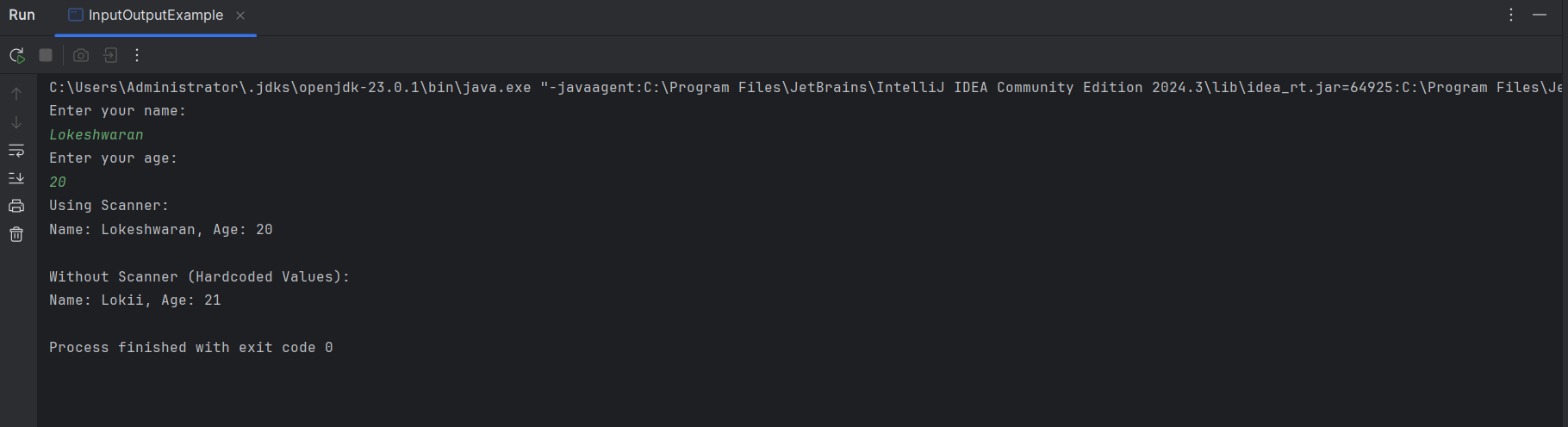
****

**5.Input Output Examples**

**Code:**

import java.util.Scanner;  
  
public class InputOutputExample {  
 public static void main(String[] args) {  
 // Using Scanner  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Enter your name: ");  
 String name = sc.nextLine();  
 System.*out*.println("Enter your age: ");  
 int age = sc.nextInt();  
  
 System.*out*.println("Using Scanner:");  
 System.*out*.println("Name: " + name + ", Age: " + age);  
  
 // Without Scanner  
 System.*out*.println("\nWithout Scanner (Hardcoded Values):");  
 String hardcodedName = "Lokii";  
 int hardcodedAge = 21;  
 System.*out*.println("Name: " + hardcodedName + ", Age: " + hardcodedAge);  
  
 sc.close();  
 }  
}

**Output:**

****

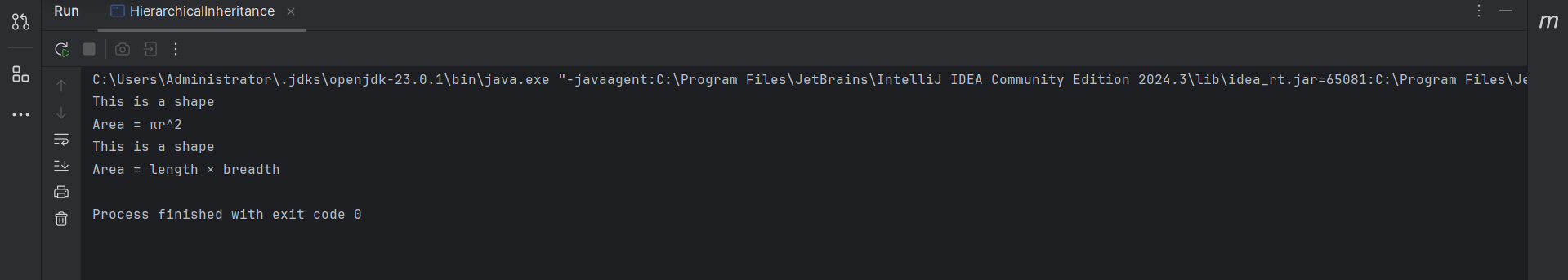
**Exercise 2**

**1.Heirarchical Inheritence**

**Code:**

package Exercise2;  
  
class Shape {  
 void display() {  
 System.*out*.println("This is a shape");  
 }  
}  
  
class Circle extends Shape {  
 void area() {  
 System.*out*.println("Area = πr^2");  
 }  
}  
  
class Rectangle extends Shape {  
 void area() {  
 System.*out*.println("Area = length × breadth");  
 }  
}  
  
public class HierarchicalInheritance {  
 public static void main(String[] args) {  
 Circle c = new Circle();  
 c.display();  
 c.area();  
  
 Rectangle r = new Rectangle();  
 r.display();  
 r.area();  
 }  
}

**Output:**

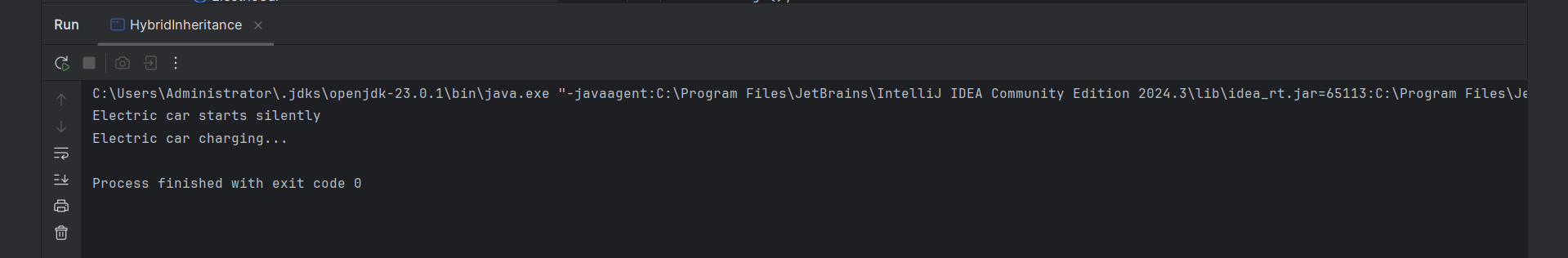


**2. Hybrid Inheritance**

**Code:**

package Exercise2;  
  
interface Engine {  
 void start();  
}  
  
interface Battery {  
 void charge();  
}  
  
class ElectricCar implements Engine, Battery {  
 public void start() {  
 System.*out*.println("Electric car starts silently");  
 }  
  
 public void charge() {  
 System.*out*.println("Electric car charging...");  
 }  
}  
  
public class HybridInheritance {  
 public static void main(String[] args) {  
 ElectricCar ec = new ElectricCar();  
 ec.start();  
 ec.charge();  
 }  
}

**Output:**

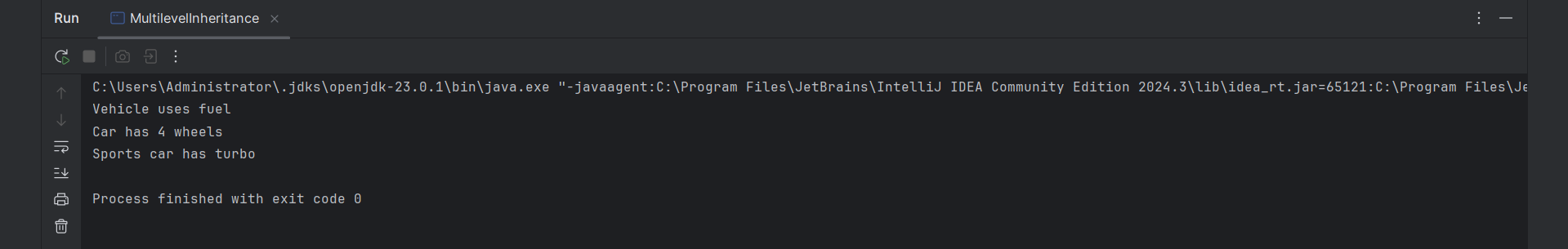
****

**3.MultiLevel Inheritance**

**Code:**

package Exercise2;  
  
class Vehicle {  
 void fuel() {  
 System.*out*.println("Vehicle uses fuel");  
 }  
}  
  
class Car extends Vehicle {  
 void wheels() {  
 System.*out*.println("Car has 4 wheels");  
 }  
}  
  
class SportsCar extends Car {  
 void turbo() {  
 System.*out*.println("Sports car has turbo");  
 }  
}  
  
public class MultilevelInheritance {  
 public static void main(String[] args) {  
 SportsCar sc = new SportsCar();  
 sc.fuel(); // from Vehicle  
 sc.wheels(); // from Car  
 sc.turbo(); // from SportsCar  
 }  
}

**Output:**

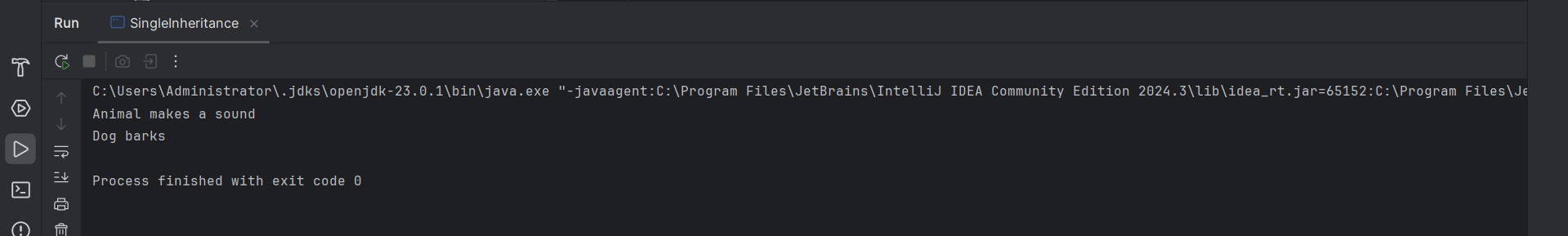
****

**4.SingleInheritance**

**Code:**

package Exercise2;  
  
class Animal {  
 void sound() {  
 System.*out*.println("Animal makes a sound");  
 }  
}  
  
class Dog extends Animal {  
 void bark() {  
 System.*out*.println("Dog barks");  
 }  
}  
  
public class SingleInheritance {  
 public static void main(String[] args) {  
 Dog d = new Dog();  
 d.sound(); // inherited method  
 d.bark(); // child method  
 }  
}

**Output:**

****

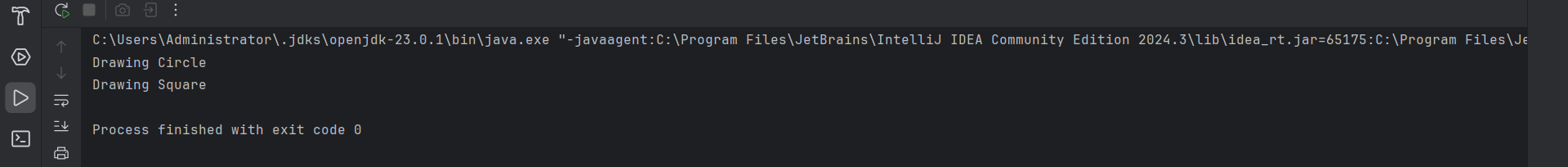
**Exercise 3**

**1.Absrtract Polymorphism**

**Code:**

package Exercise3;  
  
abstract class Shape {  
 abstract void draw(); // abstract method  
}  
  
class Circle extends Shape {  
 void draw() {  
 System.*out*.println("Drawing Circle");  
 }  
}  
  
class Square extends Shape {  
 void draw() {  
 System.*out*.println("Drawing Square");  
 }  
}  
  
public class AbstractPolymorphism {  
 public static void main(String[] args) {  
 Shape s;  
  
 s = new Circle();  
 s.draw();  
  
 s = new Square();  
 s.draw();  
 }  
}

**Output:**

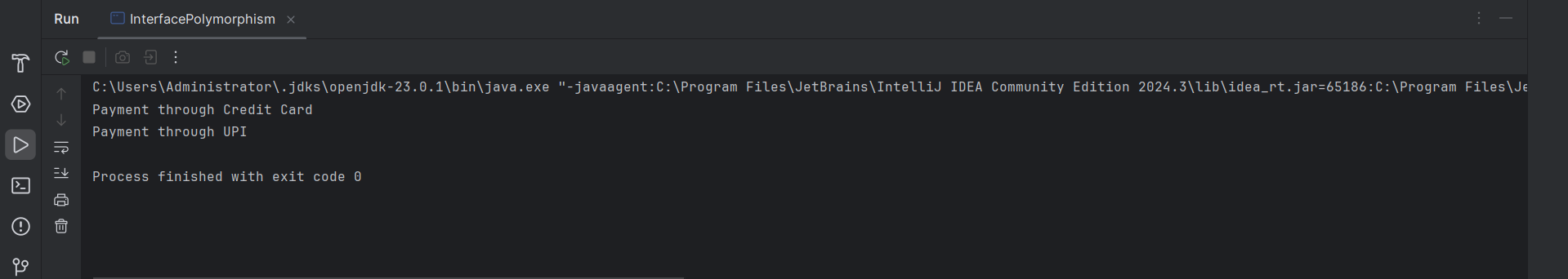
****

**2.Interface Polymorphism**

**Code:**

package Exercise3;  
  
interface Payment {  
 void pay();  
}  
  
class CreditCard implements Payment {  
 public void pay() {  
 System.*out*.println("Payment through Credit Card");  
 }  
}  
  
class UPI implements Payment {  
 public void pay() {  
 System.*out*.println("Payment through UPI");  
 }  
}  
  
public class InterfacePolymorphism {  
 public static void main(String[] args) {  
 Payment p;  
  
 p = new CreditCard();  
 p.pay();  
  
 p = new UPI();  
 p.pay();  
 }  
}

**Output:**

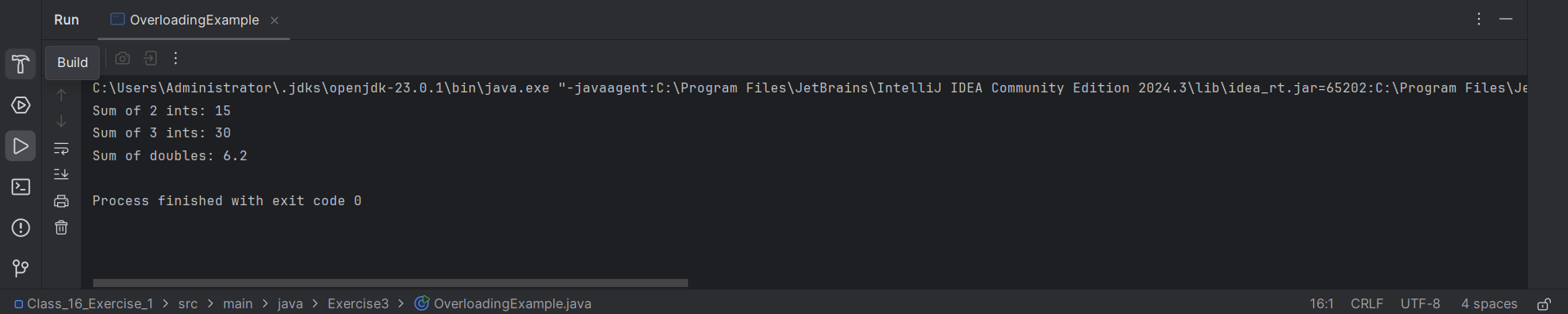
****

**3.Overloading Example**

**Code:**

package Exercise3;  
  
class Calculator {  
 int add(int a, int b) {  
 return a + b;  
 }  
  
 int add(int a, int b, int c) {  
 return a + b + c;  
 }  
  
 double add(double a, double b) {  
 return a + b;  
 }  
}  
  
public class OverloadingExample {  
 public static void main(String[] args) {  
 Calculator c = new Calculator();  
 System.*out*.println("Sum of 2 ints: " + c.add(5, 10));  
 System.*out*.println("Sum of 3 ints: " + c.add(5, 10, 15));  
 System.*out*.println("Sum of doubles: " + c.add(2.5, 3.7));  
 }  
}

**Output:**

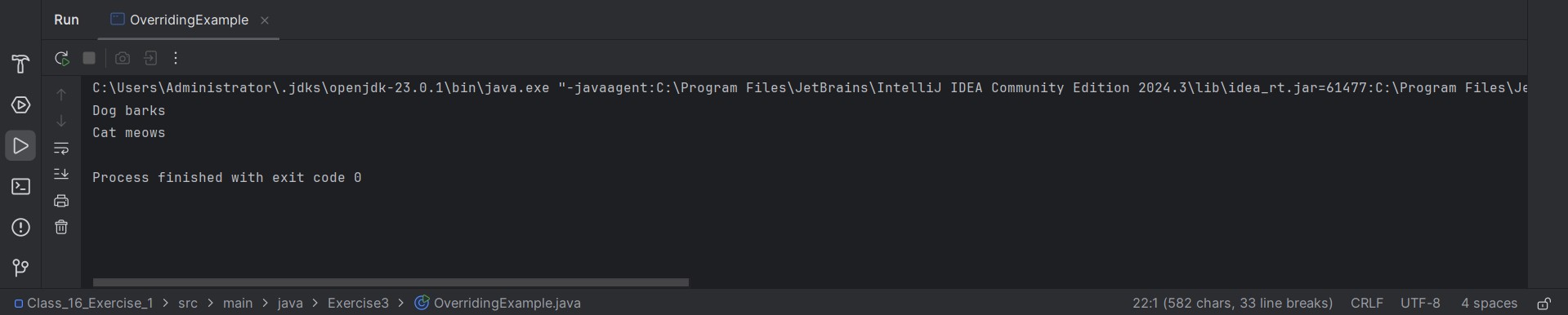
****

**4.Ovverrding Example**

**Code:**

package Exercise3;  
  
class Animal {  
 void sound() {  
 System.*out*.println("Animals make sounds");  
 }  
}  
  
class Dog extends Animal {  
 @Override  
 void sound() {  
 System.*out*.println("Dog barks");  
 }  
}  
  
class Cat extends Animal {  
 @Override  
 void sound() {  
 System.*out*.println("Cat meows");  
 }  
}  
  
public class OverridingExample {  
 public static void main(String[] args) {  
 Animal a;  
  
 a = new Dog(); // upcasting  
 a.sound(); // Dog's sound()  
  
 a = new Cat();  
 a.sound(); // Cat's sound()  
 }  
}

**Output**

****