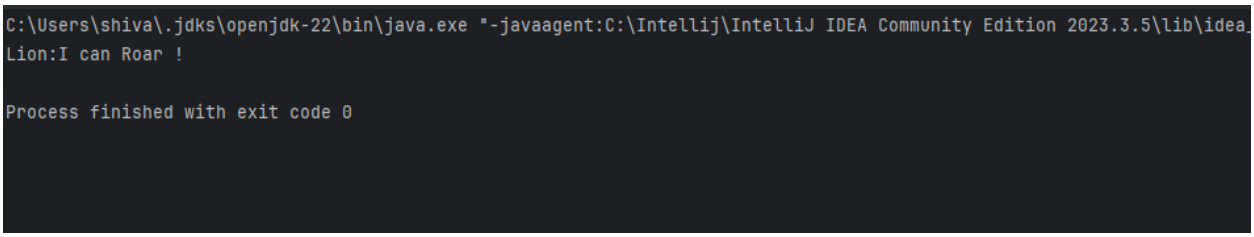
```
ABSTRACTION IMPLEMENTATION WITH EXAMPLE PROGRAMS:
1.DATA ABSTRACTION
2.PROCESS ABSTRACTION

// Data Abstraction:
abstract class Laptop
{
    public abstract void Turnon();
}


class Poweron extends Laptop
{
    public void Turnon()
    {
        System.out.println("Turning on the Laptop.........");
    }
}


public class Abstraction {
    public static void main(String[] args) {
        Laptop obj = new Poweron();
        obj.Turnon();
    }
}
```

```
C:\Users\shiva\.jdks\openjdk-22\bin\java.exe "-javaagent:C:\Intellij\IntelliJ IDEA Community Edition
Turning on the Laptop.........

Process finished with exit code 0
```

```
//Process Abstraction :
abstract class calc
{
    public static int multiply(int a,int b)
    {
        return a*b;
    }
}

public class ProcessAbstarctiom {
    public static void main(String[] args) {

        int result= calc.multiply(5,10);
        System.out.println(result);
    }
}
```

```
C:\Users\shiva\.jdks\openjdk-22\bin\java.exe "-javaagent:C:\Intellij\IntelliJ IDEA Community Edition 2023.3.5
50

Process finished with exit code 0
```

```java
abstract class Animalsss
{
    public abstract void givesound();

}


class Lion extends Animalsss
{
    public void givesound()
    {
        System.out.println("Lion:I can Roar !");
    }
}

class Catt extends Animalsss
{
    public void givesound()
    {
        System.out.println("Cat:I can Meow !");
    }
}

class Dogg extends Animalsss
{
    public void givesound()
    {
        System.out.println("Dog:I can bark !");
    }
}



public class Abstractioneg {

    public static void main(String[] args) {

        Animalsss obj = new Lion();
        obj.givesound();
    }
}
```



• Abstract **classes** are used to provide common features to all subclasses while enforcing that certain methods must be implemented.

• Abstract **methods** serve as placeholders, and subclasses are responsible for providing specific implementations.