Day 9

class ----->  class ======> extends  ====> no multiple inheritance  
  
interface ----> class =====> Implements  
  
interface ---> interface ====> extends

Exceptions:   --> unexpected situations

  2 categoried  
    - Checked  ---> compile time  
    - Unchecked  ---> Runtime time..

  pre defined exception classes  
    FileNotFound  
    ArrayIndexoutofBounds exception  
    NullPointer Exception  
    Arthematic Exception ...  
  
  user defined Exceptions  
  
try catch blocks  
  try{

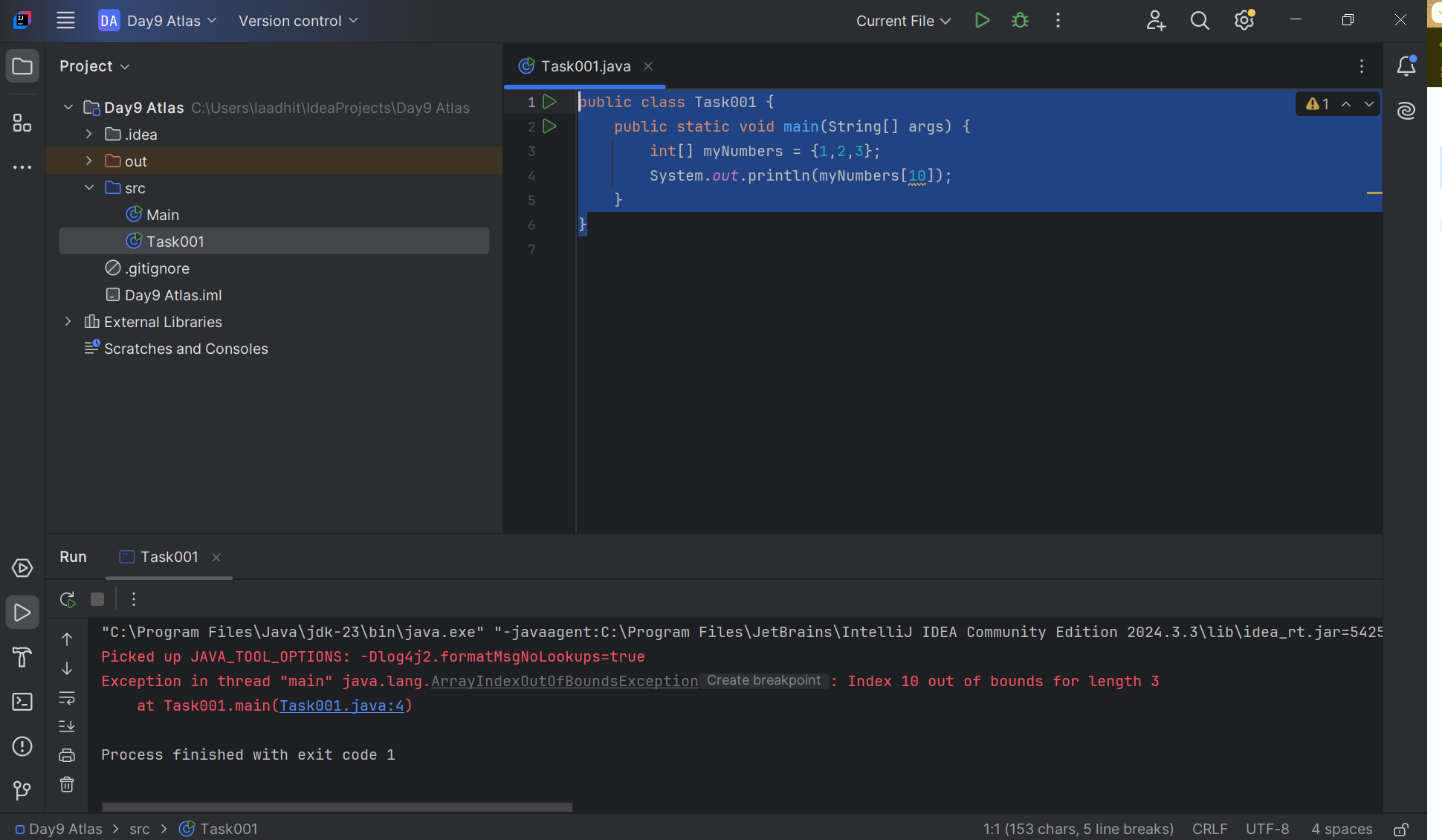
  }catch(Exception ex){

  }catch(ArthematicException ex1{  
  
  }  
  finally{

  sout(" ");  
  }

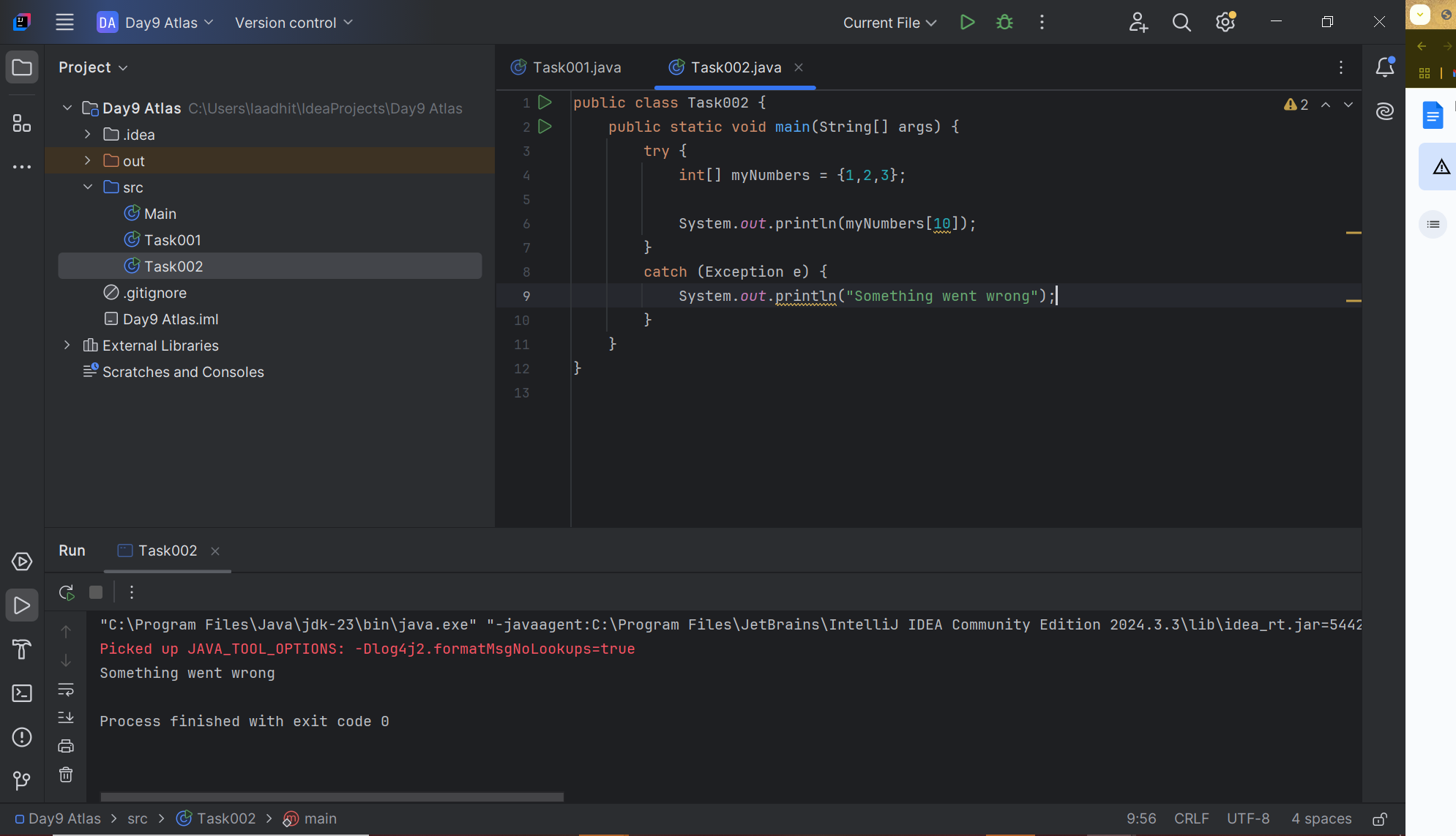
Task 1

public class Task001 {  
 public static void main(String[] args) {  
 int[] myNumbers = {1,2,3};  
 System.*out*.println(myNumbers[10]);  
 }  
}



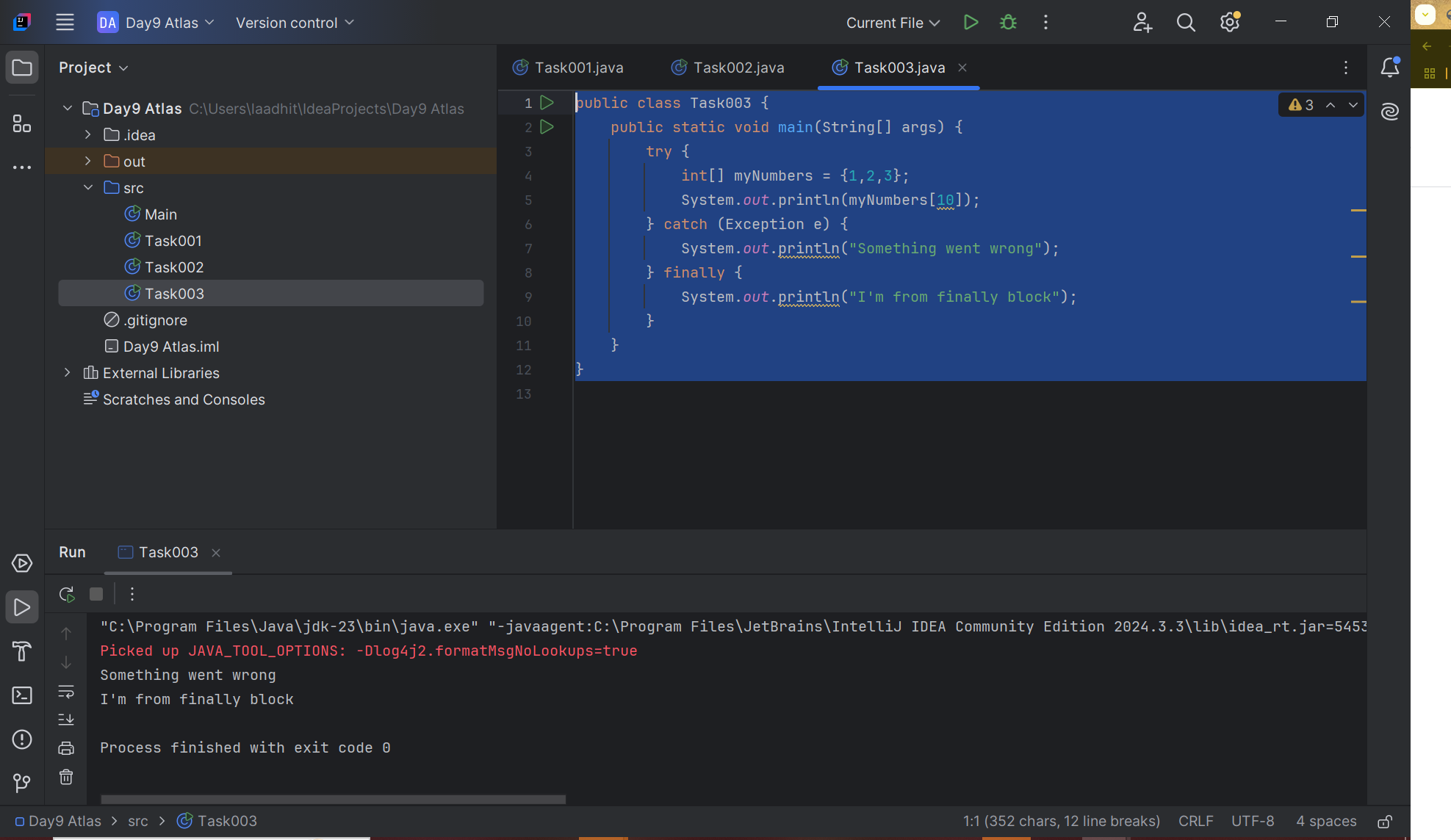
 Task 2

public class Task002 {  
 public static void main(String[] args) {  
 try {  
 int[] myNumbers = {1,2,3};  
  
 System.*out*.println(myNumbers[10]);  
 }  
 catch (Exception e) {  
 System.*out*.println("Something went wrong");  
 }  
 }  
}



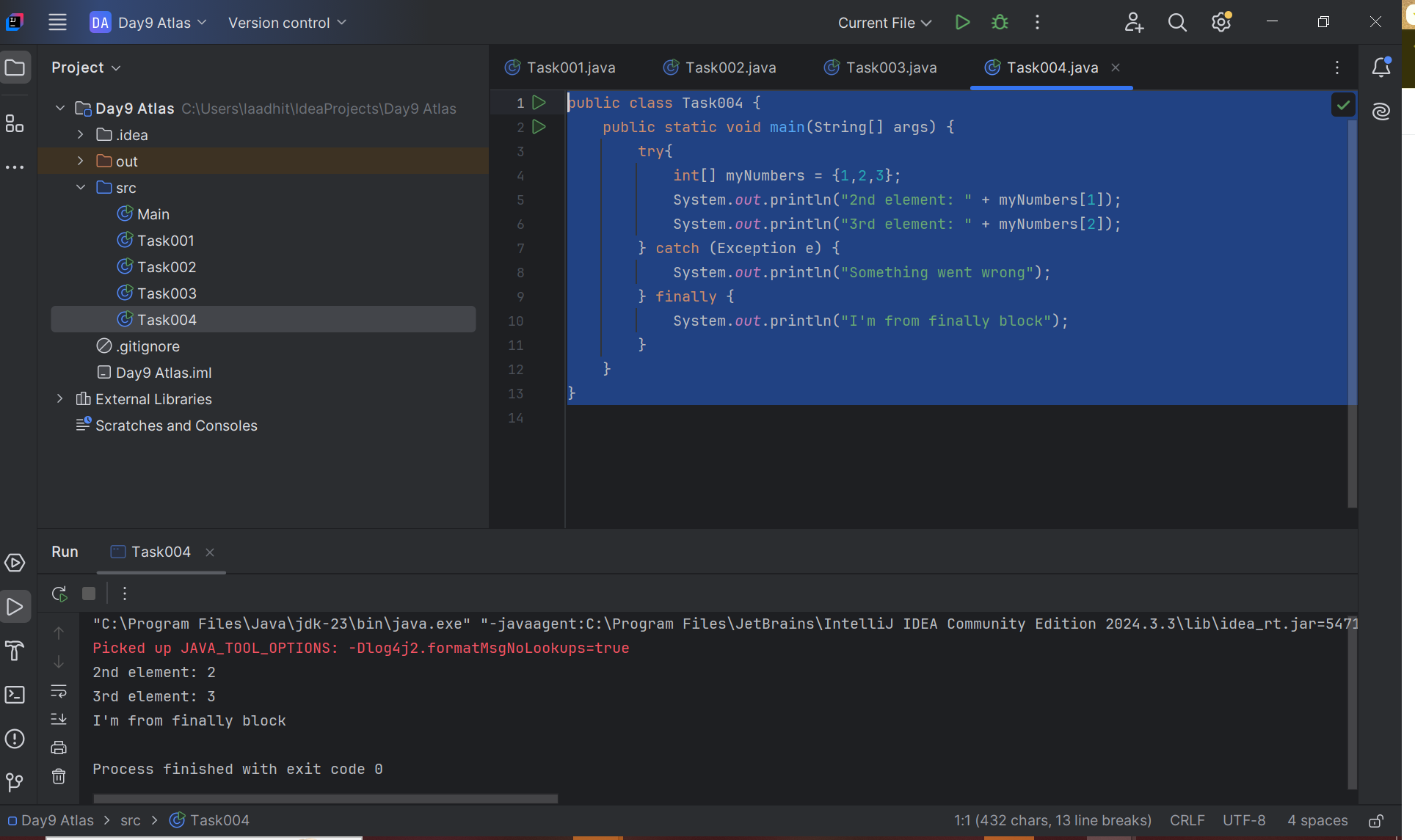
Task003:

public class Task003 {  
 public static void main(String[] args) {  
 try {  
 int[] myNumbers = {1,2,3};  
 System.*out*.println(myNumbers[10]);  
 } catch (Exception e) {  
 System.*out*.println("Something went wrong");  
 } finally {  
 System.*out*.println("I'm from finally block");  
 }  
 }  
}



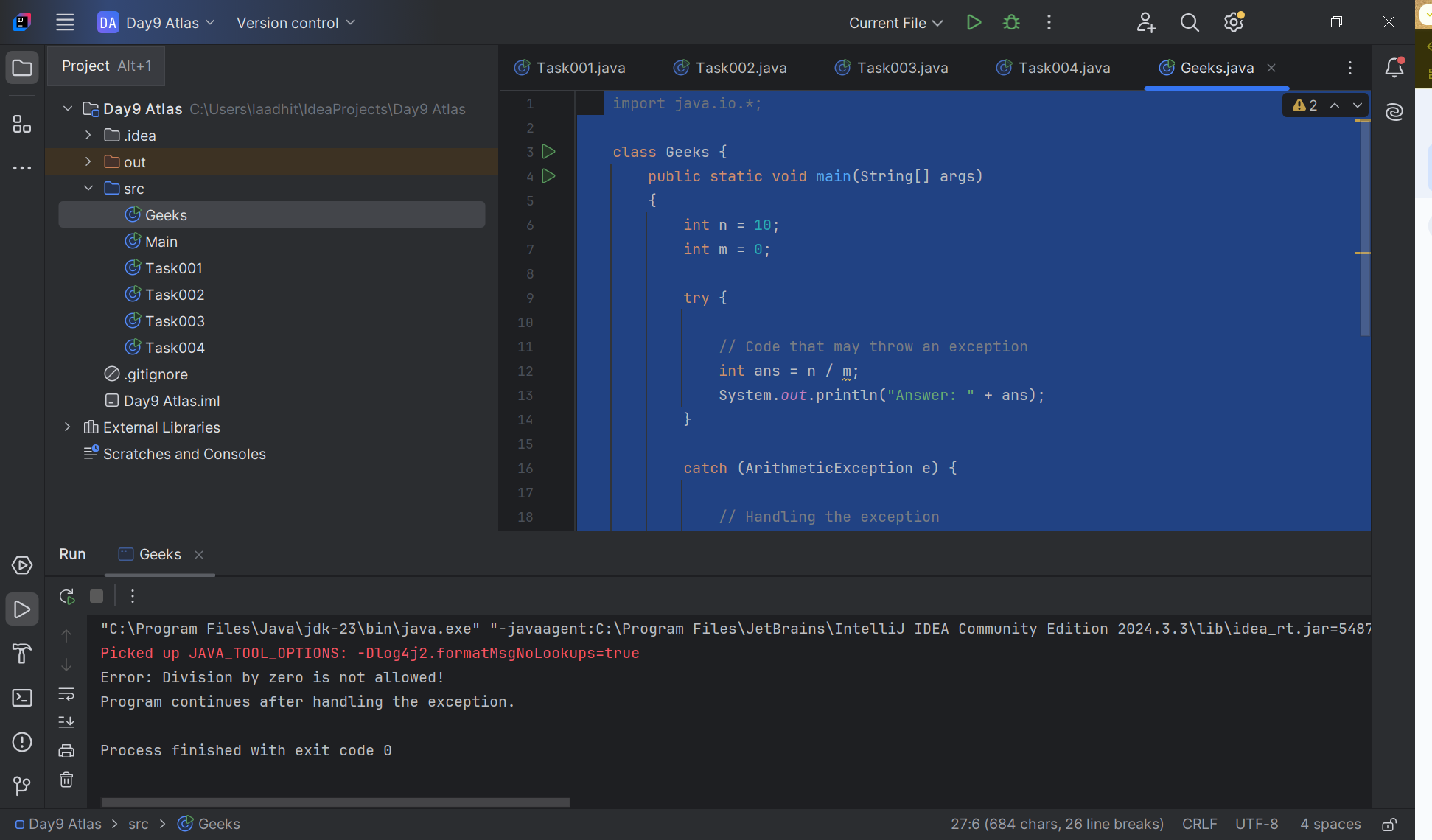
Task004:

public class Task004 {  
 public static void main(String[] args) {  
 try{  
 int[] myNumbers = {1,2,3};  
 System.*out*.println("2nd element: " + myNumbers[1]);  
 System.*out*.println("3rd element: " + myNumbers[2]);  
 } catch (Exception e) {  
 System.*out*.println("Something went wrong");  
 } finally {  
 System.*out*.println("I'm from finally block");  
 }  
 }  
}



Adhoc tasks:

import java.io.\*;  
  
class Geeks {  
 public static void main(String[] args)  
 {  
 int n = 10;  
 int m = 0;  
  
 try {  
  
 // Code that may throw an exception  
 int ans = n / m;  
 System.*out*.println("Answer: " + ans);  
 }  
  
 catch (ArithmeticException e) {  
  
 // Handling the exception  
 System.*out*.println(  
 "Error: Division by zero is not allowed!");  
 }  
 finally {  
 System.*out*.println(  
 "Program continues after handling the exception.");  
 }  
 }  
}



Try with Multiple catch blocks  …. Execute the below code snippet n display the out .. along with reason..

**Task 0005:**

public class Task0005{

   public static void main(String args[]) {

      try {

         int a[] = new int[2];

         int b = 0;

         int c = 1/b;

         System.out.println("Access element three :" + a[3]);

      }

      catch (ArrayIndexOutOfBoundsException e) {

         System.out.println("ArrayIndexOutOfBoundsException thrown  :" + e);

      }catch (Exception e) {

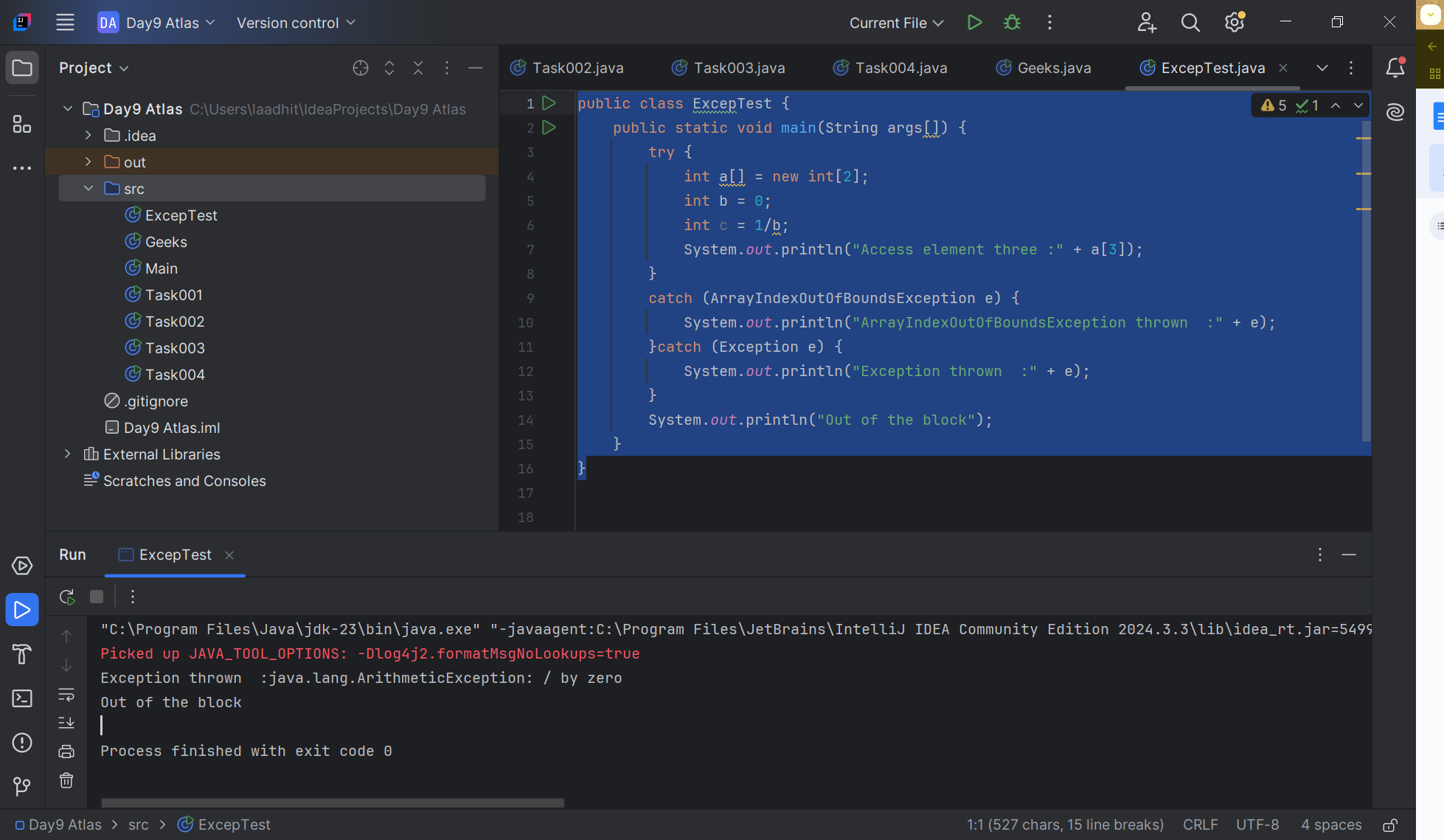
          System.out.println("Exception thrown  :" + e);

      }

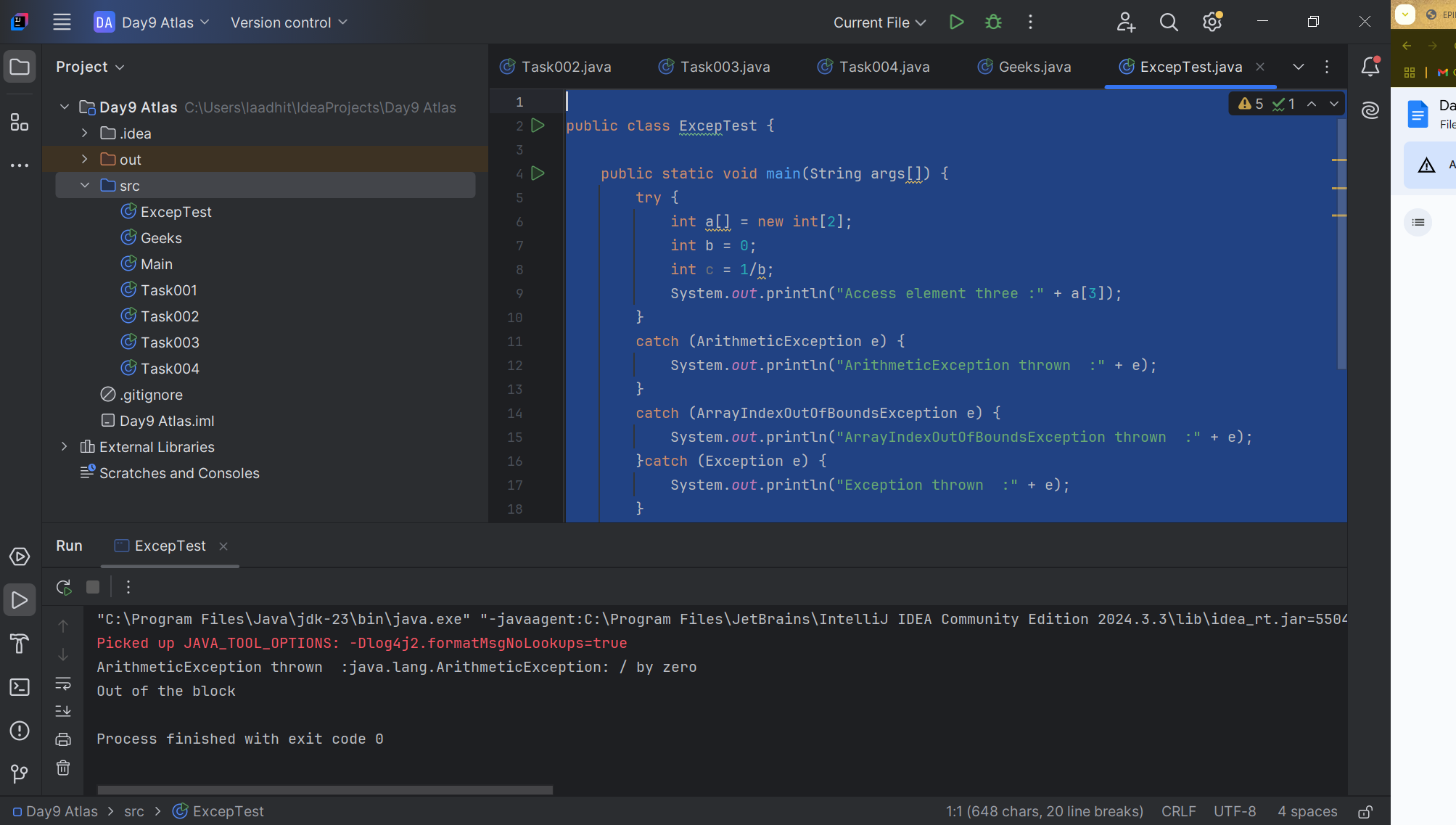
      System.out.println("Out of the block");

   }

}



public class Task0005 {  
  
 public static void main(String args[]) {  
 try {  
 int a[] = new int[2];  
 int b = 0;  
 int c = 1/b;  
 System.*out*.println("Access element three :" + a[3]);  
 }  
 catch (ArithmeticException e) {  
 System.*out*.println("ArithmeticException thrown :" + e);  
 }  
 catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println("ArrayIndexOutOfBoundsException thrown :" + e);  
 }catch (Exception e) {  
 System.*out*.println("Exception thrown :" + e);  
 }  
 System.*out*.println("Out of the block");  
 }  
}



Task 6:

Nested try blocks

public class Task0006{

   public static void main(String args[]) {

      try {

         int a[] = new int[2];

        System.out.println("Access element three :" + a[2]); // try with a[0] or a[1]  ===> and check if control goes to inner try block..

         try {

            int b = 0;

            int c = 1/b;

         }catch(Exception e) {

            System.out.println("Exception thrown: " + e);

         }

         System.out.println("Access element three :" + a[3]);

      }

      catch (ArrayIndexOutOfBoundsException e) {

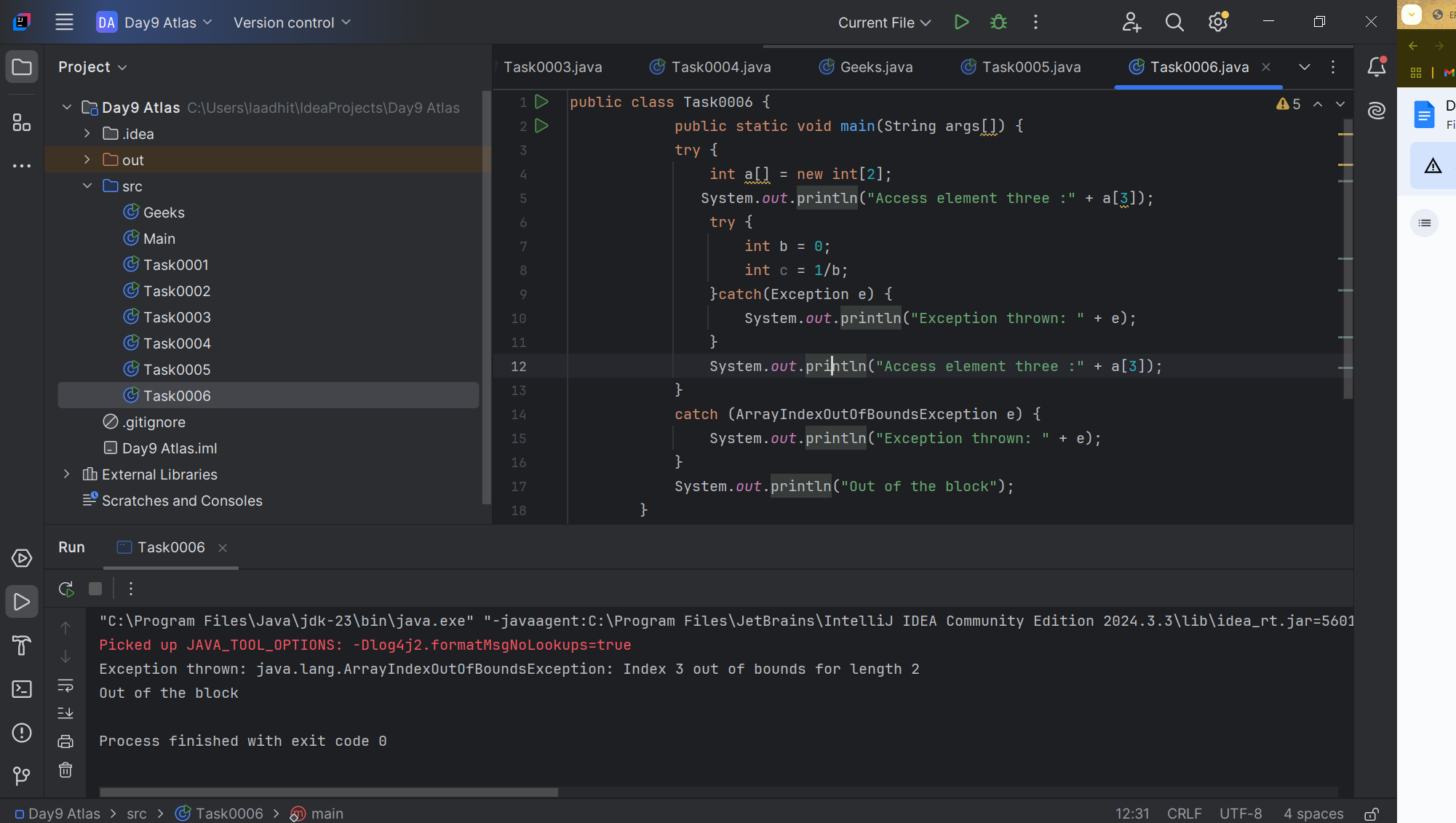
         System.out.println("Exception thrown: " + e);

      }

      System.out.println("Out of the block");

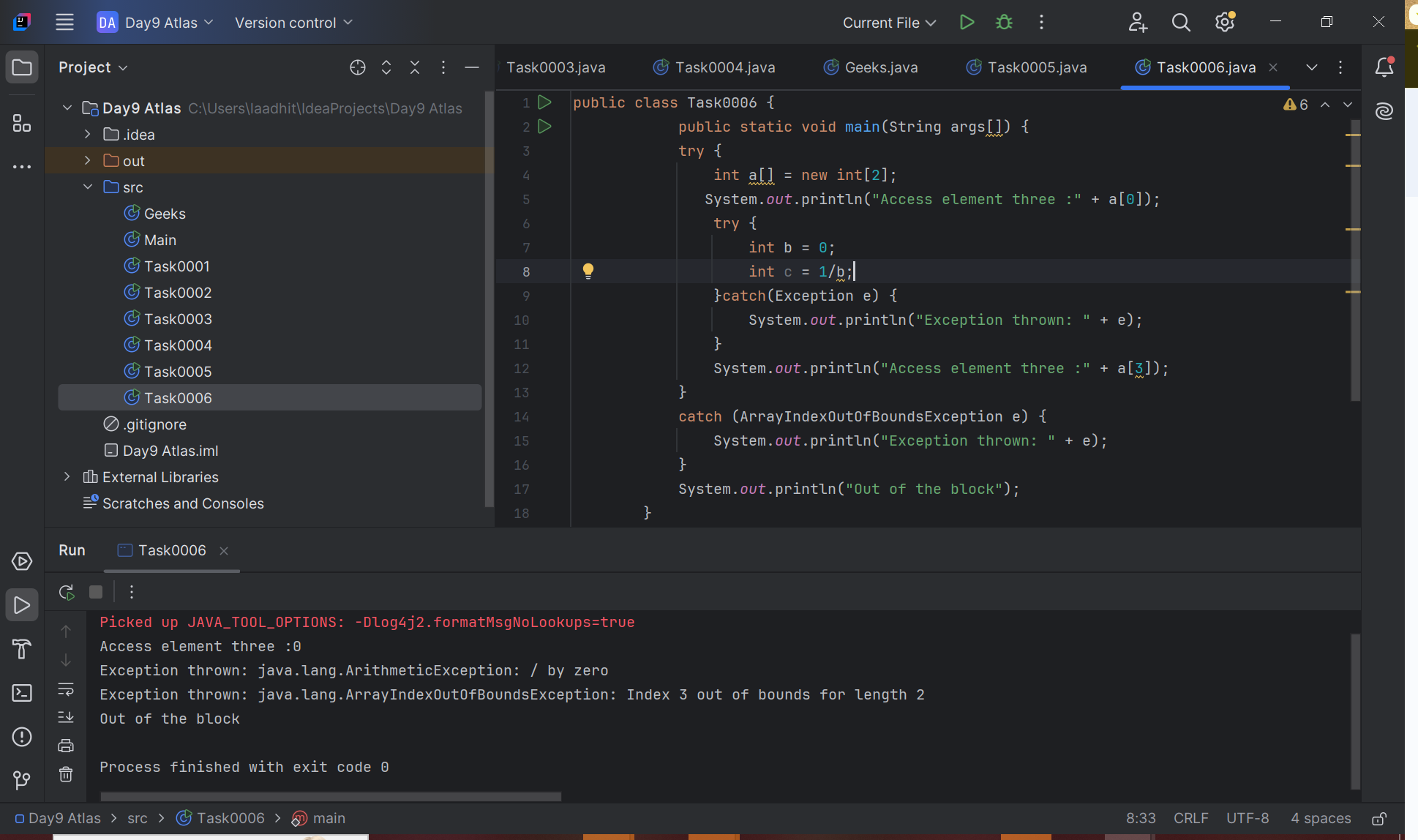
   }

}



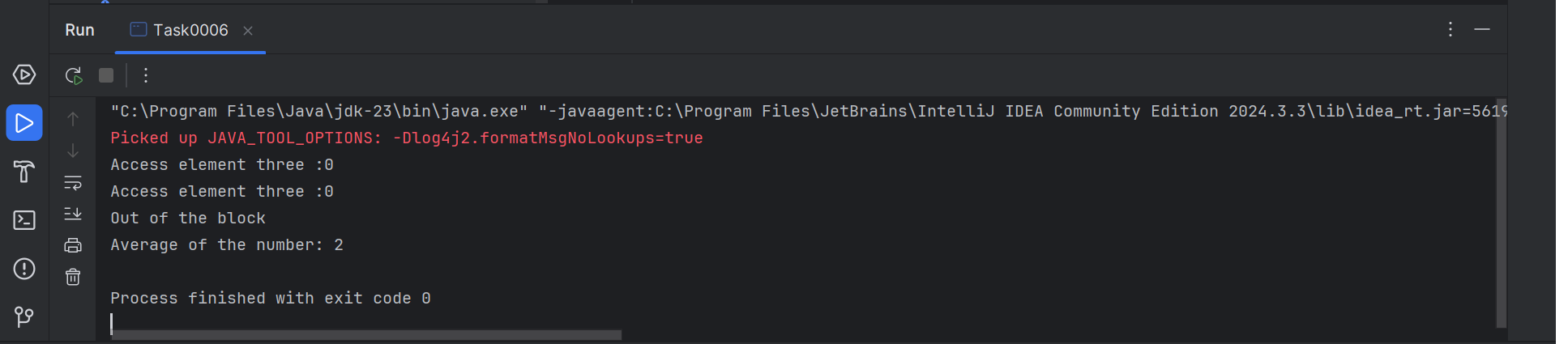
**Changing the index from a[3] to a[0] or a[1];**

**public class Task0006 {  
 public static void main(String args[]) {  
 try {  
 int a[] = new int[2];  
 System.*out*.println("Access element three :" + a[0]);  
 try {  
 int b = 0;  
 int c = 1/b;  
 }catch(Exception e) {  
 System.*out*.println("Exception thrown: " + e);  
 }  
 System.*out*.println("Access element three :" + a[3]);  
 }  
 catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println("Exception thrown: " + e);  
 }  
 System.*out*.println("Out of the block");  
 }  
 }**



**Small try on task 0006:**

**public class Task0006 {  
 public static void main(String args[]) {  
 int c = 0;  
 try {  
 int a[] = new int[2];  
 System.*out*.println("Access element three :" + a[0]);  
 try {  
 int b = 10;  
 c = 20/b;  
 }catch(Exception e) {  
 System.*out*.println("Exception thrown: " + e);  
 }  
 System.*out*.println("Access element three :" + a[1]);  
 }  
 catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println("Exception thrown: " + e);  
 }finally {  
 System.*out*.println("Out of the block");  
 System.*out*.println("Average of the number: " + c);  
 }  
 }  
 }**



Task 0007:

// Demonstrating how to throw an exception

class MyClass {

    static void fun() throws IllegalAccessException

    {

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

        throw new IllegalAccessException("demo exception by fun method");

    }

    public static void main(String args[])

    {

        try {

            fun();

//method2();   → arrayindex…

//Method3()  —> file not found….

        }

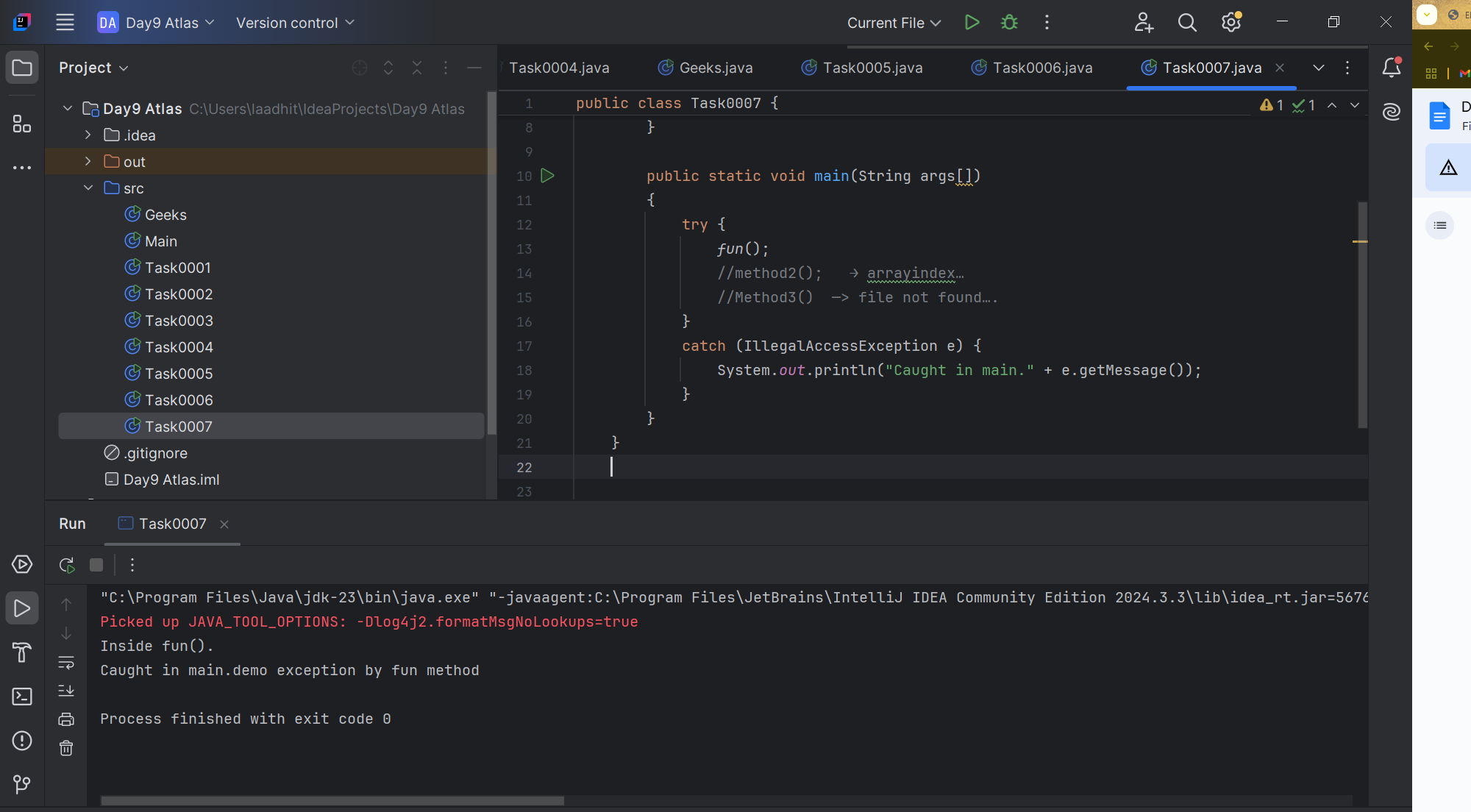
        catch (IllegalAccessException e) {

            System.out.println("Caught in main." + e.getMessage());

        }

    }

}



Task 0008:

Task 8:

Custom exceptions: // user defined exception:

// A Class that represents user-defined exception

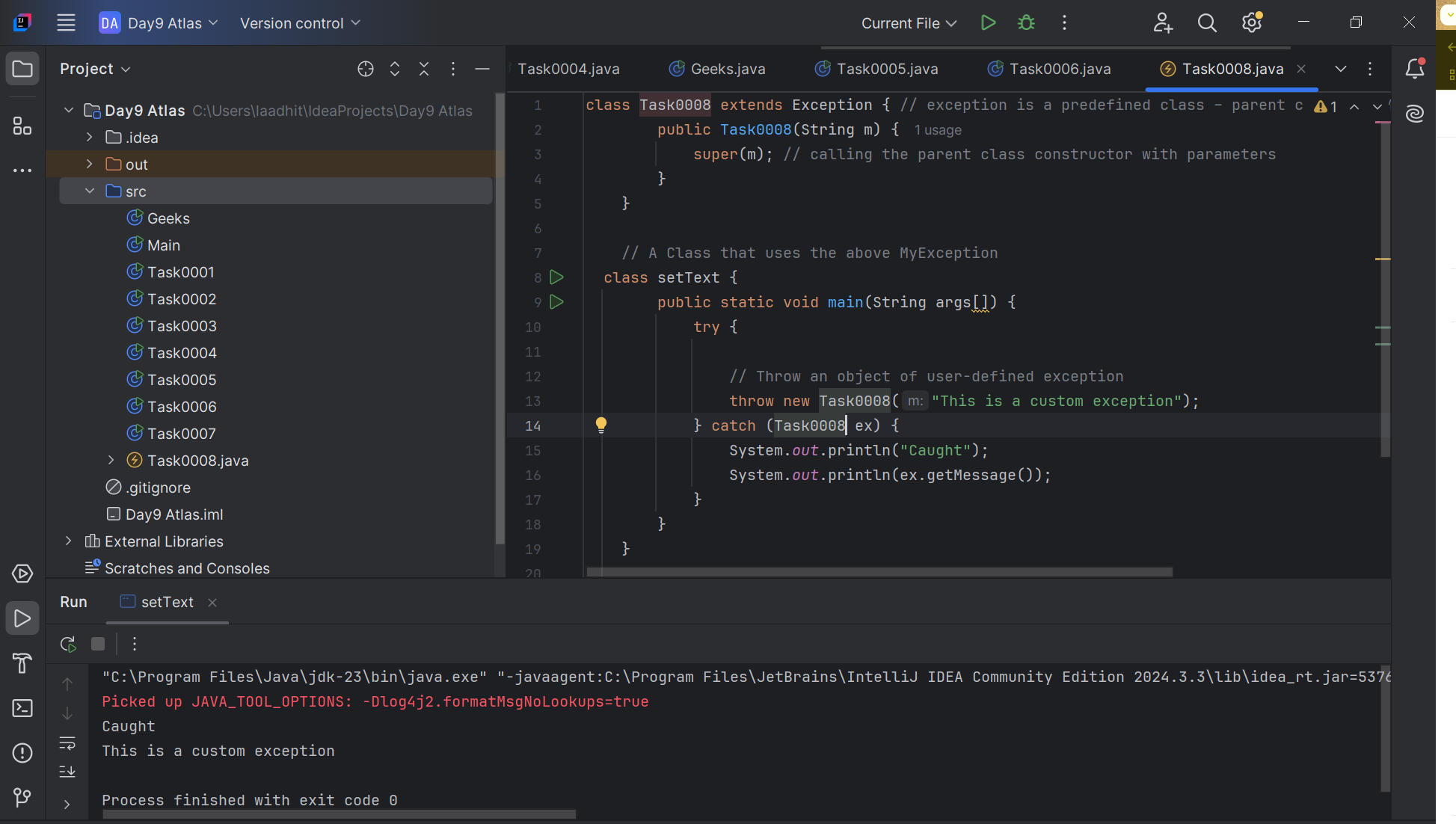
class MyException extends Exception { // exception is a predefined class – parent class for MyException

    public MyException(String m) {

        super(m); // calling the parent class constructor with parameters

    }

}



// A Class that uses the above MyException

public class setText {

    public static void main(String args[]) {

        try {

            // Throw an object of user-defined exception

            throw new MyException("This is a custom exception");

        }

        catch (MyException ex) {

            System.out.println("Caught");

            System.out.println(ex.getMessage());

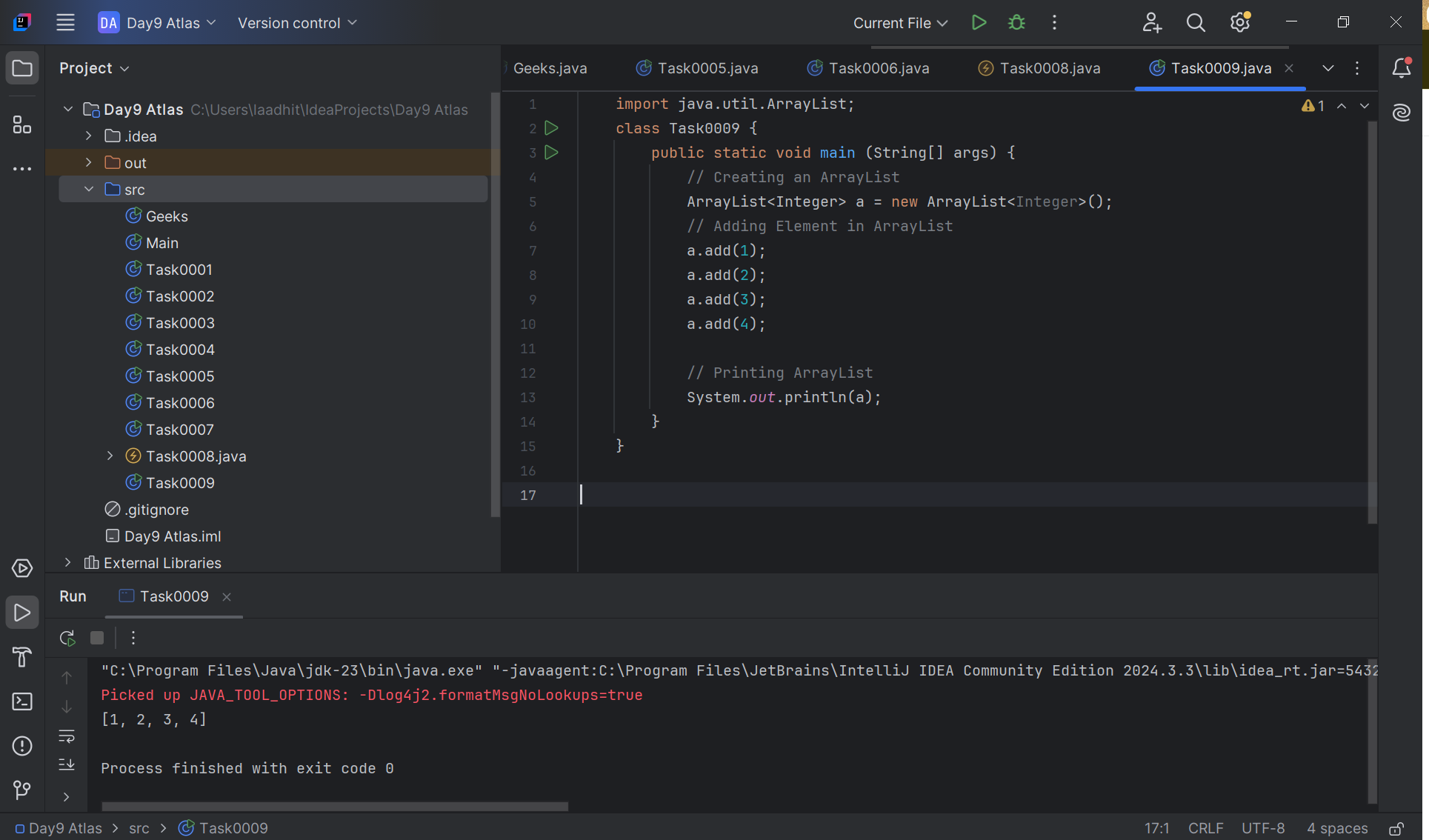
        }

    }

}

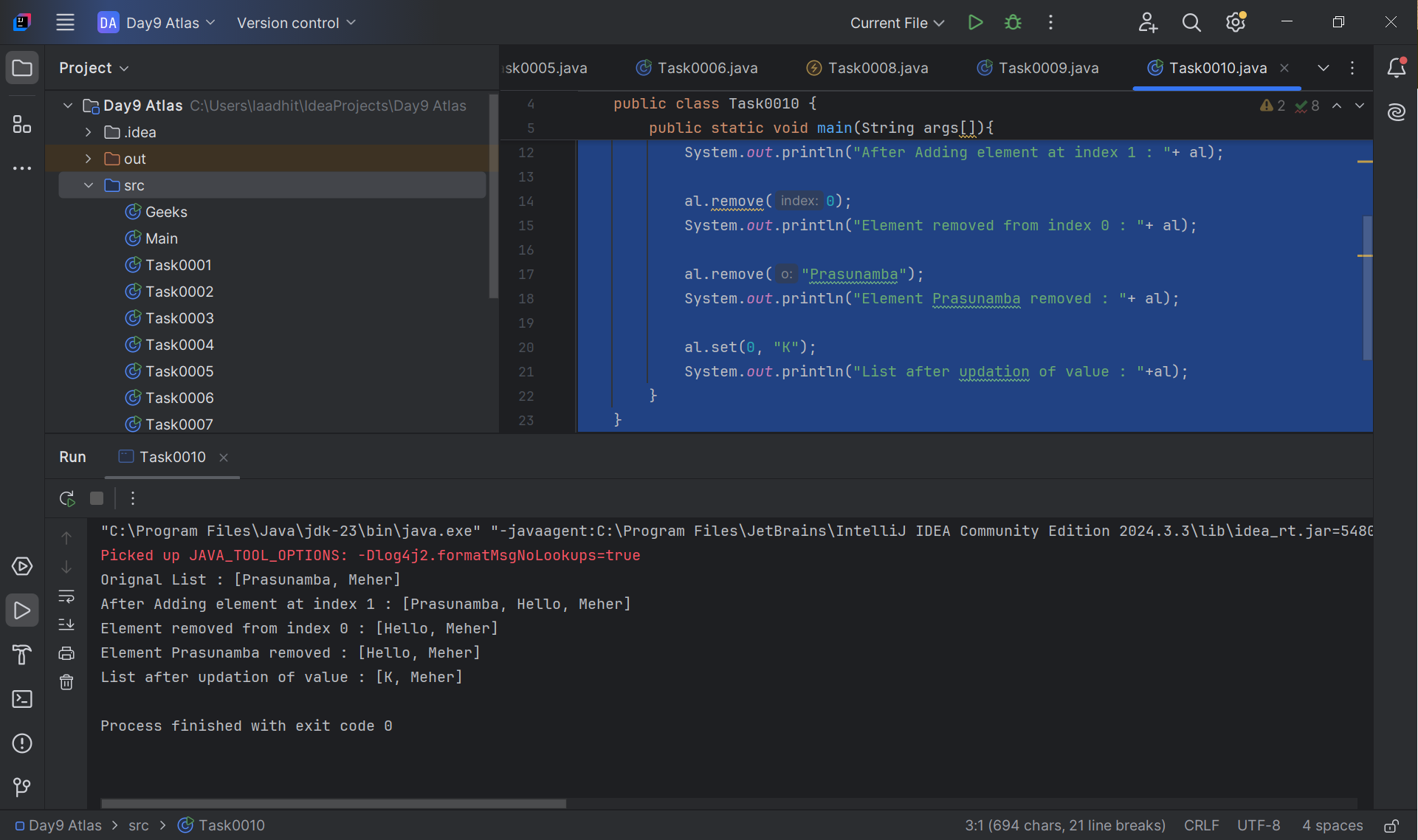
Task0009:

import java.util.ArrayList;  
class Task0009 {  
 public static void main (String[] args) {  
 // Creating an ArrayList  
 ArrayList<Integer> a = new ArrayList<Integer>();  
 // Adding Element in ArrayList  
 a.add(1);  
 a.add(2);  
 a.add(3);  
 a.add(4);  
  
 // Printing ArrayList  
 System.*out*.println(a);  
 }  
}



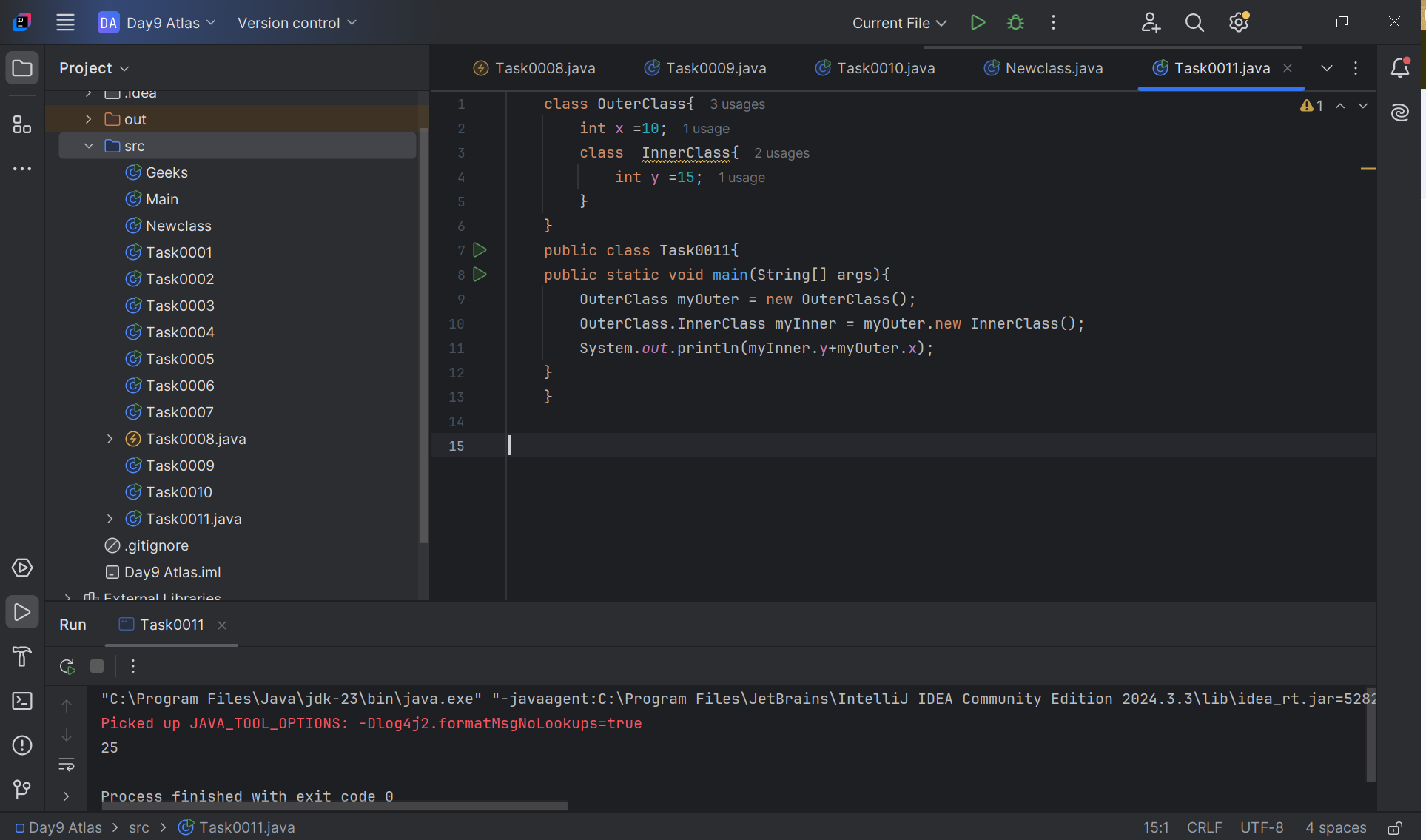
Task0010:

import java.util.\*;  
 public class Task0010 {  
 public static void main(String args[]){  
 ArrayList<String> al = new ArrayList<>();  
 al.add("Prasunamba");  
 al.add("Meher");  
 System.*out*.println("Orignal List : "+al);  
  
 al.add(1, "Hello");  
 System.*out*.println("After Adding element at index 1 : "+ al);  
  
 al.remove(0);  
 System.*out*.println("Element removed from index 0 : "+ al);  
  
 al.remove("Prasunamba");  
 System.*out*.println("Element Prasunamba removed : "+ al);  
  
 al.set(0, "K");  
 System.*out*.println("List after updation of value : "+al);  
 }  
 }



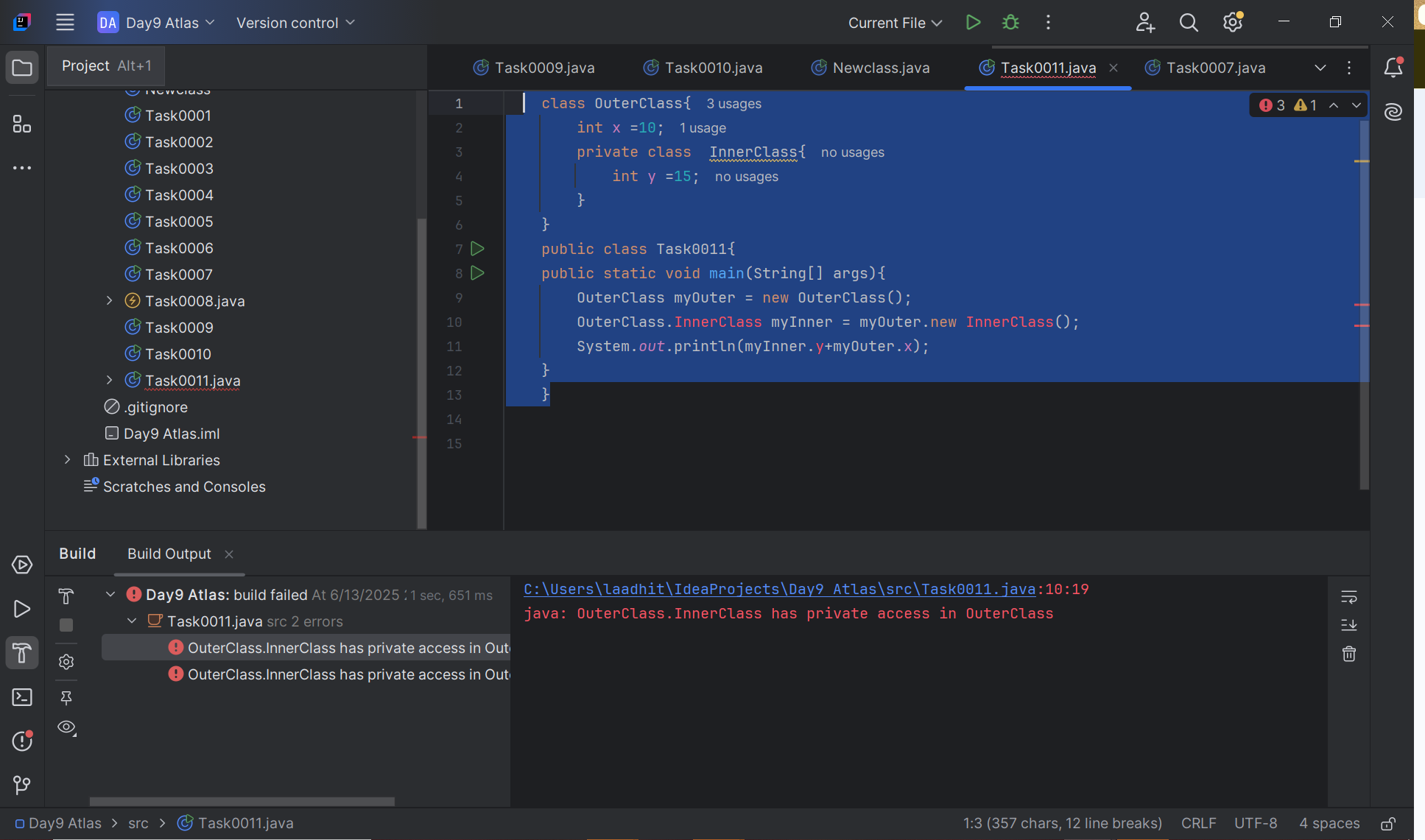
Task0011:

class OuterClass{  
 int x =10;  
 class InnerClass{  
 int y =15;  
 }  
}  
public class Task0011{  
public static void main(String[] args){  
 OuterClass myOuter = new OuterClass();  
 OuterClass.InnerClass myInner = myOuter.new InnerClass();  
 System.*out*.println(myInner.y+myOuter.x);  
}  
}



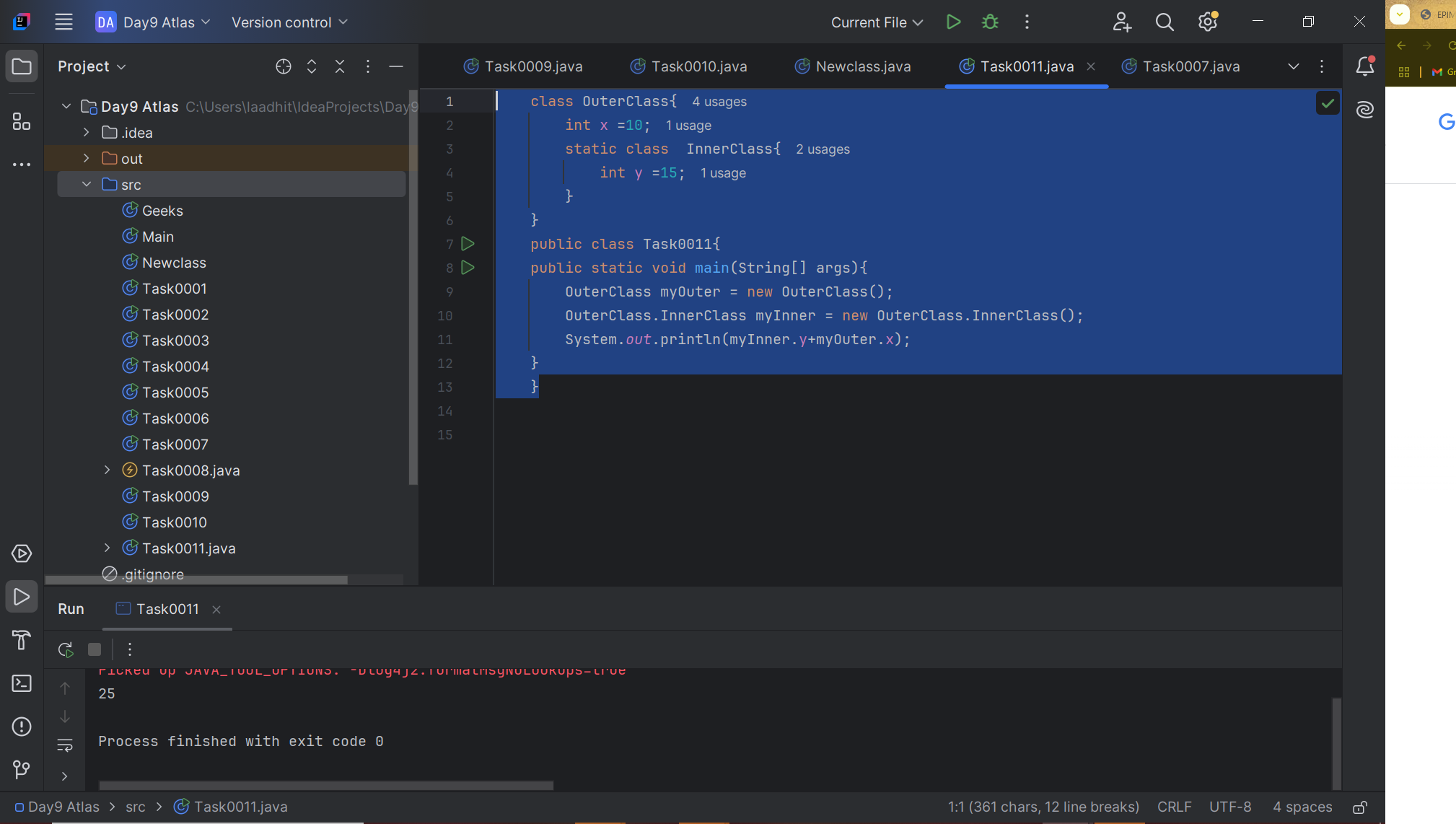
Task0012: Private class cannot be accessed

class OuterClass{  
 int x =10;  
 private class InnerClass{  
 int y =15;  
 }  
}  
public class Task0011{  
public static void main(String[] args){  
 OuterClass myOuter = new OuterClass();  
 OuterClass.InnerClass myInner = myOuter.new InnerClass();  
 System.*out*.println(myInner.y+myOuter.x);  
}  
}



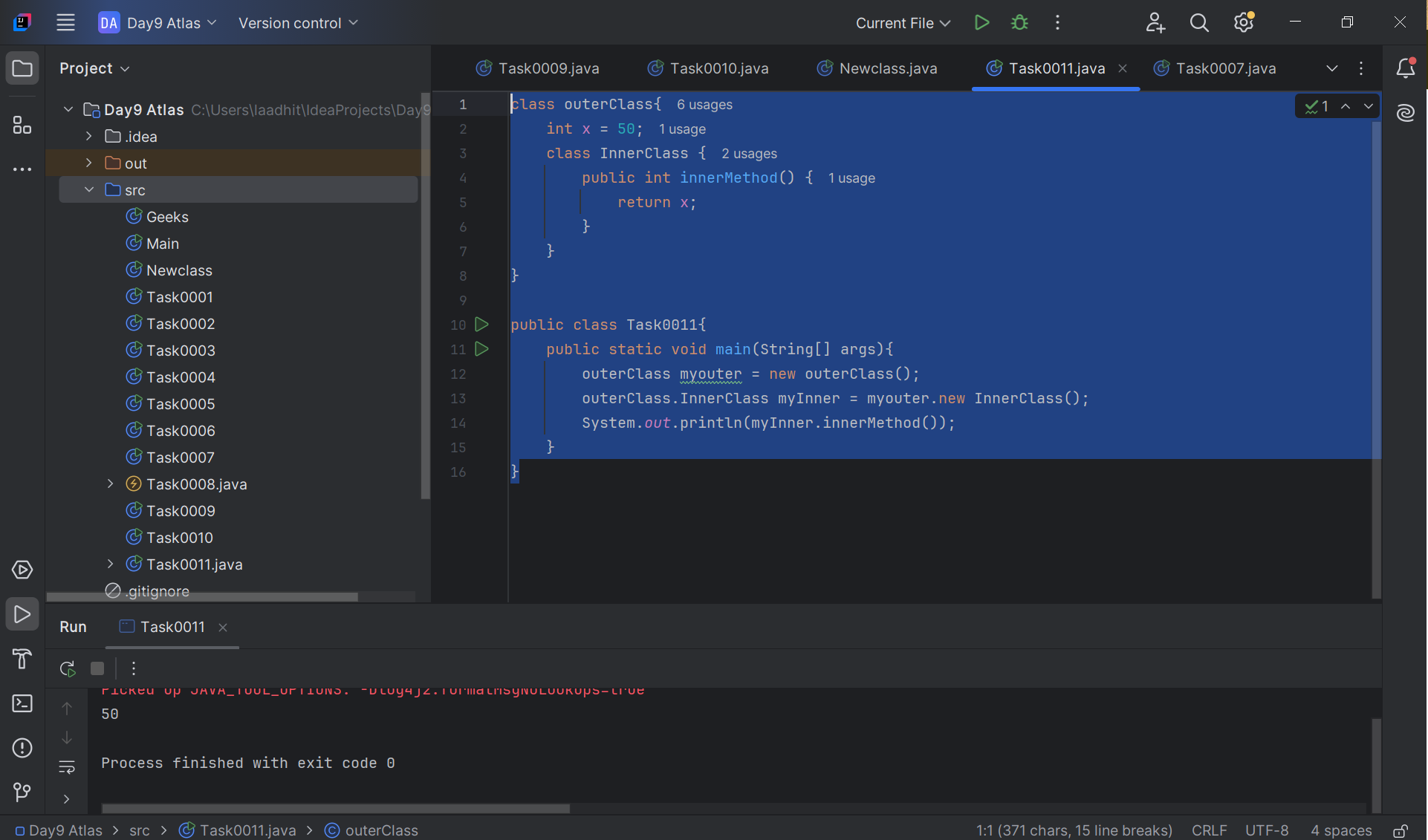
Task0013: Static

class OuterClass{  
 int x =10;  
 static class InnerClass{  
 int y =15;  
 }  
}  
public class Task0011{  
public static void main(String[] args){  
 OuterClass myOuter = new OuterClass();  
 OuterClass.InnerClass myInner = new OuterClass.InnerClass();  
 System.*out*.println(myInner.y+myOuter.x);  
}  
}



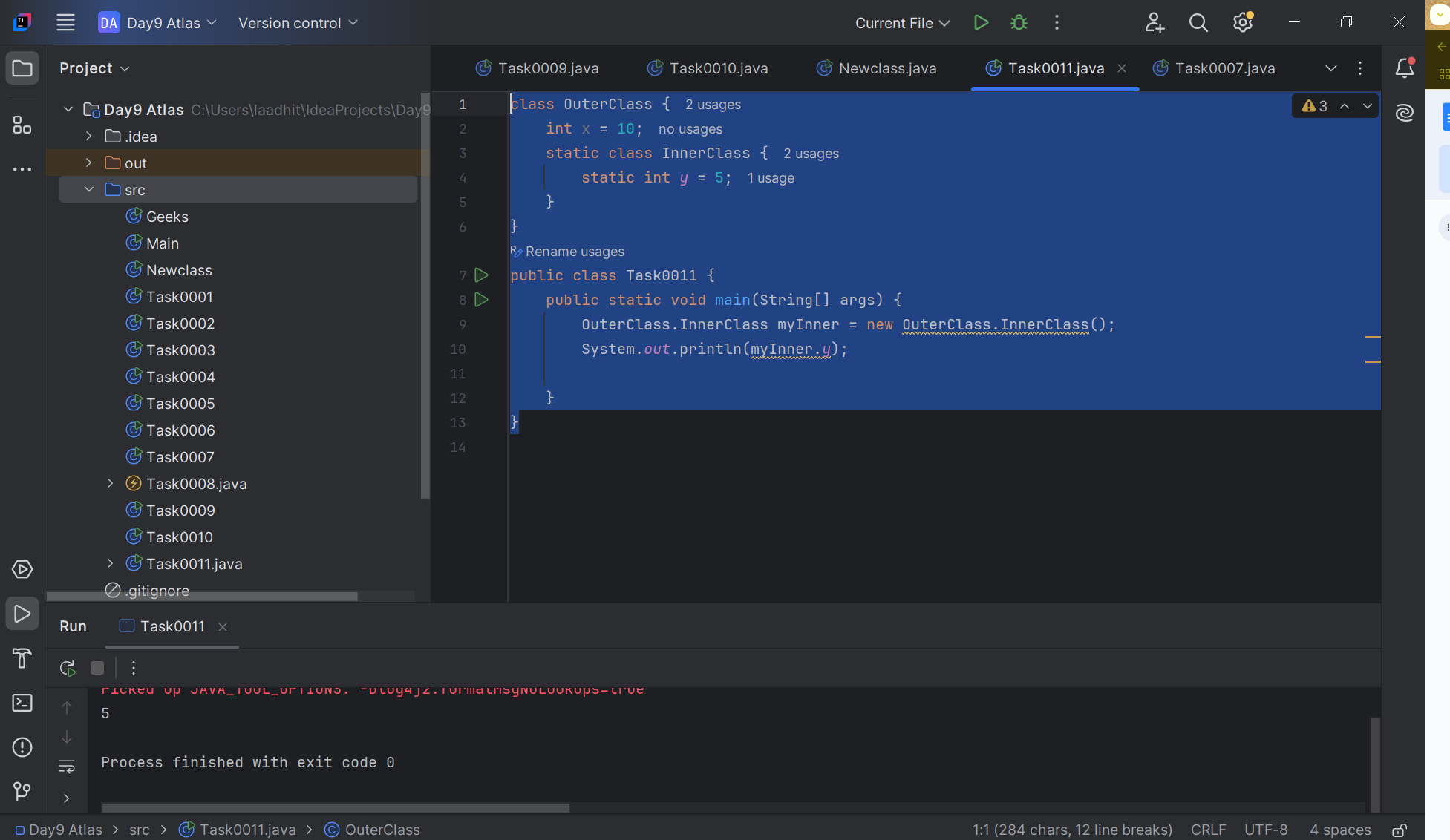
Task014:

class outerClass{  
 int x = 50;  
 class InnerClass {  
 public int innerMethod() {  
 return x;  
 }  
 }  
}  
  
public class Task0011{  
 public static void main(String[] args){  
 outerClass myouter = new outerClass();  
 outerClass.InnerClass myInner = myouter.new InnerClass();  
 System.*out*.println(myInner.innerMethod());  
 }  
}



Task015:

class OuterClass {  
 int x = 10;  
 static class InnerClass {  
 static int *y* = 5;  
 }  
}  
public class Task0011 {  
 public static void main(String[] args) {  
 OuterClass.InnerClass myInner = new OuterClass.InnerClass();  
 System.*out*.println(myInner.*y*);  
  
 }  
}



Task 0017:

Features of Java 8.

Core Enhancements:

* **Lambda Expressions:**

Enables functional programming by treating actions as objects. This allows for more concise and readable code, particularly when working with collections.

* **Functional Interfaces:**

Interfaces with a single abstract method, designed to be targets for lambda expressions.

* **Streams API:**

Provides a powerful way to process collections of data in a declarative manner, enabling operations like filtering, mapping, and reducing.

* **Method References:**

Allows referring to methods by their names, simplifying lambda expressions.

* **Optional Class:**

A container object that may or may not contain a non-null value, reducing the risk of NullPointerExceptions.

API Improvements:

* **Date and Time API:**

Introduces a new set of packages that provide a comprehensive date-time model, addressing long-standing issues with the older java.util.Date and java.util.Calendar classes.

* **Collection API Enhancements:**

Adds new methods to collections, such as forEach, removeIf, and replaceAll.

* **Concurrency API Improvements:**

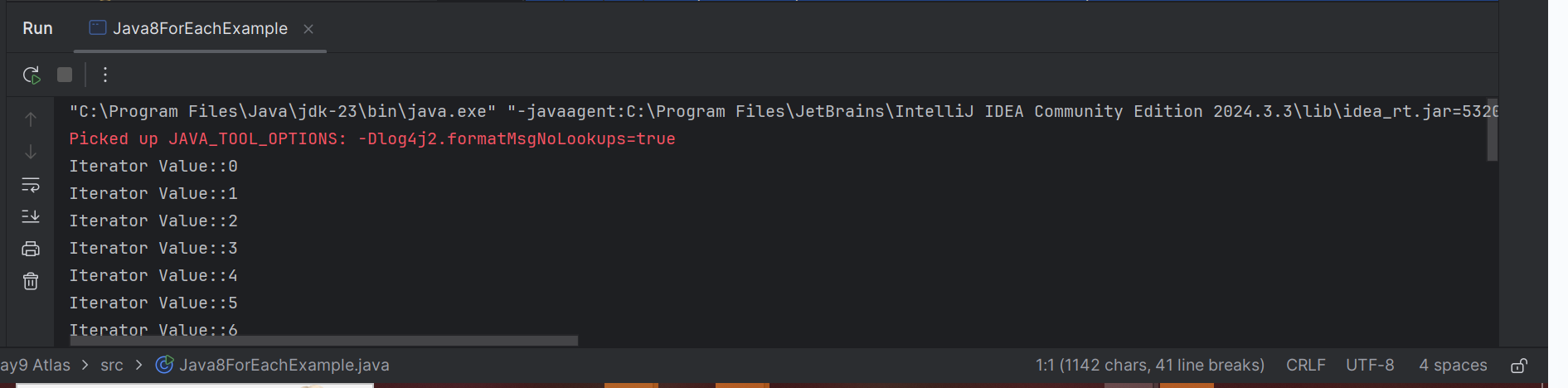
Includes new classes for parallel processing, such as CompletableFuture, enhancing support for concurrent operations.

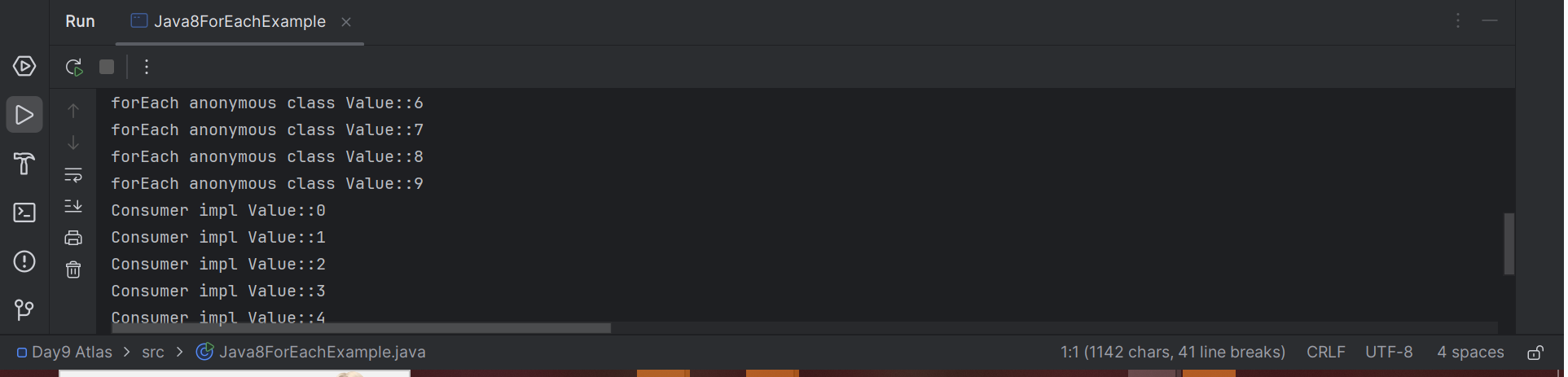
Other Notable Features:

* **Default and Static Methods in Interfaces:** Allows interfaces to have method implementations, enabling the evolution of interfaces without breaking existing implementations.
* **Nashorn JavaScript Engine:** Enables the execution and evaluation of JavaScript code within the JVM.
* **Parallel Array Sorting:** Enhances the efficiency of sorting arrays by leveraging multi-threading.

Task0018:

import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
import java.util.function.Consumer;  
import java.lang.Integer;  
  
 public class Java8ForEachExample {  
  
 public static void main(String[] args) {  
  
 List<Integer> myList = new ArrayList<Integer>();  
 for(int i=0; i<10; i++) myList.add(i);  
  
 Iterator<Integer> it = myList.iterator();  
 while(it.hasNext()){  
 Integer i = it.next();  
 System.*out*.println("Iterator Value::"+i);  
 }  
  
 myList.forEach(new Consumer<Integer>() {  
  
 public void accept(Integer t) {  
 System.*out*.println("forEach anonymous class Value::"+t);  
 }  
  
 });  
  
 //traversing with Consumer interface implementation  
 MyConsumer action = new MyConsumer();  
 myList.forEach(action);  
  
 }  
  
 }  
  
 //Consumer implementation that can be reused  
 class MyConsumer implements Consumer<Integer>{  
  
 public void accept(Integer t) {  
 System.*out*.println("Consumer impl Value::"+t);  
 }  
 }





Task0020:

import java.util.\*;  
 public class Task0020 {  
  
 public static void main(String[] args) {  
 // ArrayList  
 List a1 = new ArrayList();  
 a1.add("Zara");  
 a1.add("Mahnaz");  
 a1.add("Ayan");  
 System.*out*.println(" ArrayList Elements");  
 System.*out*.print("\t" + a1);  
  
 // LinkedList  
 List l1 = new LinkedList();  
 l1.add("Zara");  
 l1.add("Mahnaz");  
 l1.add("Ayan");  
 System.*out*.println();  
 System.*out*.println(" LinkedList Elements");  
 System.*out*.print("\t" + l1);  
  
 // HashSet  
 Set s1 = new HashSet();  
 s1.add("Zara");  
 s1.add("Mahnaz");  
 s1.add("Ayan");  
 System.*out*.println();  
 System.*out*.println(" Set Elements");  
 System.*out*.print("\t" + s1);  
  
 // HashMap  
 Map m1 = new HashMap();  
 m1.put("Zara", "8");  
 m1.put("Mahnaz", "31");  
 m1.put("Ayan", "12");  
 m1.put("Daisy", "14");  
 System.*out*.println();  
 System.*out*.println(" Map Elements");  
 System.*out*.print("\t" + m1);  
 }  
 }

