Day 9

Class ----- > class === extends

Interface -------- > class ===implement

Interface ---------- >interface == extends

Exceptions:

2 catergories==== >

- checked …>complile tim(at the time of writing the code)

- unchecked…..>runtime time (occure at the time of executing the code)

Pre defeined class exceptions:

Filenotfound

Arrayindexoutofbounce exception

Nullpointer exceptions

Arthmetic Exeptions:

User defined exceptions: (creating our own exceptions)

Taking care of exceptions:

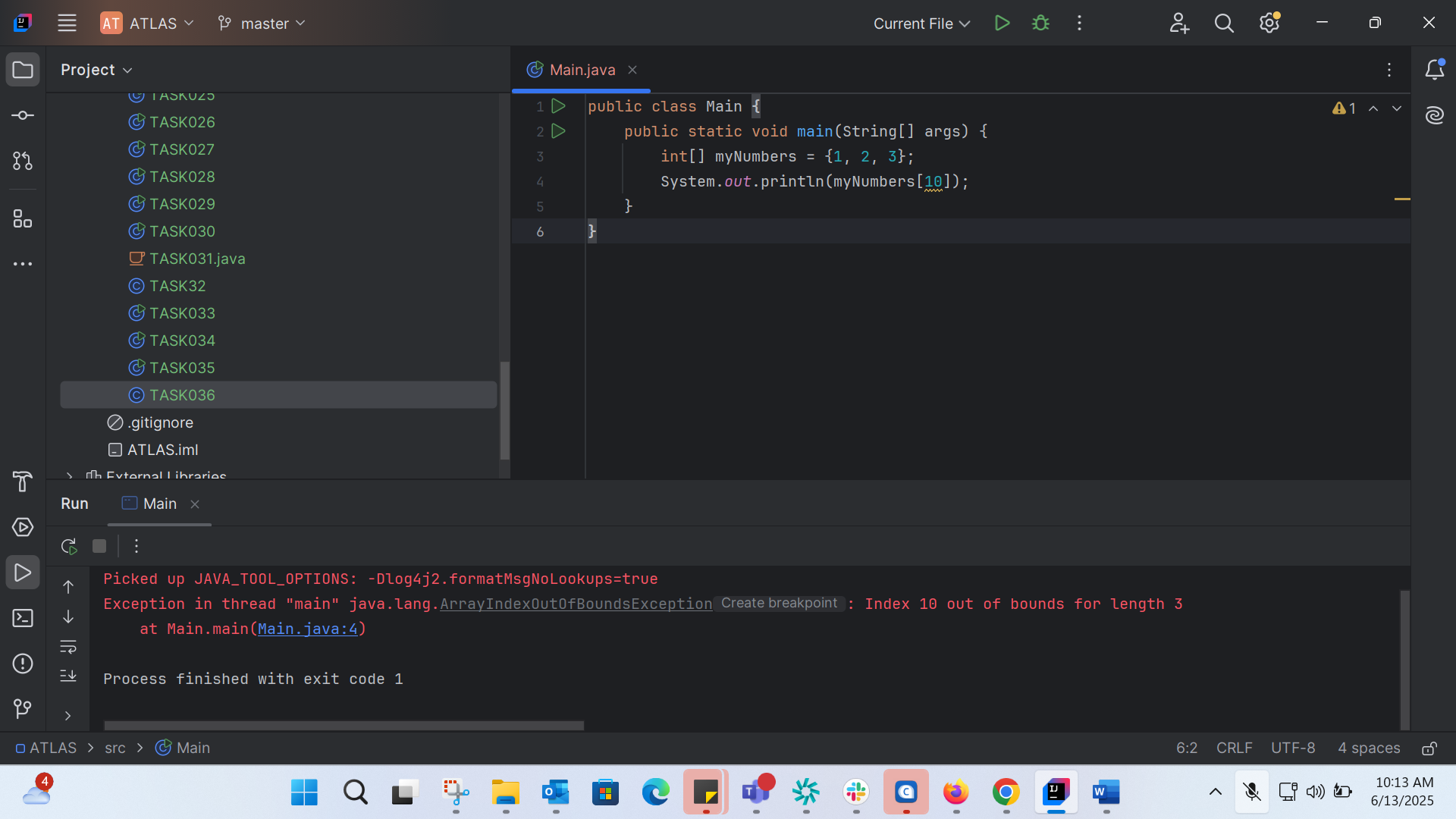
try{

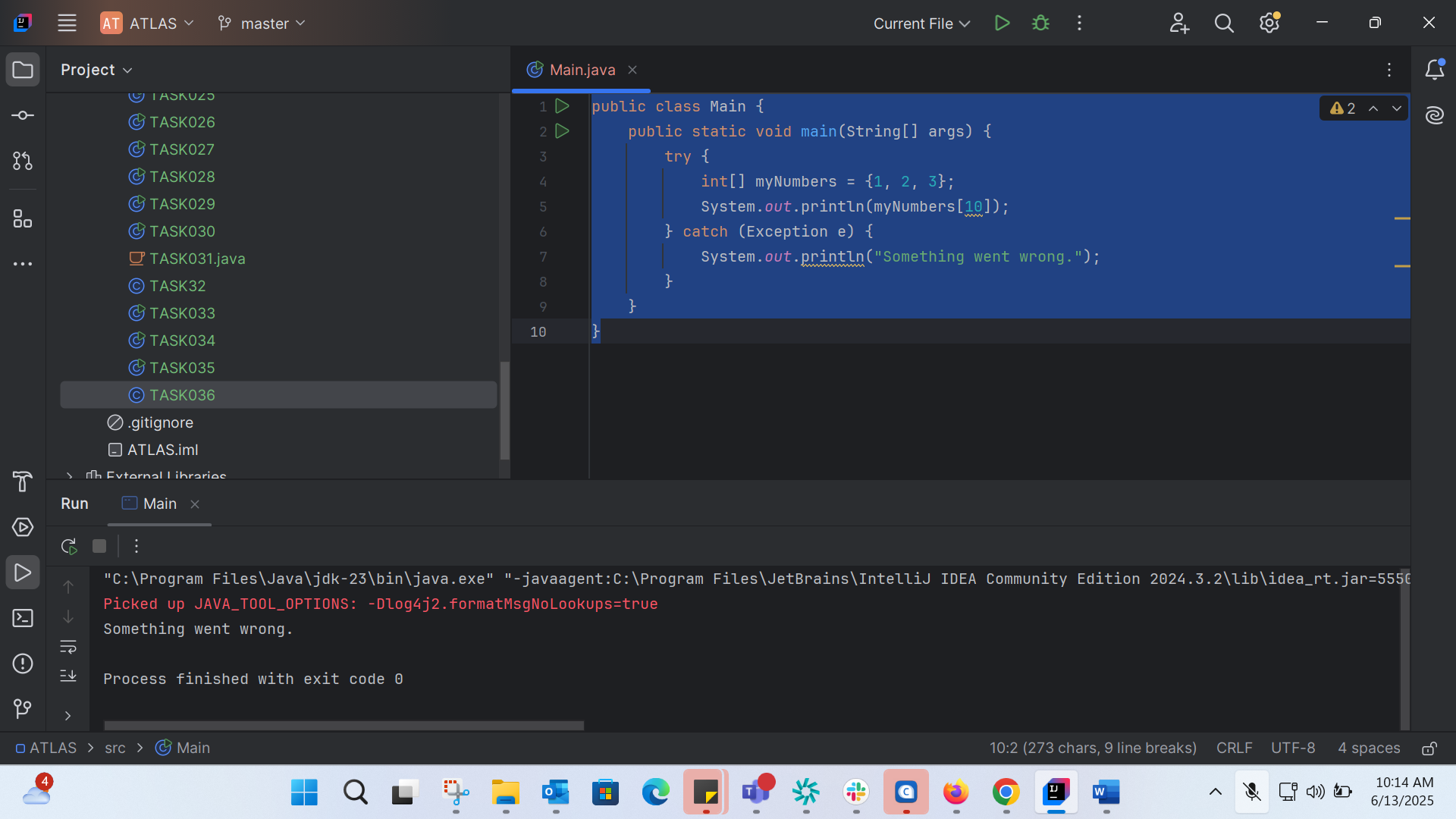
  }catch(Exception ex){

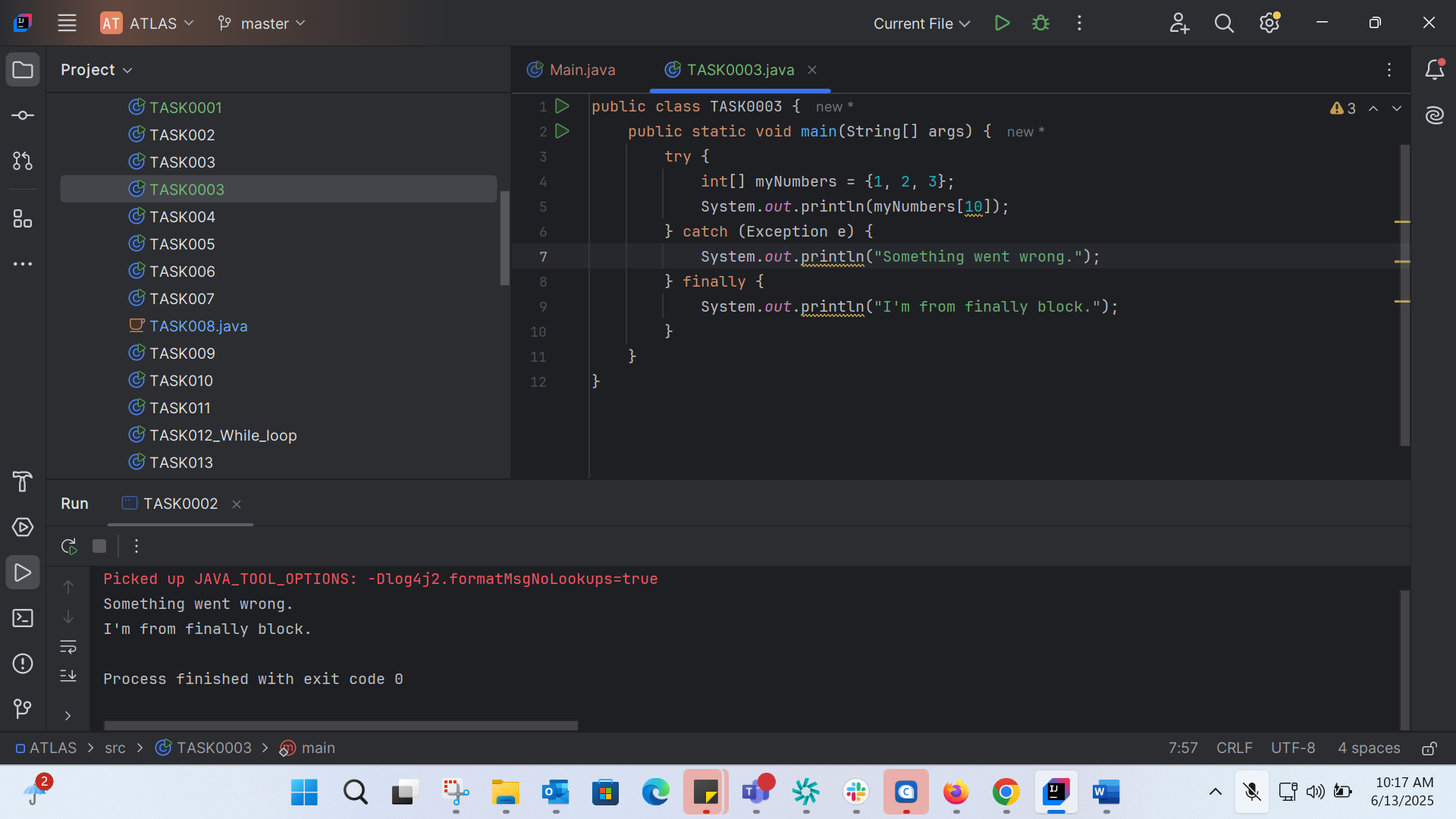
  }catch(ArthematicException ex1{  
  }  
  finally{

  sout(" ");  
  }

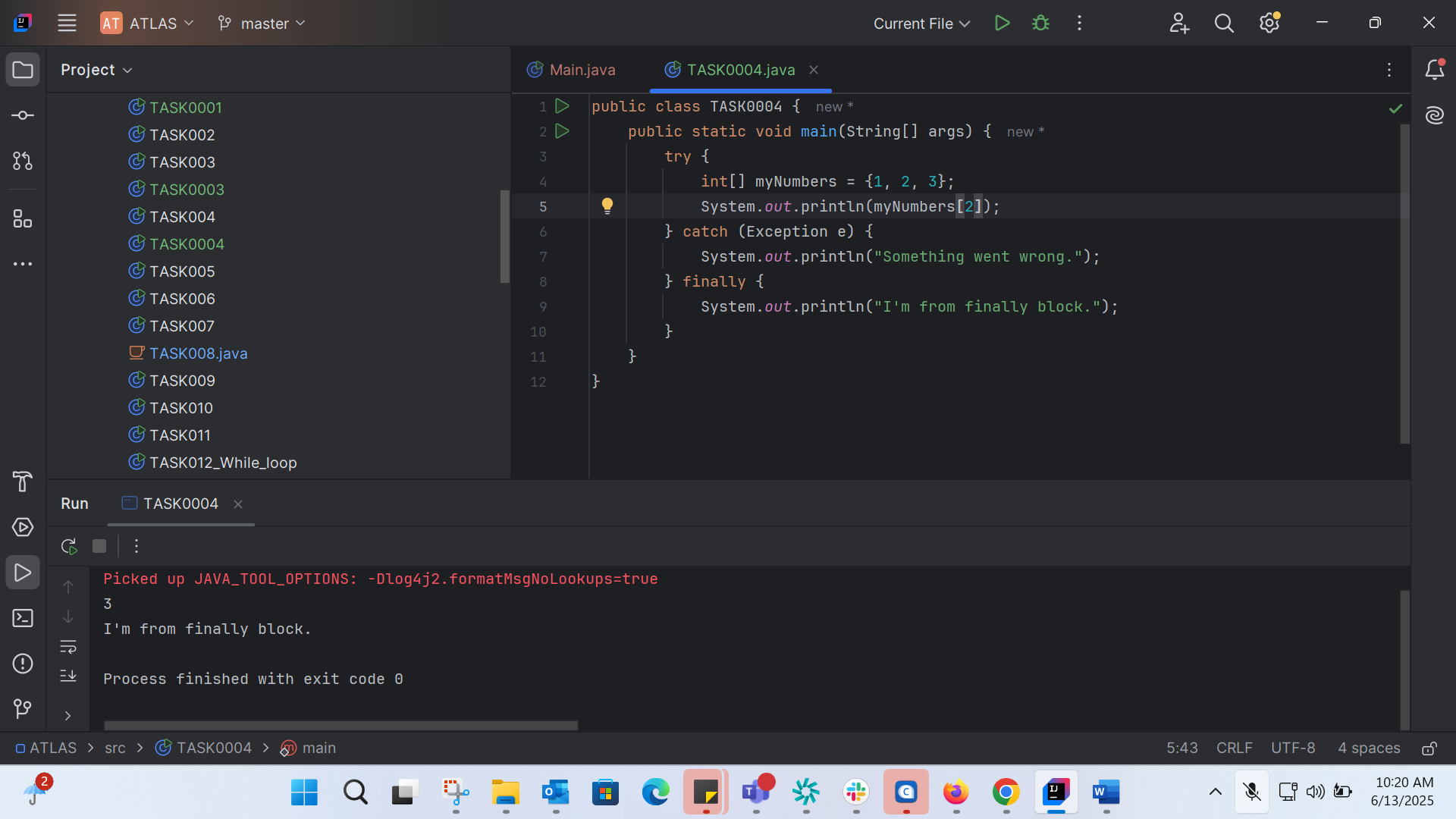
TASK 001:

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

TASK 002: Correct the above code: 

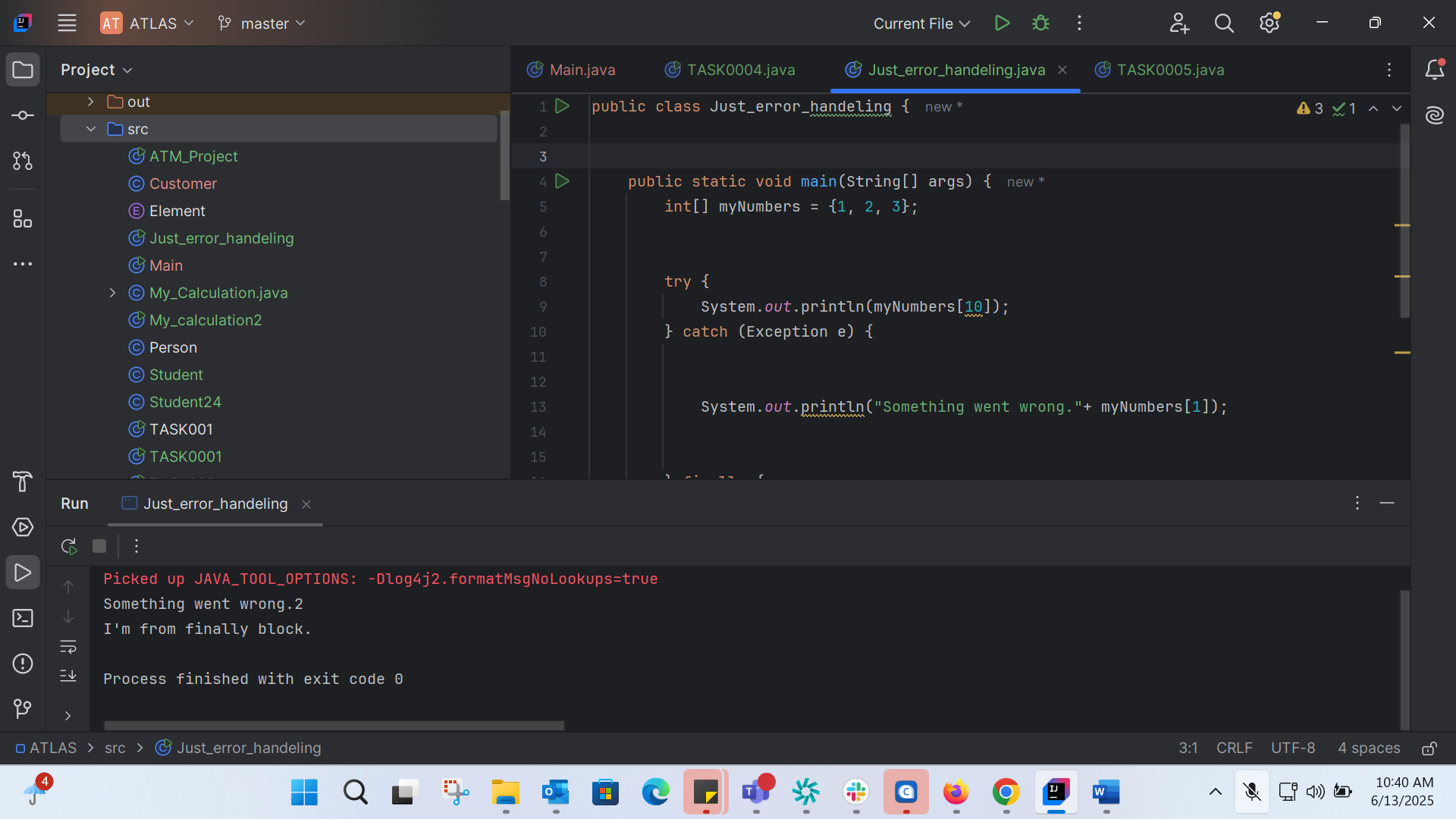
TASK0003: 

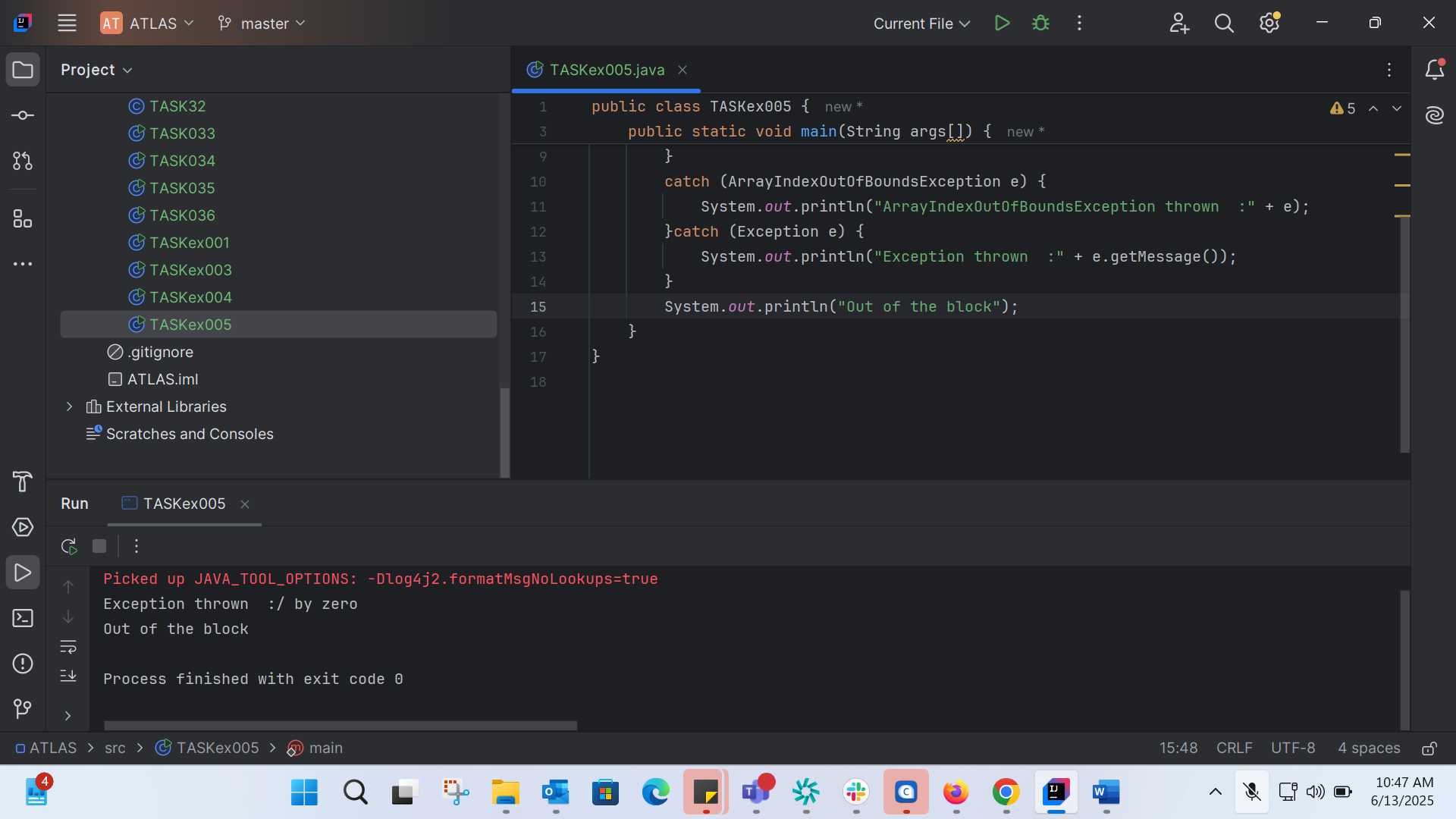
public class TASK0003 {  
 public static void main(String[] args) {  
 try {  
 int[] myNumbers = {1, 2, 3};  
 System.*out*.println(myNumbers[10]);  
 } catch (Exception e) {  
 System.*out*.println("Something went wrong.");  
 } finally {  
 System.*out*.println("I'm from finally block.");  
 }  
 }  
}

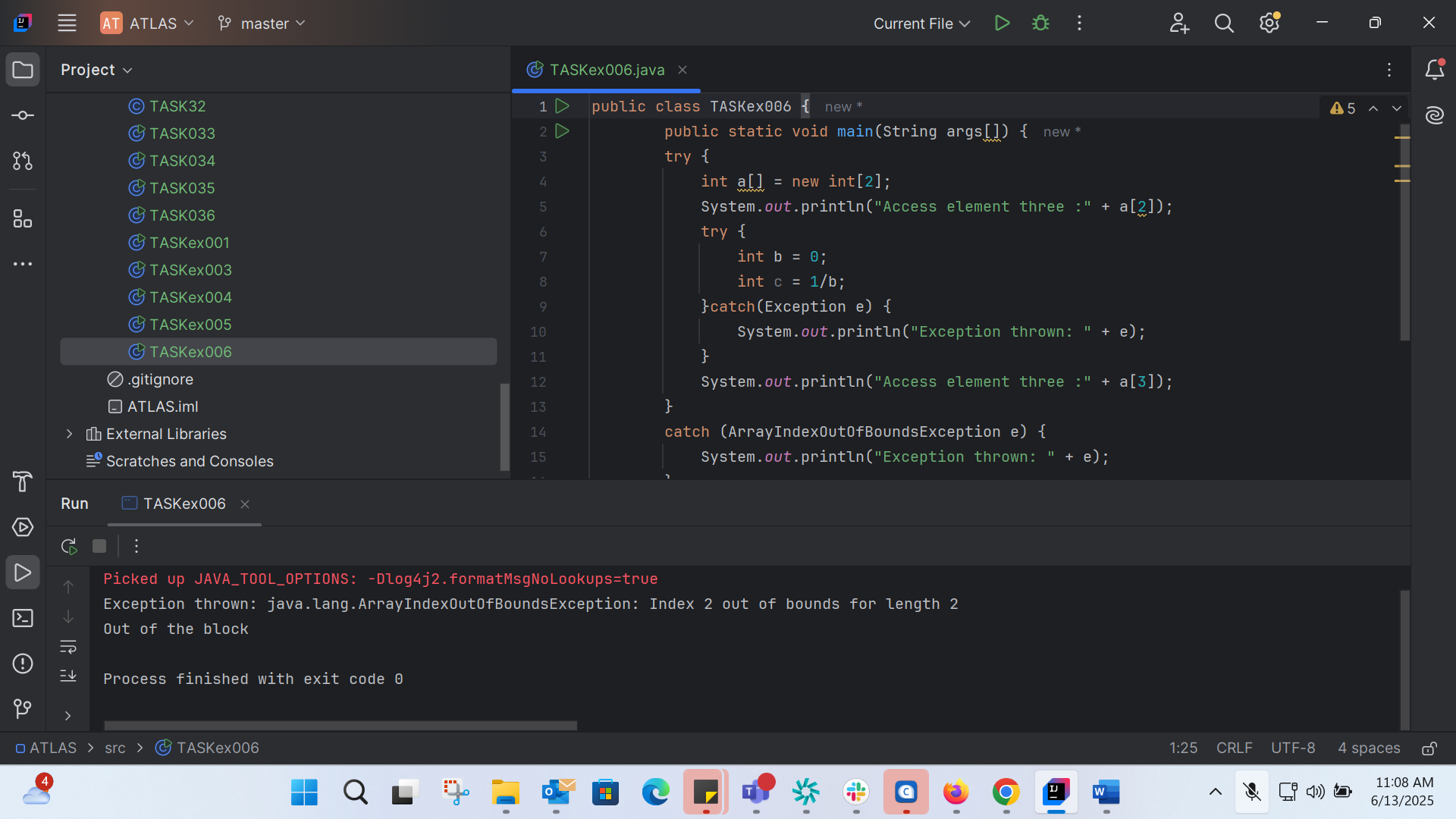
TASK0004: 

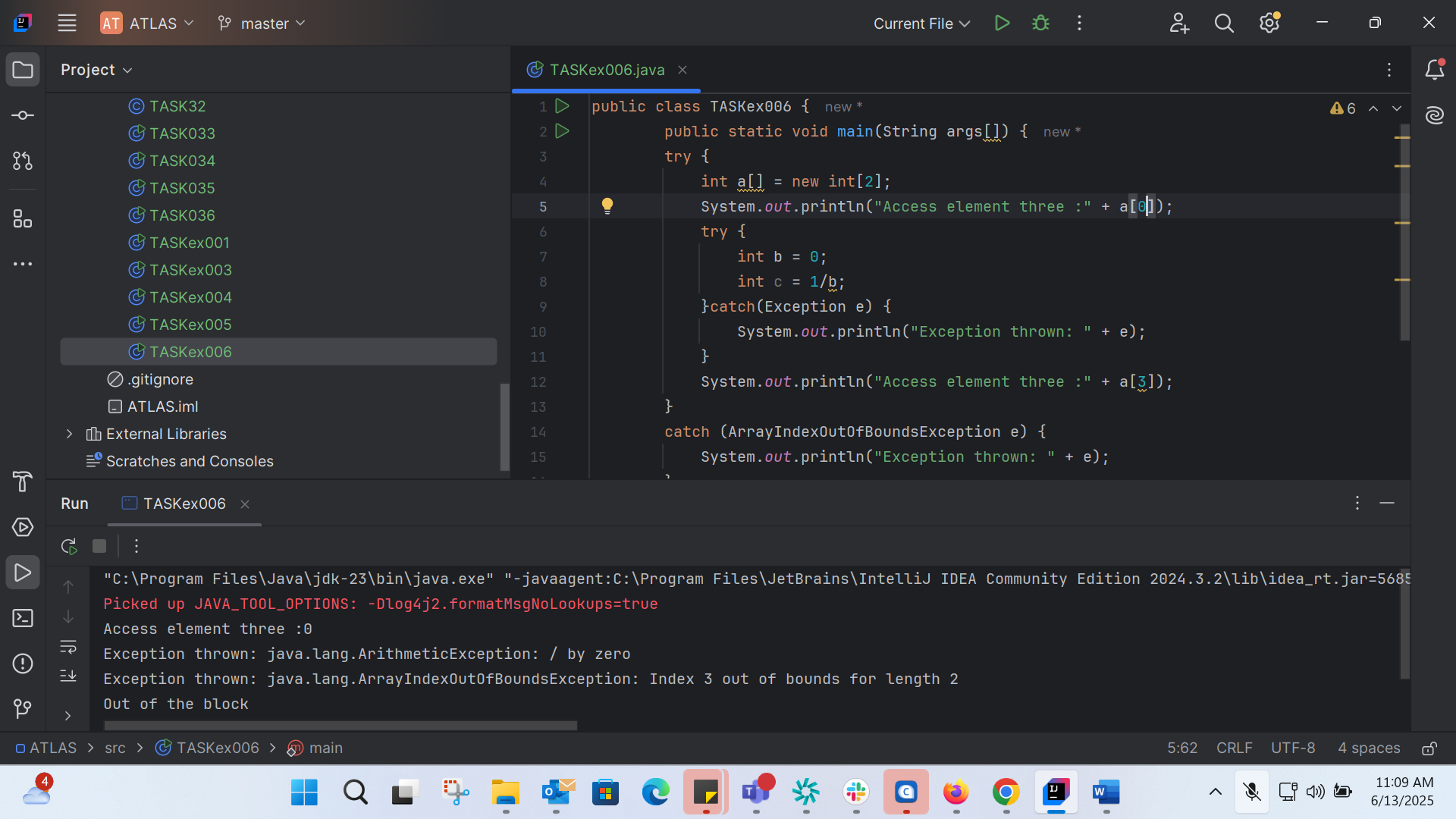
public class TASK0004 {  
 public static void main(String[] args) {  
 try {  
 int[] myNumbers = {1, 2, 3};  
 System.*out*.println(myNumbers[2]);  
 } catch (Exception e) {  
 System.*out*.println("Something went wrong.");  
 } finally {  
 System.*out*.println("I'm from finally block.");  
 }  
 }  
}

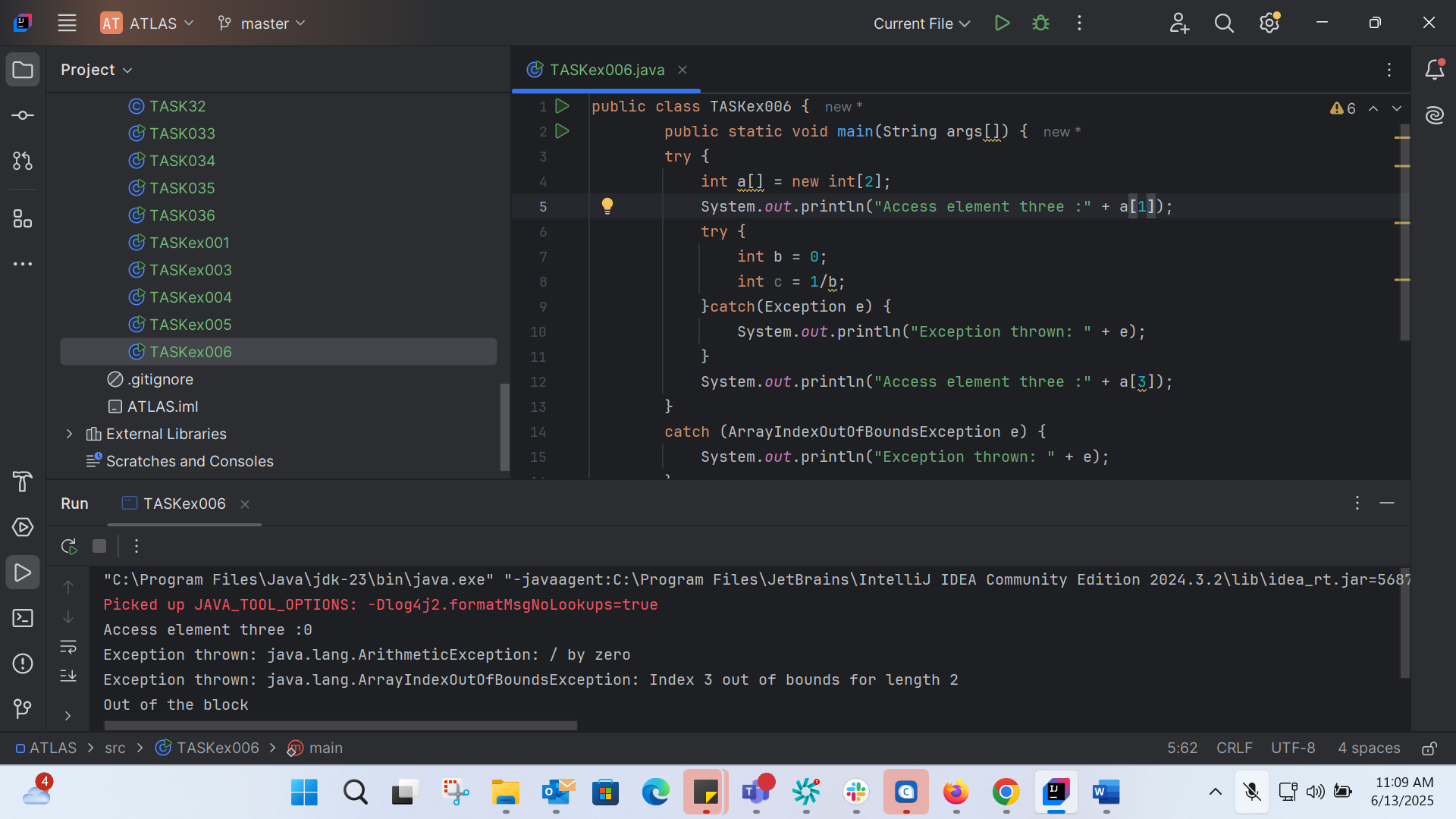
Error Handeling:



TASK0005: 

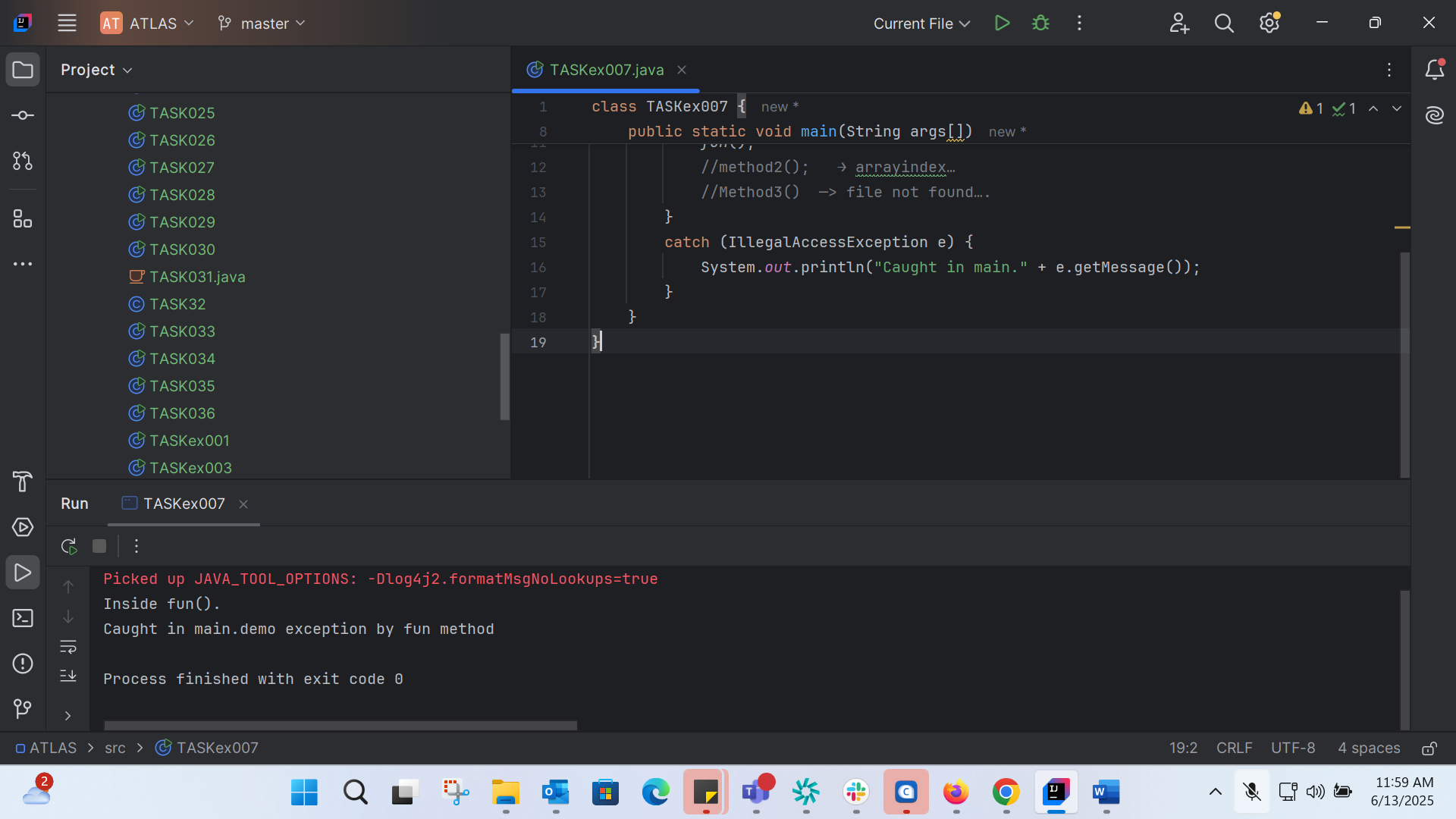
TASK006: 

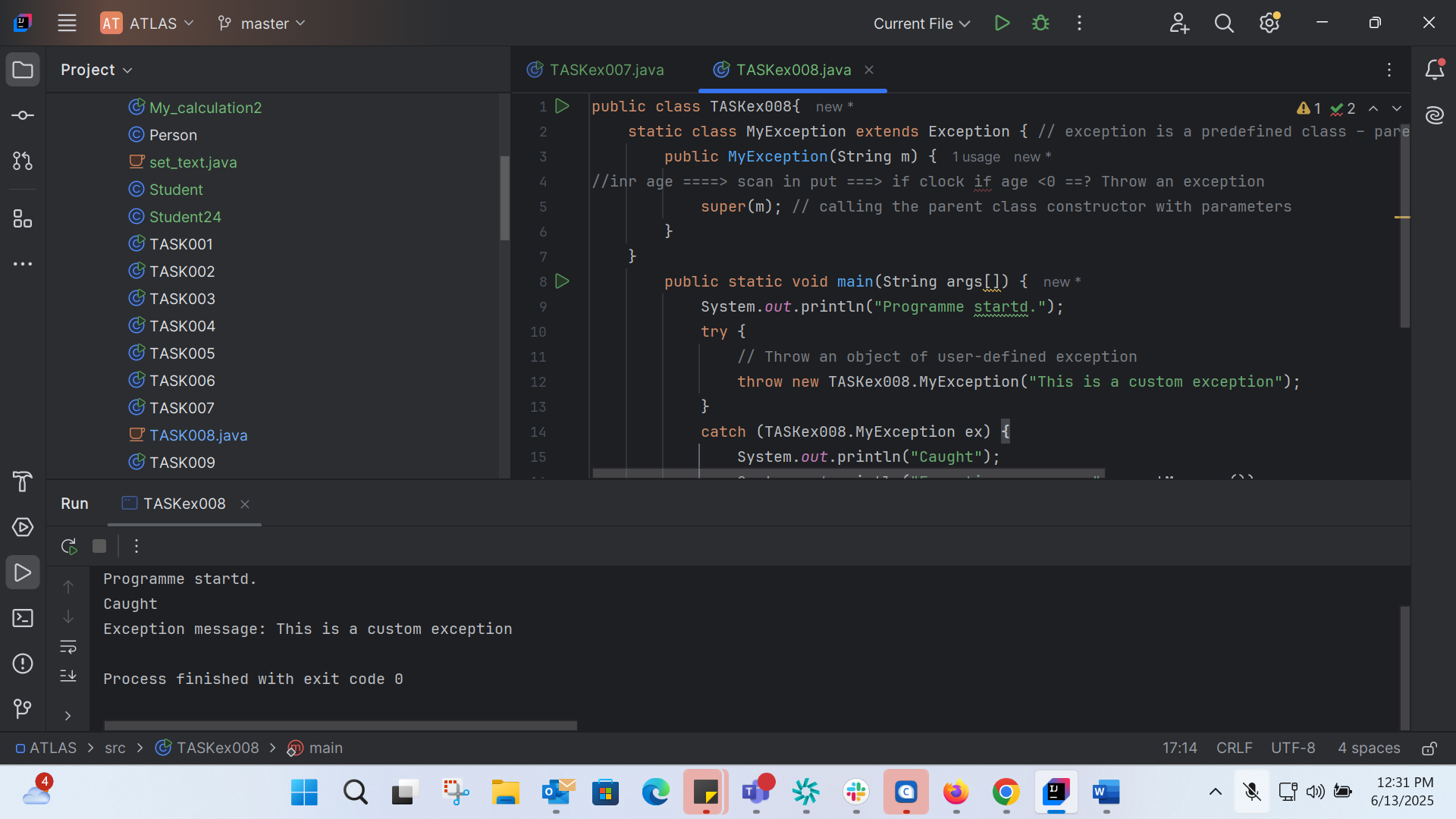
Accessing 0 index: 

Accessing 1: 

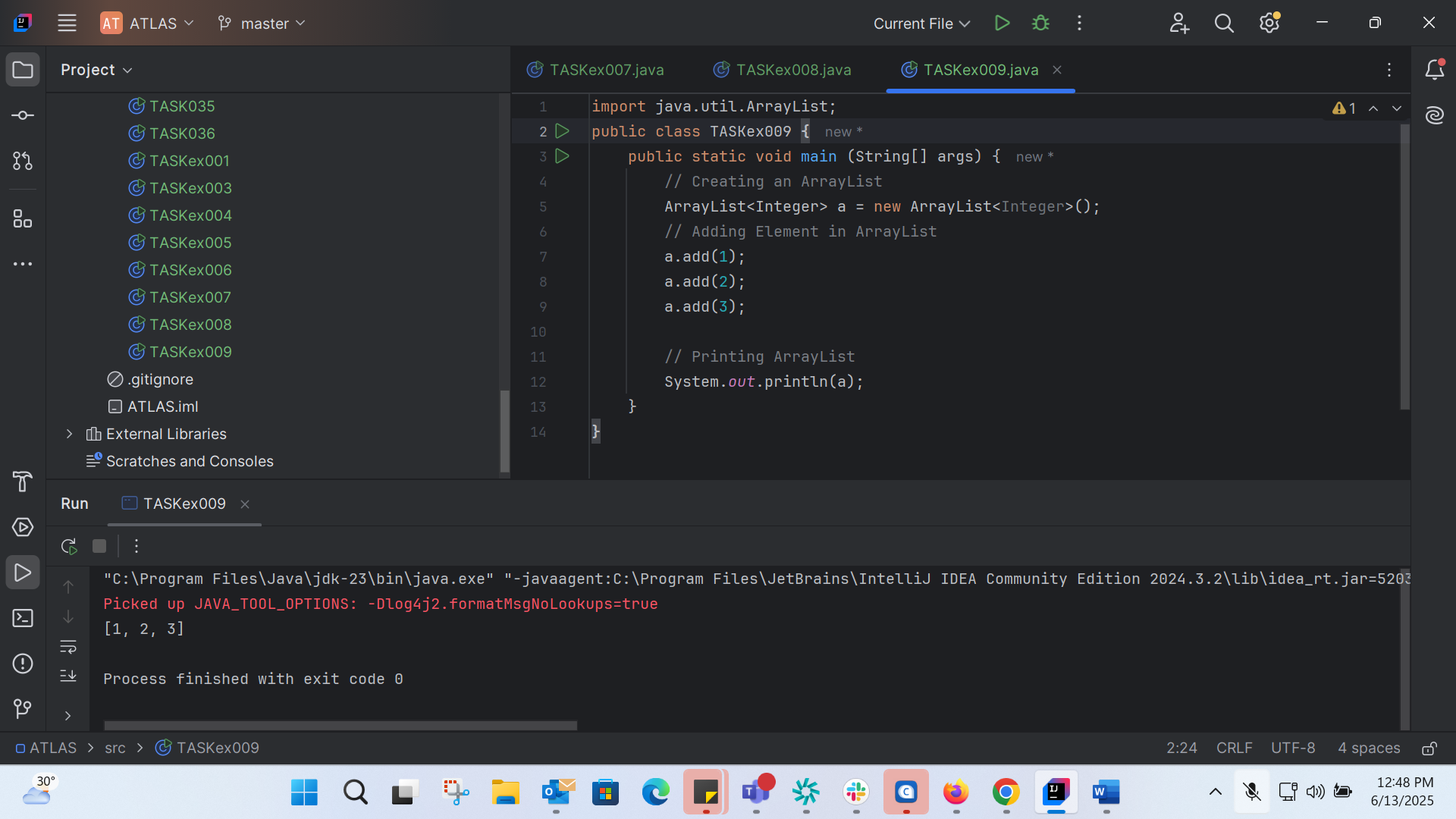
TASK007:

class TASKex007 {  
 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());  
 }  
 }  
}

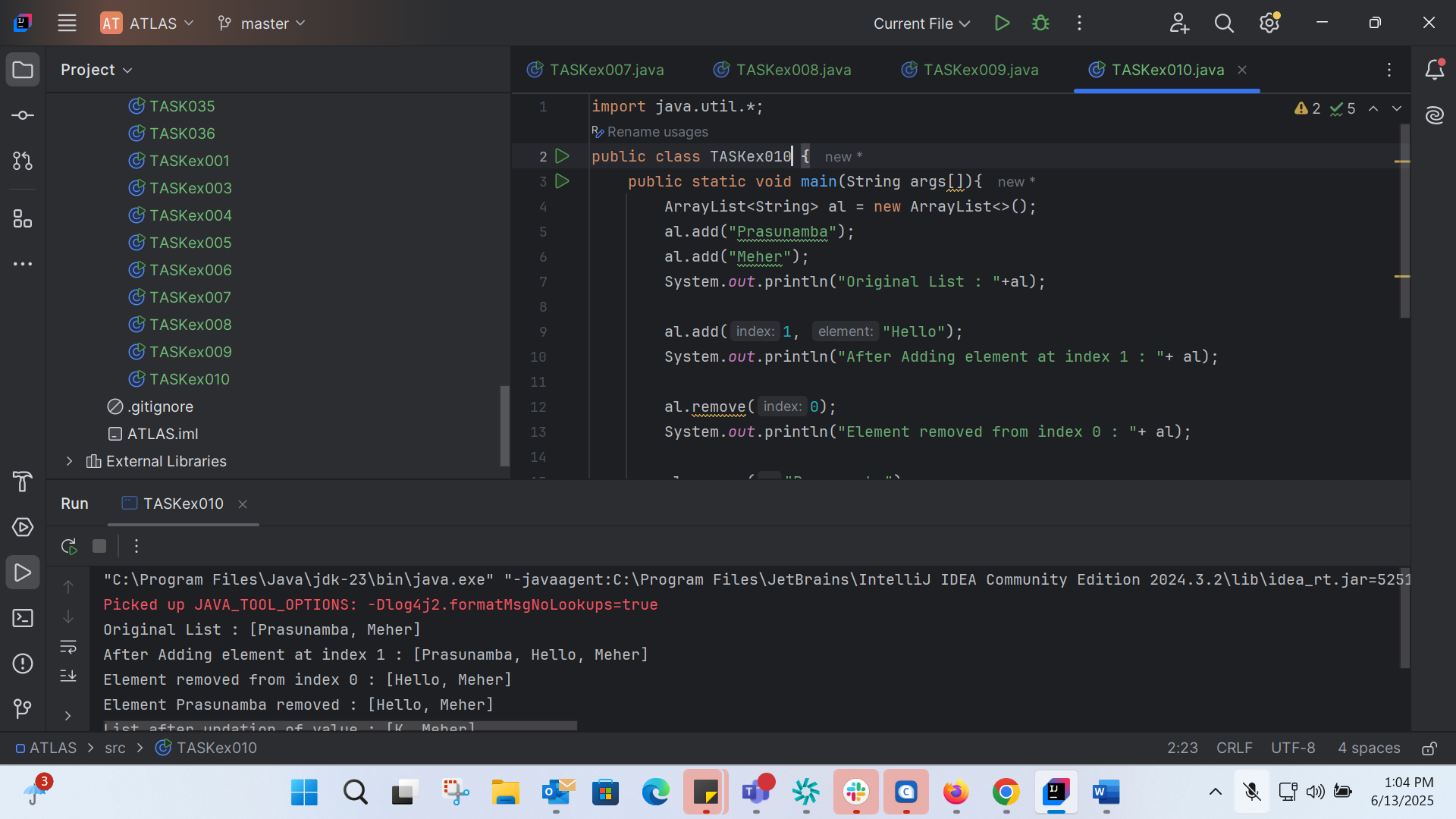


TASK008: 

public class TASKex008{  
 static class MyException extends Exception { // exception is a predefined class – parent class for MyException  
 public MyException(String m) {  
//inr age ====> scan in put ===> if clock if age <0 ==? Throw an exception  
 super(m); // calling the parent class constructor with parameters  
 }  
 }  
 public static void main(String args[]) {  
 System.*out*.println("Programme startd.");  
 try {  
 // Throw an object of user-defined exception  
 throw new TASKex008.MyException("This is a custom exception");  
 }  
 catch (TASKex008.MyException ex) {  
 System.*out*.println("Caught");  
 System.*out*.println("Exception message: "+ex.getMessage());  
 }  
 }  
 }

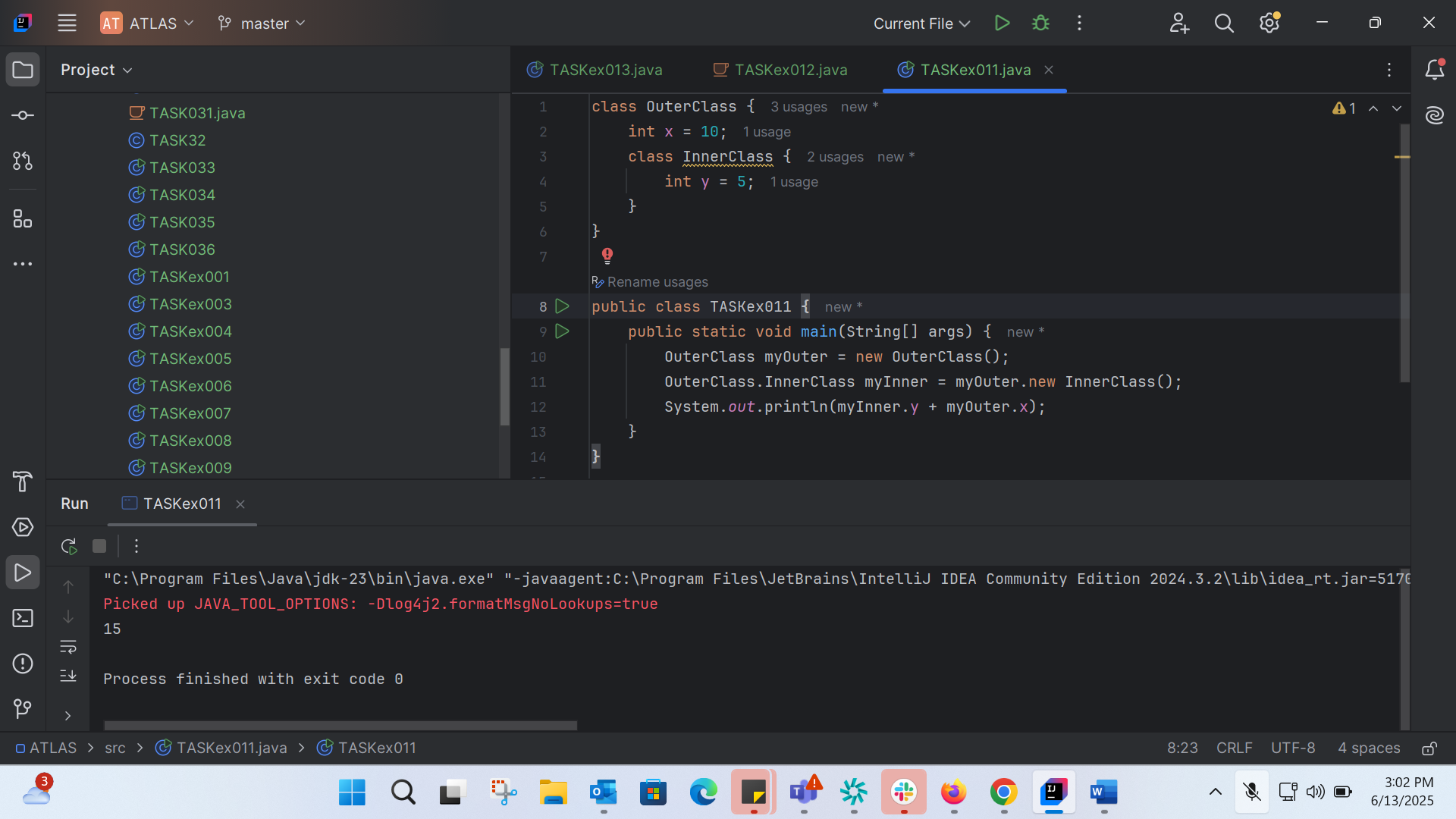
TASK09: 

import java.util.ArrayList;  
public class TASKex009 {  
 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);  
  
 // Printing ArrayList  
 System.*out*.println(a);  
 }  
}

TASK 10: 

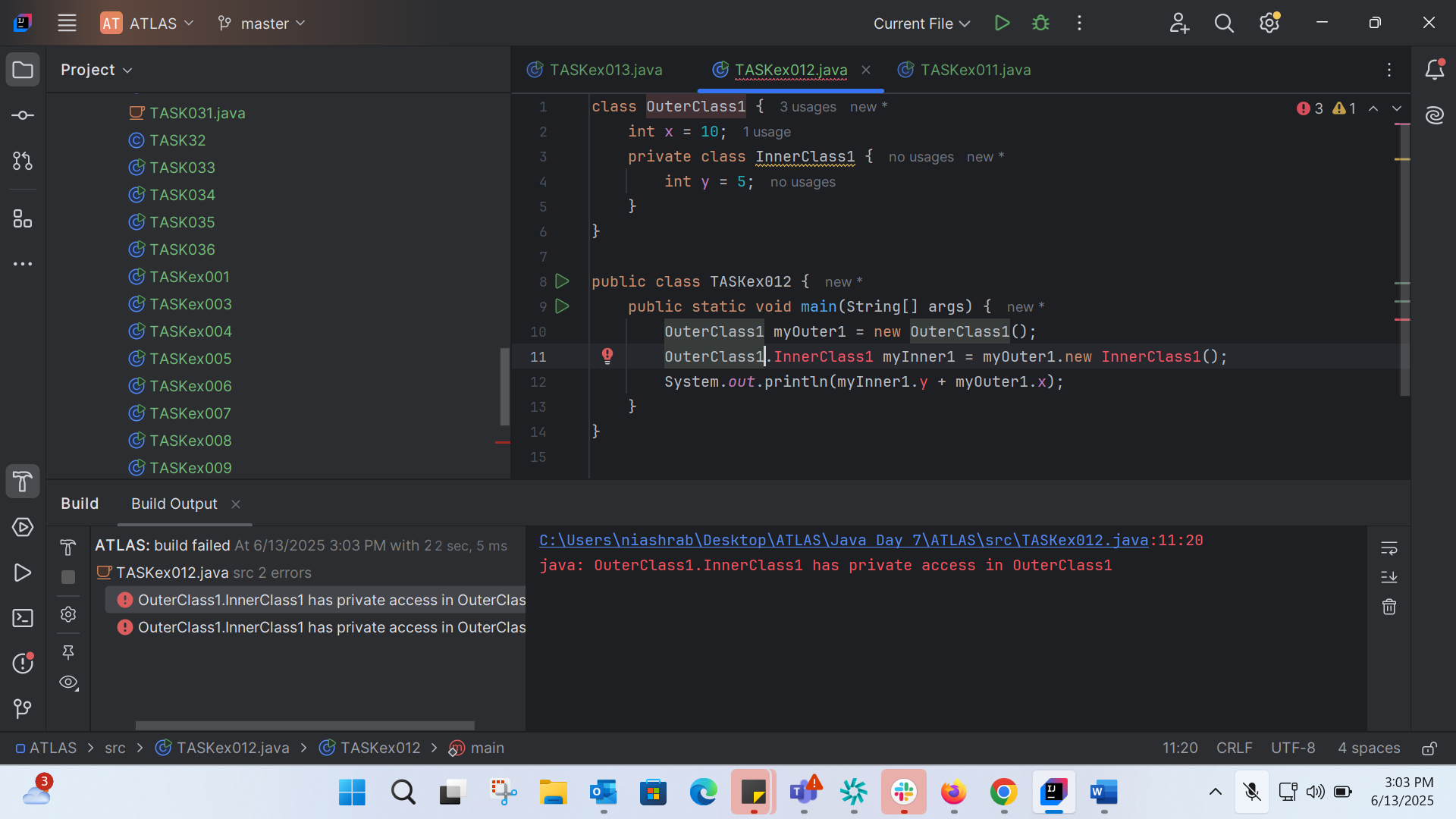
import java.util.\*;  
public class TASKex010 {  
 public static void main(String args[]){  
 ArrayList<String> al = new ArrayList<>();  
 al.add("Prasunamba");  
 al.add("Meher");  
 System.*out*.println("Original List : "+al);  
  
 al.add(1, "Hello");  
 System.*out*.println("After Adding element 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);  
 }  
}

TASK 11:

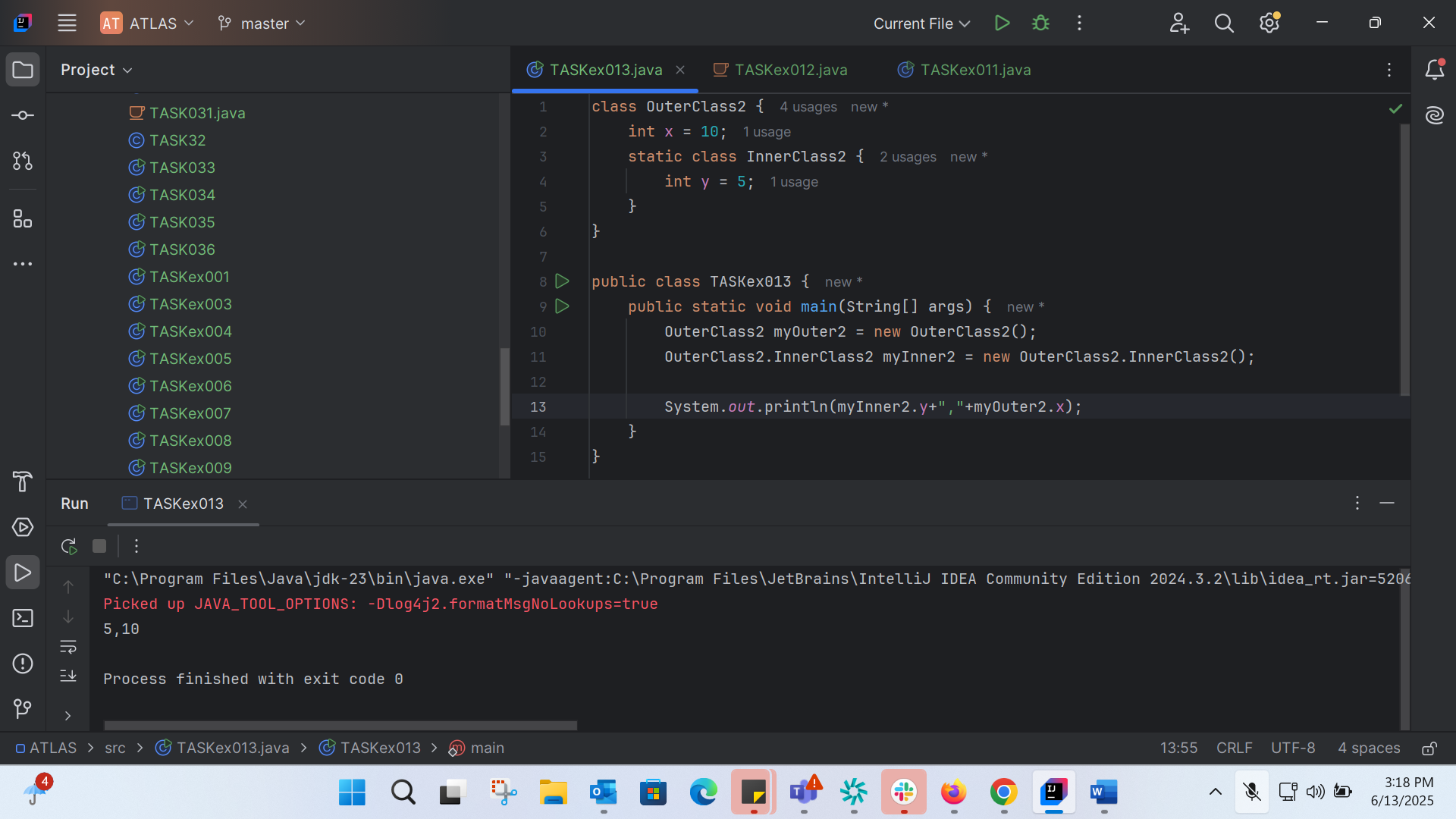


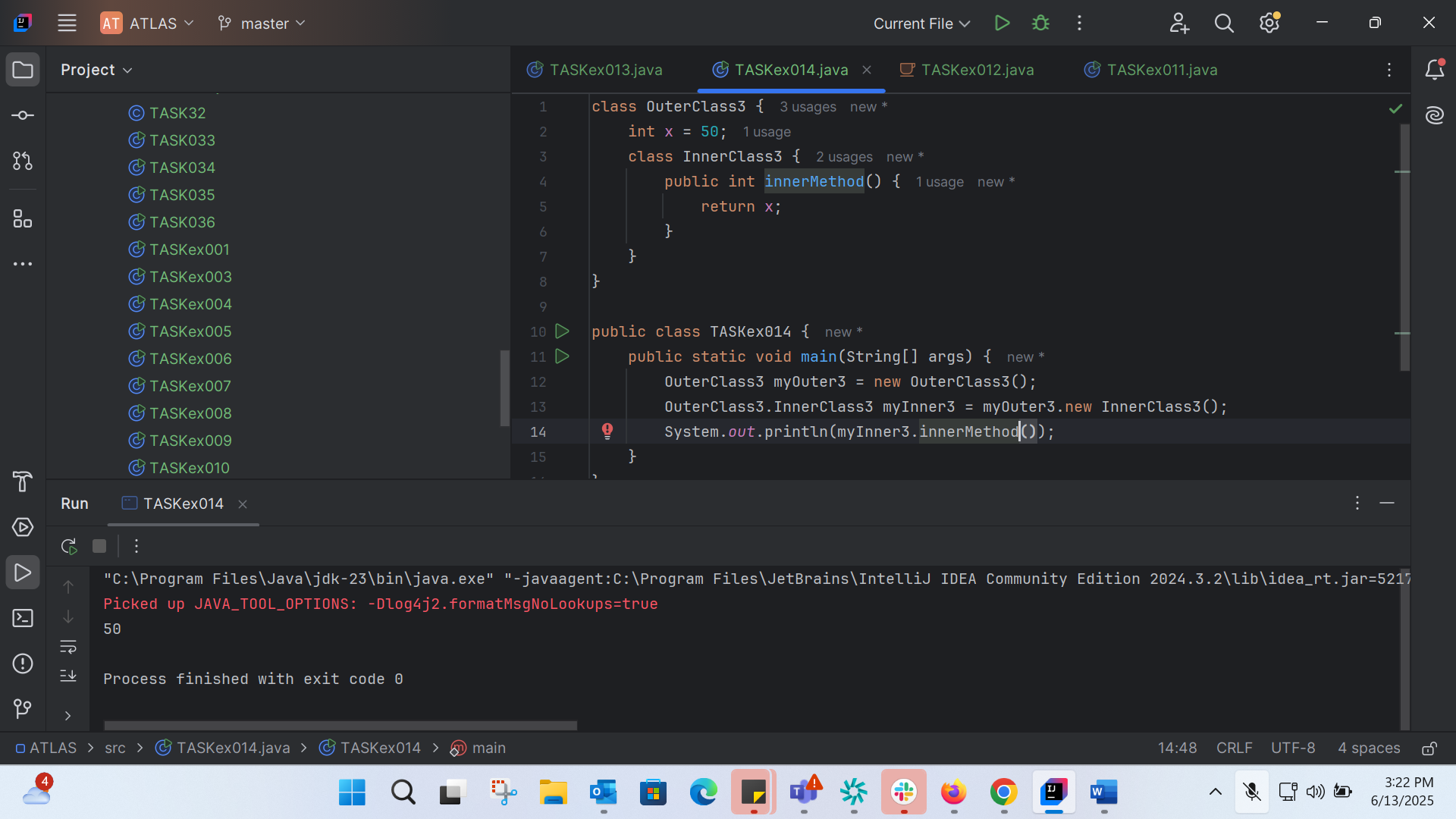
class OuterClass {  
 int x = 10;  
 class InnerClass {  
 int y = 5;  
 }  
}  
  
public class TASKex011 {  
 public static void main(String[] args) {  
 OuterClass myOuter = new OuterClass();  
 OuterClass.InnerClass myInner = myOuter.new InnerClass();  
 System.*out*.println(myInner.y + myOuter.x);  
 }  
}

TASK 12:



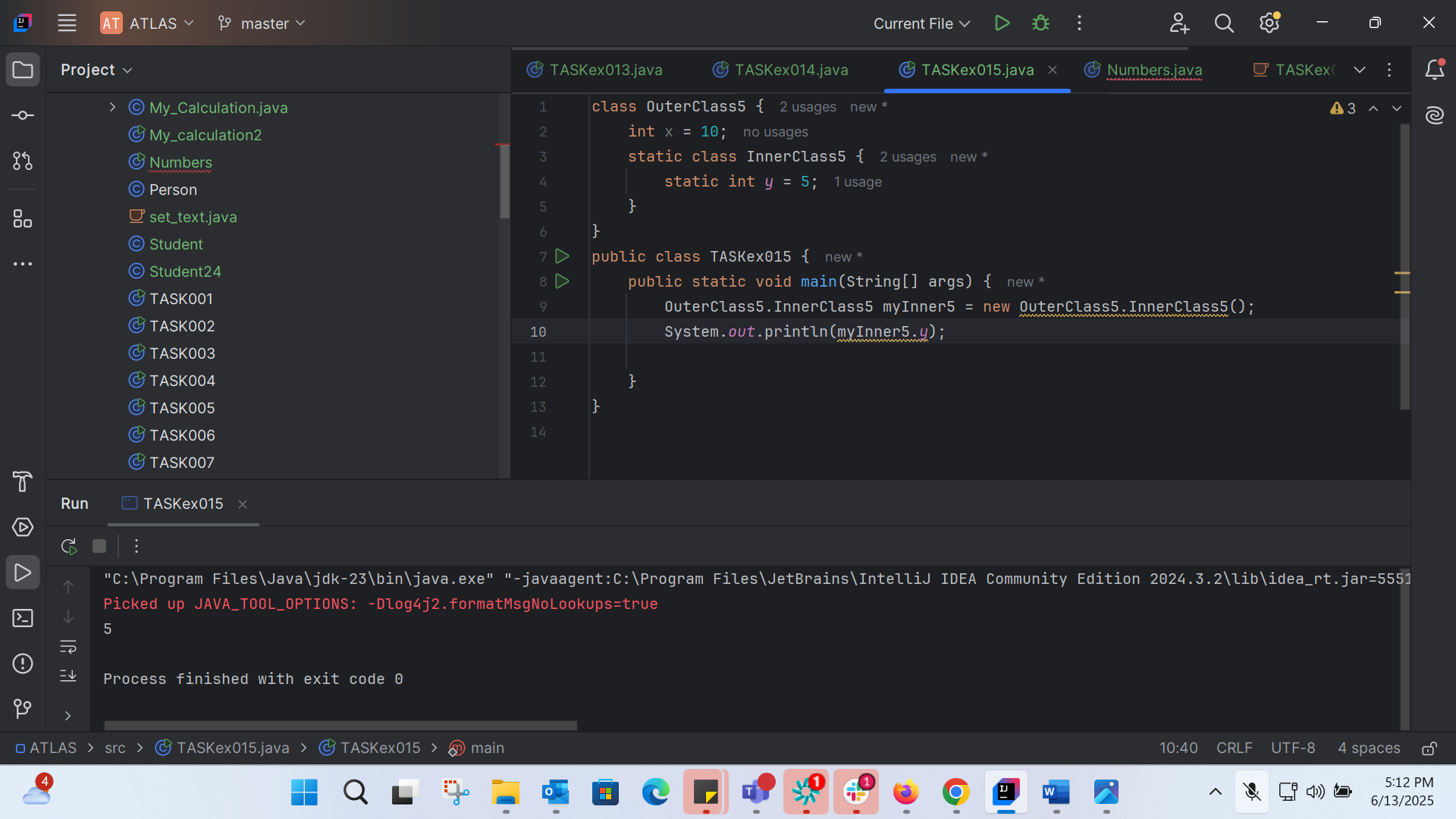
class OuterClass1 {  
 int x = 10;  
 private class InnerClass1 {  
 int y = 5;  
 }  
}  
  
public class TASKex012 {  
 public static void main(String[] args) {  
 OuterClass1 myOuter1 = new OuterClass1();  
 OuterClass1.InnerClass1 myInner1 = myOuter1.new InnerClass1();  
 System.*out*.println(myInner1.y + myOuter1.x);  
 }  
}

TASK 13: 

TASK 14: 

class OuterClass3 {  
 int x = 50;  
 class InnerClass3 {  
 public int innerMethod() {  
 return x;  
 }  
 }  
}  
  
public class TASKex014 {  
 public static void main(String[] args) {  
 OuterClass3 myOuter3 = new OuterClass3();  
 OuterClass3.InnerClass3 myInner3 = myOuter3.new InnerClass3();  
 System.*out*.println(myInner3.innerMethod());  
 }  
}

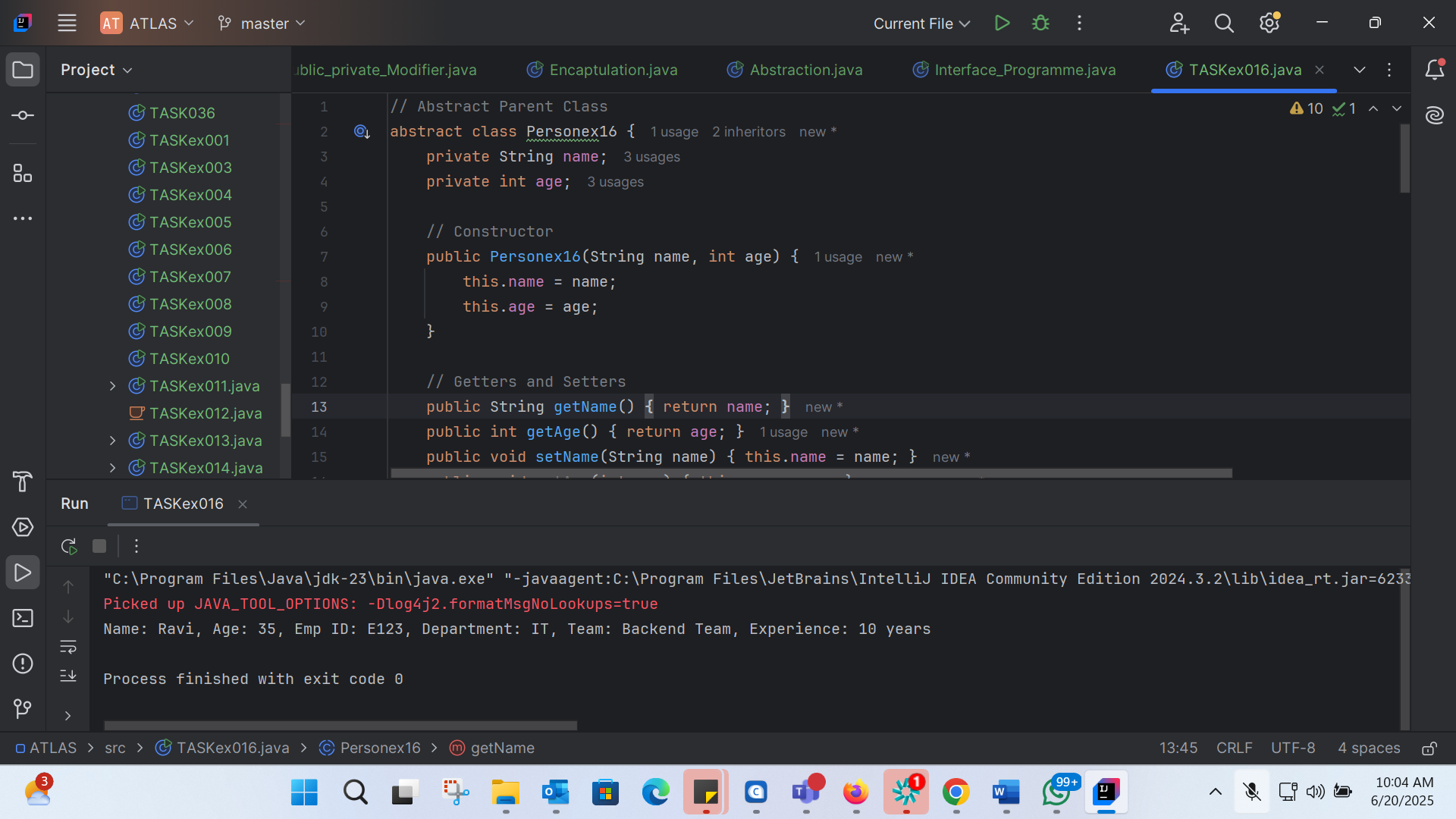
TASK015:



class OuterClass5 {  
 int x = 10;  
 static class InnerClass5 {  
 static int *y* = 5;  
 }  
}  
public class TASKex015 {  
 public static void main(String[] args) {  
 OuterClass5.InnerClass5 myInner5 = new OuterClass5.InnerClass5();  
 System.*out*.println(myInner5.*y*);  
  
 }  
}

TASK 16 :

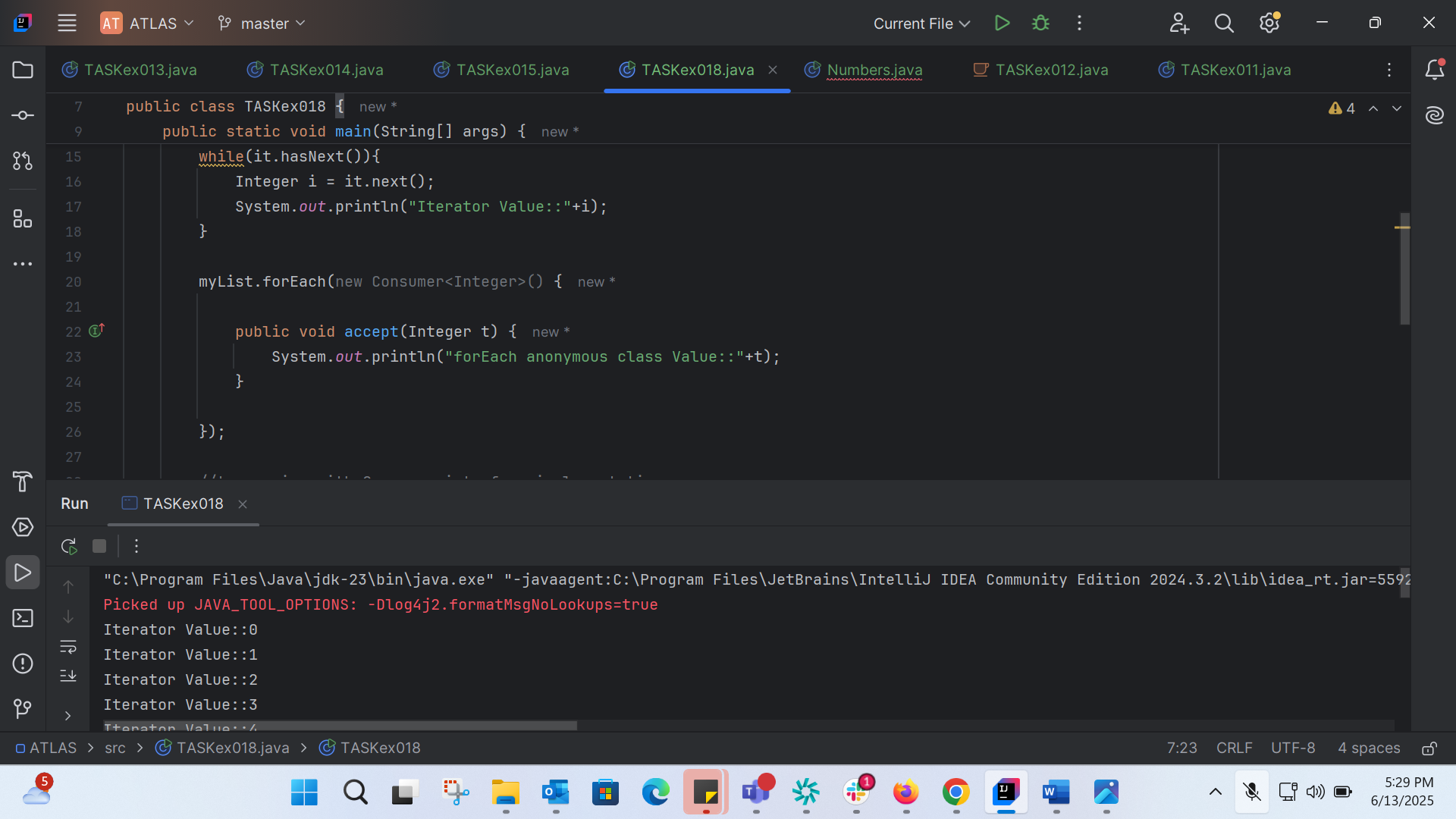
// Abstract Parent Class  
abstract class Personex16 {  
 private String name;  
 private int age;  
  
 // Constructor  
 public Personex16(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 // Getters and Setters  
 public String getName() { return name; }  
 public int getAge() { return age; }  
 public void setName(String name) { this.name = name; }  
 public void setAge(int age) { this.age = age; }  
  
 // Abstract Method  
 public abstract String toString();  
}  
  
// Child class of Person  
class Employee extends Personex16 {  
 private String empId;  
 private String department;  
  
 public Employee(String name, int age, String empId, String department) {  
 super(name, age);  
 this.empId = empId;  
 this.department = department;  
 }  
  
 public String getEmpId() { return empId; }  
 public String getDepartment() { return department; }  
 public void setEmpId(String empId) { this.empId = empId; }  
 public void setDepartment(String department) { this.department = department; }  
  
 public String toString() {  
 return "Name: " + getName() + ", Age: " + getAge() +  
 ", Emp ID: " + empId + ", Department: " + department;  
 }  
}  
  
// Sub-child class of Employee  
class Manager extends Employee {  
 private String team;  
 private int experience;  
  
 public Manager(String name, int age, String empId, String department, String team, int experience) {  
 super(name, age, empId, department);  
 this.team = team;  
 this.experience = experience;  
 }  
  
 public String getTeam() { return team; }  
 public int getExperience() { return experience; }  
 public void setTeam(String team) { this.team = team; }  
 public void setExperience(int experience) { this.experience = experience; }  
  
 public String toString() {  
 return super.toString() +  
 ", Team: " + team + ", Experience: " + experience + " years";  
 }  
}  
  
// Driver class  
public class TASKex016 {  
 public static void main(String[] args) {  
 Manager manager = new Manager("Ravi", 35, "E123", "IT", "Backend Team", 10);  
 System.*out*.println(manager.toString());  
 }  
}



TASK 17 :

Feature of JAVA 8:

1. Lambda Expressions
2. Functional Interfaces
3. Stream API
4. New Date and Time API
5. Optional Class
6. Default Methords in Interfaces
7. Method References
8. Parallel Streams
9. Nashorn JavaScript Engine

TASK 18: 

import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
import java.util.function.Consumer;  
import java.lang.Integer;  
  
public class TASKex018 {  
  
 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);  
 }  
}

Iterator Value::0

Iterator Value::1

Iterator Value::2

Iterator Value::3

Iterator Value::4

Iterator Value::5

Iterator Value::6

Iterator Value::7

Iterator Value::8

Iterator Value::9

forEach anonymous class Value::0

forEach anonymous class Value::1

forEach anonymous class Value::2

forEach anonymous class Value::3

forEach anonymous class Value::4

forEach anonymous class Value::5

forEach anonymous class Value::6

forEach anonymous class Value::7

forEach anonymous class Value::8

forEach anonymous class Value::9

Consumer impl Value::0

Consumer impl Value::1

Consumer impl Value::2

Consumer impl Value::3

Consumer impl Value::4

Consumer impl Value::5

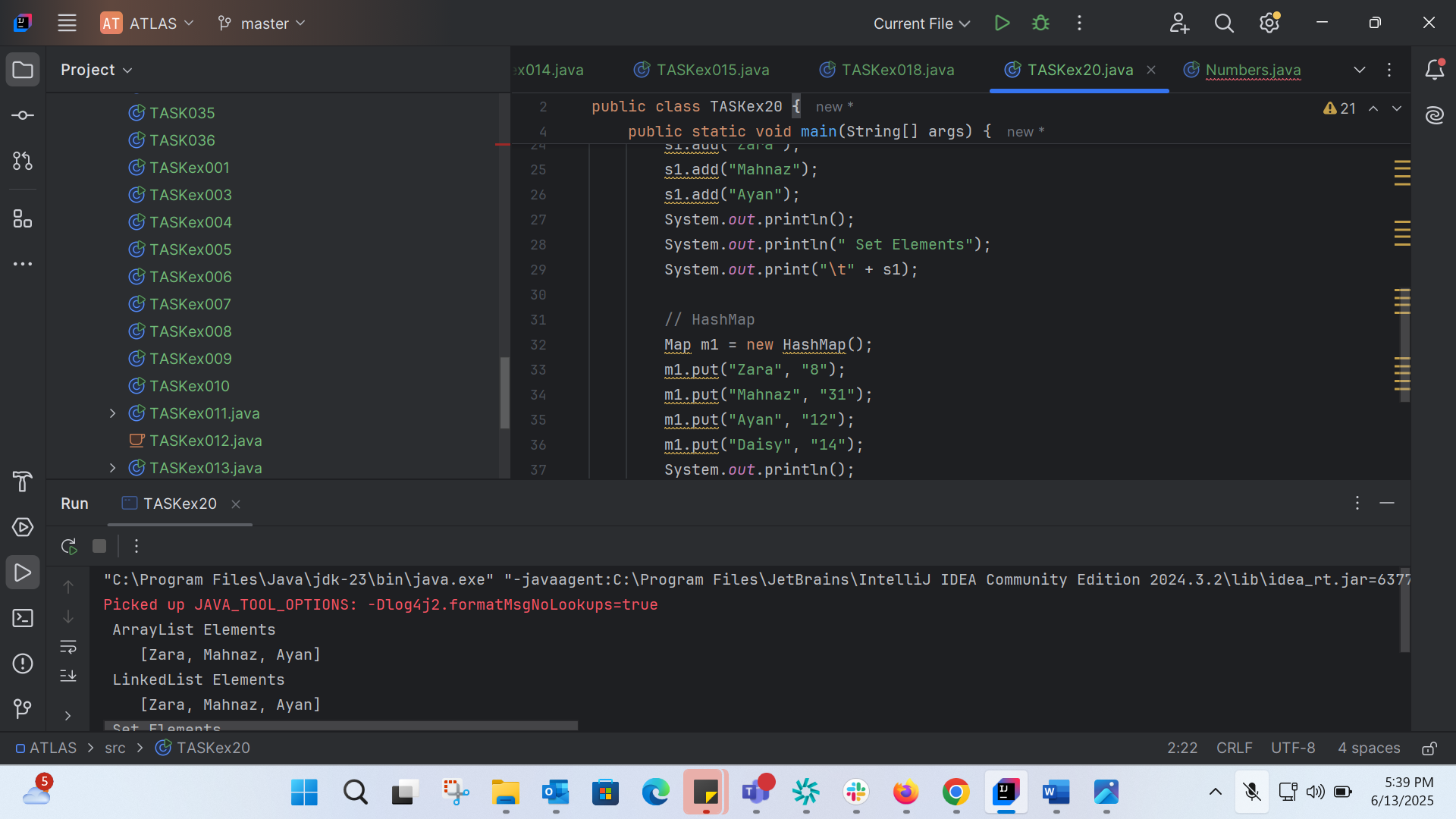
Consumer impl Value::6

Consumer impl Value::7

Consumer impl Value::8

Consumer impl Value::9

TASK 19:

TASK 20: 

import java.util.\*;  
public class TASKex20 {  
  
 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);  
 }  
}

ArrayList Elements

[Zara, Mahnaz, Ayan]

LinkedList Elements

[Zara, Mahnaz, Ayan]

Set Elements

[Ayan, Zara, Mahnaz]

Map Elements

{Daisy=14, Ayan=12, Zara=8, Mahnaz=31}