Day 9 - 11th June 2025

Name: Aravind Kasanagottu

ID:mvsnarav

Task 1:

What do you understand by exceptions?

Exceptions in Java are unexpected events or errors that occur during program execution, disrupting the normal flow.  
They help handle runtime issues like dividing by zero or accessing a missing file using try-catch blocks.

Task 2:

What are the categories of Exceptions do we have in Java? What are they?

Checked Exceptions – These are checked at compile time

Unchecked Exceptions – These occur at runtime and are not checked at compile time

Task 003:

Can you try the below code snippet and let me know which kind of exception is this ?

What is the output of the code..?

// Java program to demonstrates handling

// the exception using try-catch block

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!");

        }

catch (ArithmeticException e) {

            // Handling the exception

            System.out.println(

                "Error: Division by zero is not allowed!");

        }

        finally {

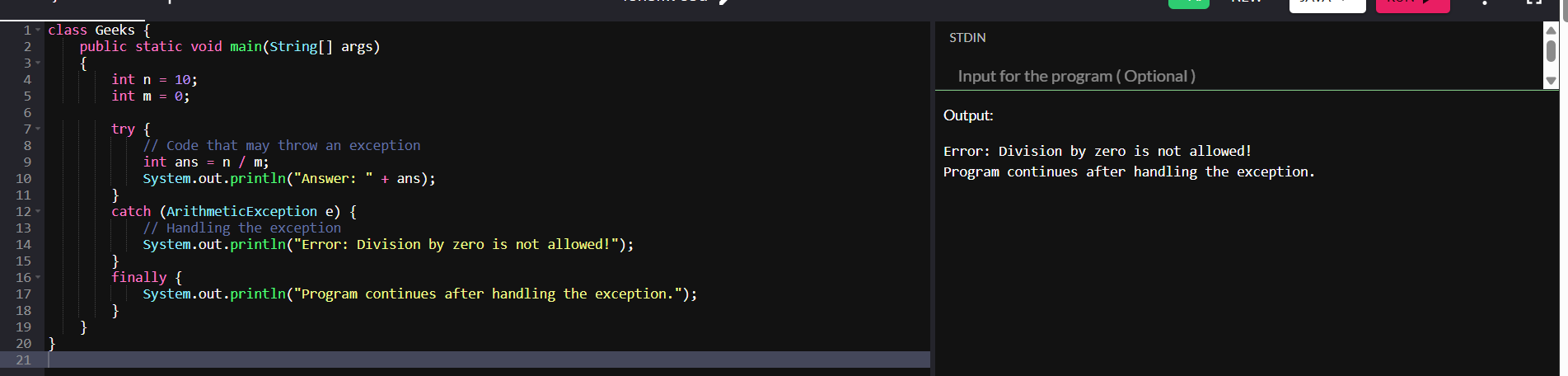
            System.out.println(

                "Program continues after handling the exception.");

        }

    }

}



**ArithmeticException** — because dividing an integer by zero

Occurs when you divide an integer by zero (int ans = n / m;).

Task 4:

List of checked and unchecked exceptions.

**Checked Exceptions (Handled at compile-time):**

* IOException
* FileNotFoundException
* SQLException
* ClassNotFoundException
* InterruptedException
* ParseException

**Unchecked Exceptions (Runtime exceptions):**

* ArithmeticException
* NullPointerException
* ArrayIndexOutOfBoundsException
* NumberFormatException
* IllegalArgumentException
* StringIndexOutOfBoundsException

Task 5:

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

public class ExcepTest {

   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) {

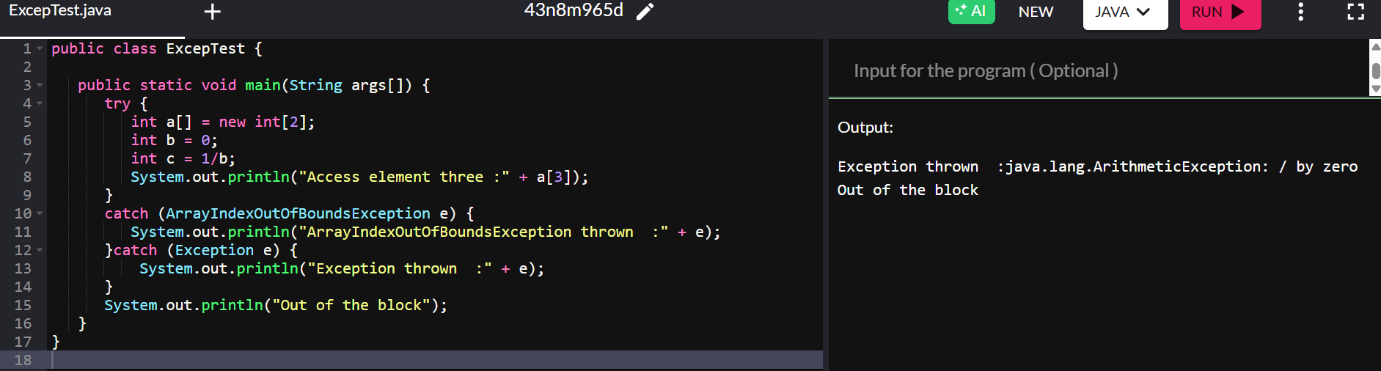
          System.out.println("Exception thrown  :" + e);

      }

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

   }

}



**ArithmeticException** — because dividing an integer by zero

ArrayIndexOutOfBoundsException would have occurred **if** the program reached a[3], which it **never did** because the division by zero happened first.

Task 6:

What is the output of the below code… give your  reason for the output

public class ExcepTest {

   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) {

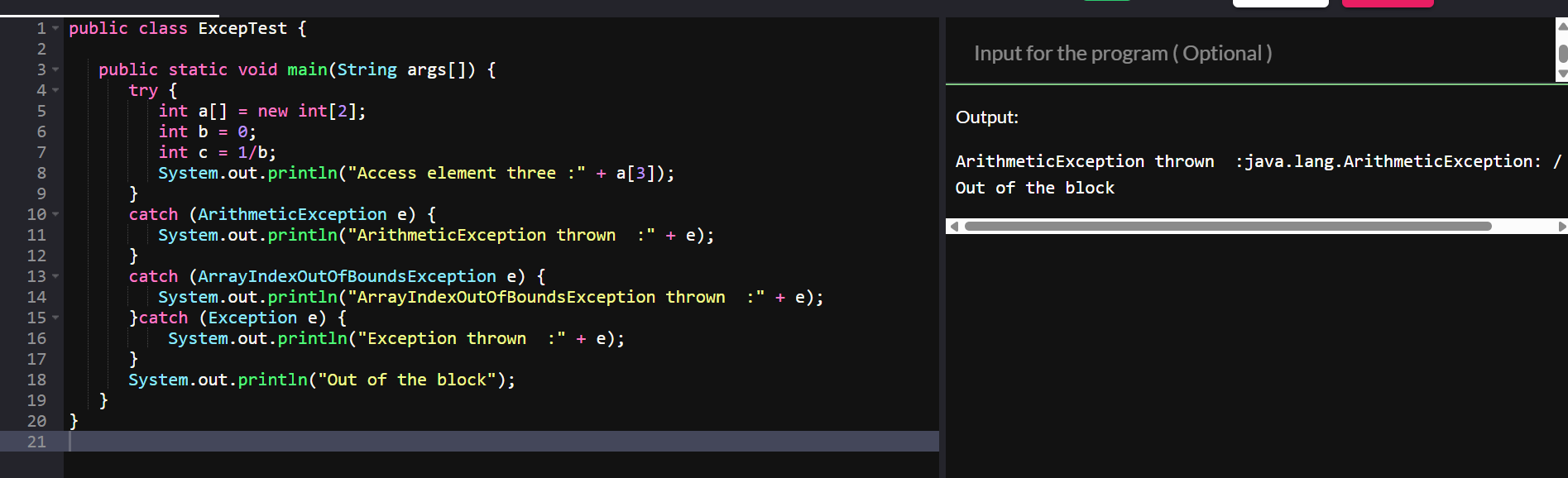
          System.out.println("Exception thrown  :" + e);

      }

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

   }

}



int c = 1 / b throws **ArithmeticException**

Since this is a **specific catch block for** ArithmeticException, it is caught there.

The program **skips the remaining code inside** try, and control jumps directly to the corresponding catch block.

After executing the catch block, the System.out.println("Out of the block"); in the main method runs.

Task 7:

In the below code we are having use multiple catch in a single statement: find the output and try to understand the code..

public class ExcepTest {

   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 | ArithmeticException e) {

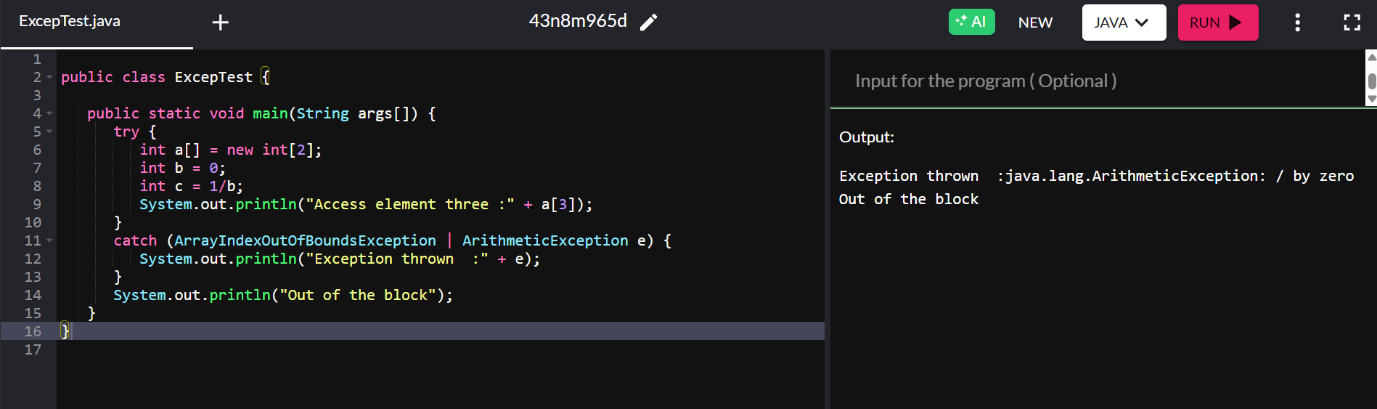
         System.out.println("Exception thrown  :" + e);

      }

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

   }

}



throws an **ArithmeticException**. Because division 1 / 0 happens After that, execution continues and the next line runs

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

Task 8:

Nested try blocks

public class ExcepTest {

   public static void main(String args[]) {

      try {

         int a[] = new int[2];

         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) {

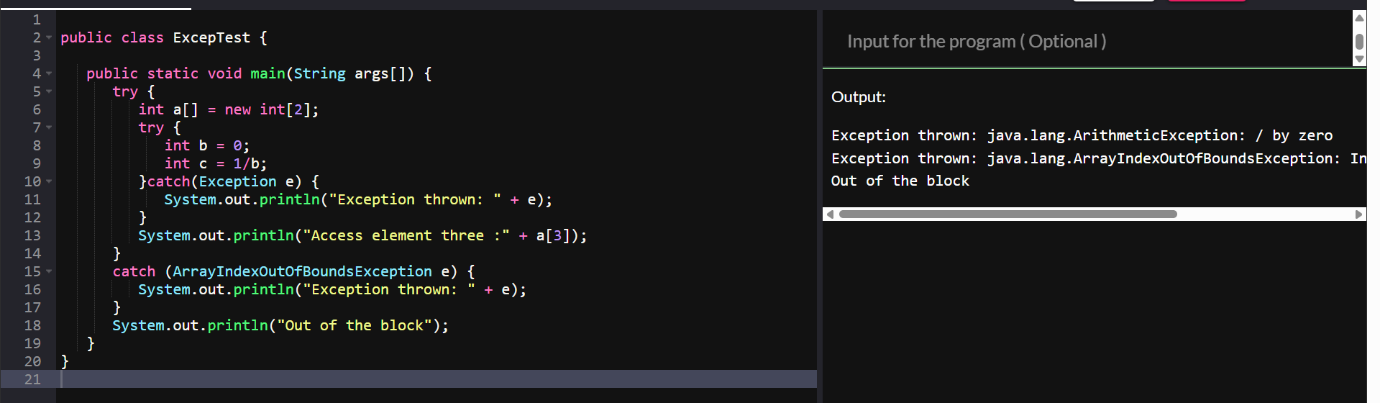
         System.out.println("Exception thrown: " + e);

      }

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

   }

}

****

java.lang.ArithmeticException: / by zero

a[3] **access causes** ArrayIndexOutOfBoundsException

The array a has only 2 elements (a[0], a[1])

Throw and Throws:

Task 009

// Demonstrating how to throw an exception

class MyClass {

    static void fun() throws IllegalAccessException

    {

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

        throw new IllegalAccessException("demo");

    }

    public static void main(String args[])

    {

        try {

            fun();

//method2();   → arrayindex…

//Method3()  —> file not found….

        }

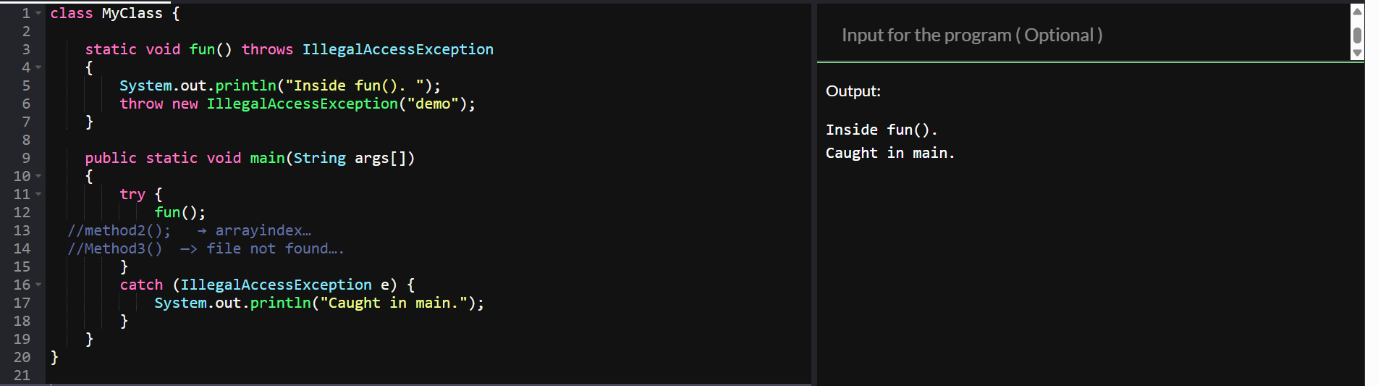
        catch (IllegalAccessException e) {

            System.out.println("Caught in main.");

        }

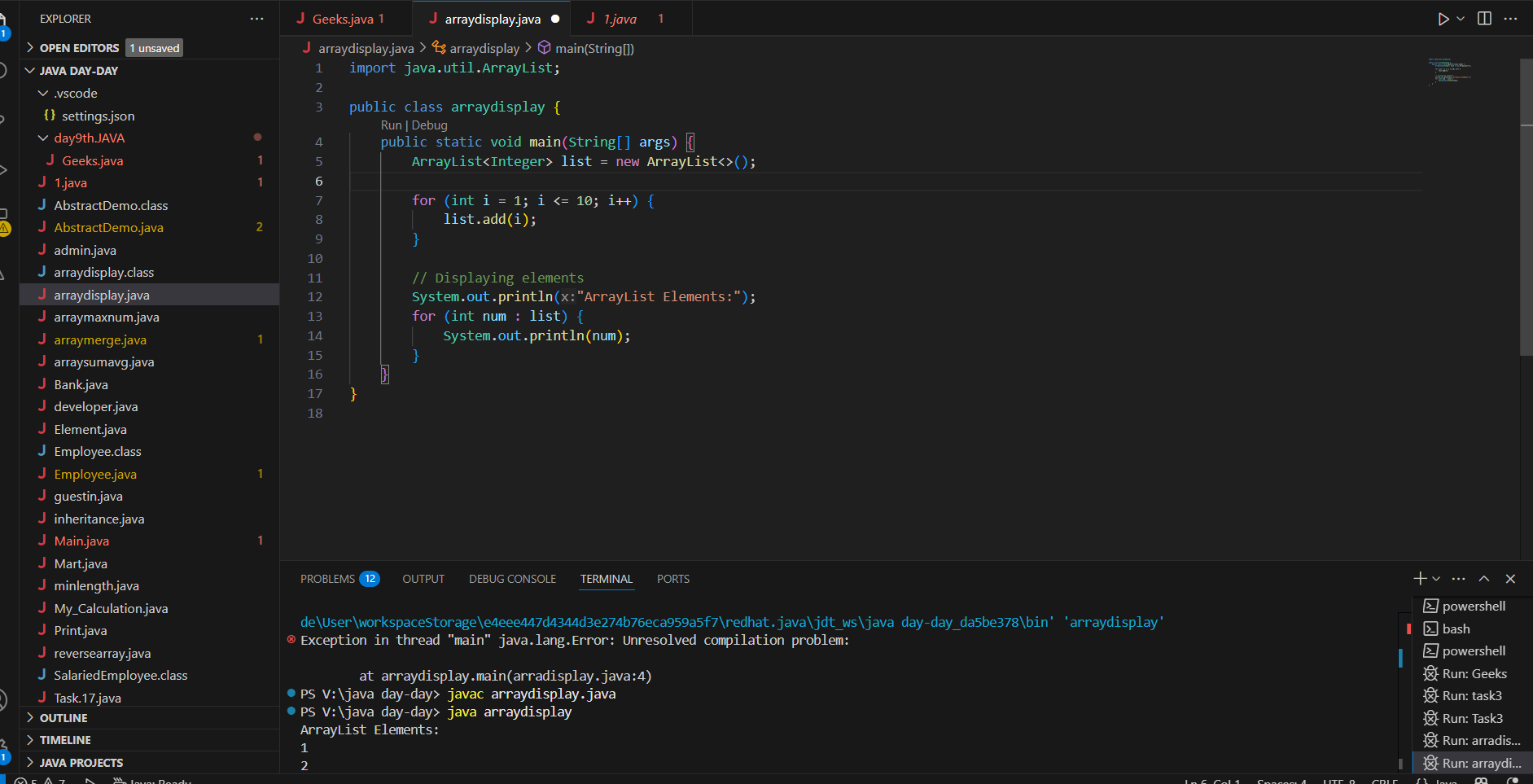
    }

}



Task 011

Wap to create an array list to display 10 elements using for loop.



Task 012

Find the output of the be code snippet..

// Addition, Deletion and Updation of Element

import java.util.\*;

class Main {

    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");

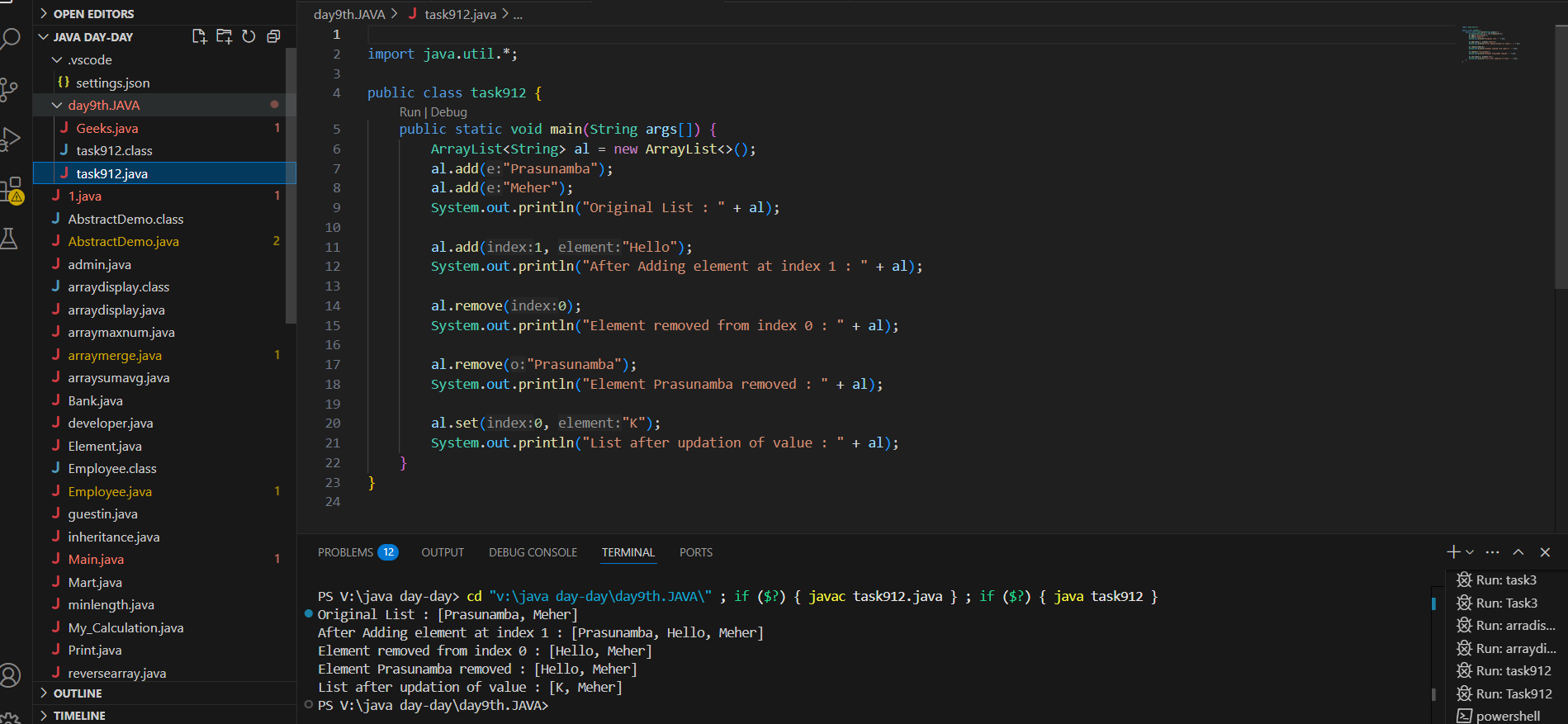
       System.out.println("Element Prasunamba removed : "+ al);

        al.set(0, "K");

        System.out.println("List after updation of value : "+al);

    }

}



Task 013

Run the code and see how the user defined exception works..

User defined Exception:

// A Class that represents user-defined exception

class Customer extends Exception {// predefined class Exception

    public Customer(String m) { // constructor with parameters

        super(m); // parent class constructor

    }

}

// A Class that uses the above Customer

public class setText {

    public static void main(String args[]) {

        try {

            // Throw an object of user-defined exception

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

        }

        catch (Customer  ex) {

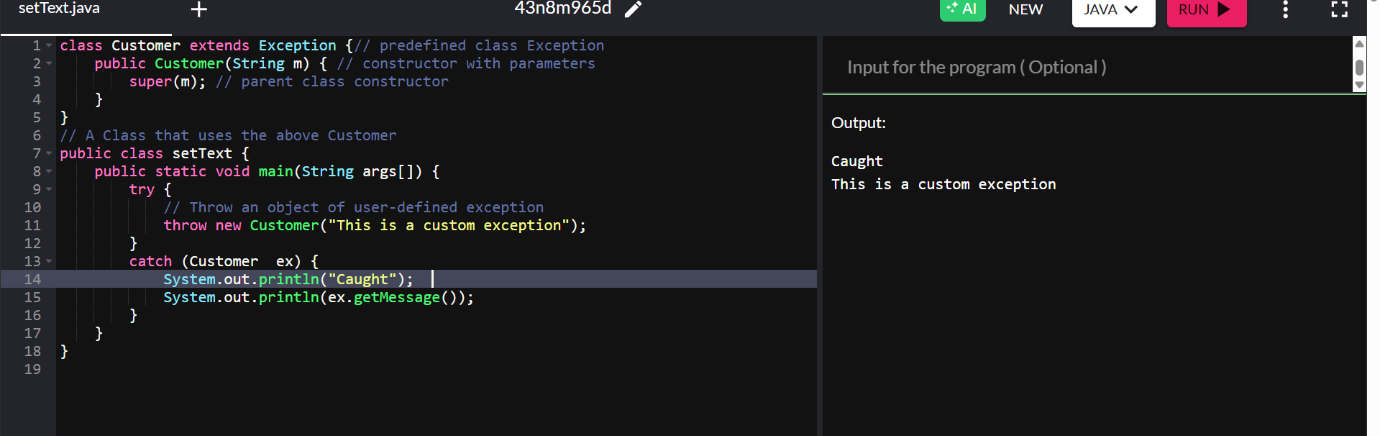
            System.out.println("Caught");

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

        }

    }

}



============================================

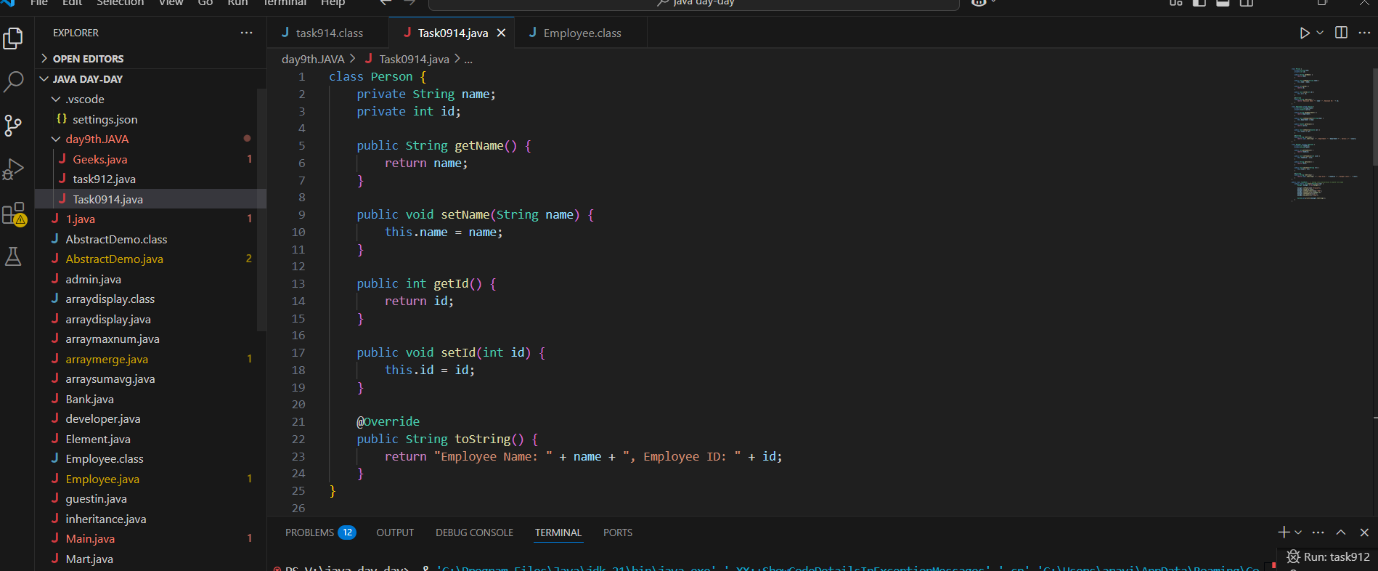
Task 014

Inheritance, Abstraction concepts..

Classes customer/ person , employee, Manager  … 2 variables in each class

Driver class – display all the variables… toString();

Hint : use getter and setters..



Task 015

What is the output of the below code snippet..  Explain ..

class OuterClass {

  int x = 10;

  class InnerClass {

    int y = 5;

  }

}

public class Main {

  public static void main(String[] args) {

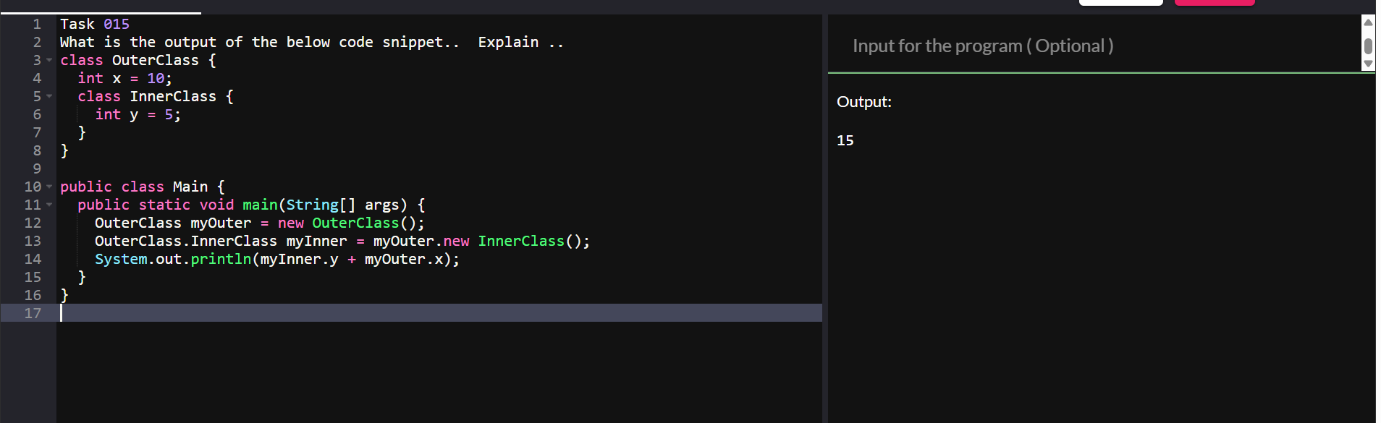
    OuterClass myOuter = new OuterClass();

    OuterClass.InnerClass myInner = myOuter.new InnerClass();

    System.out.println(myInner.y + myOuter.x);

  }

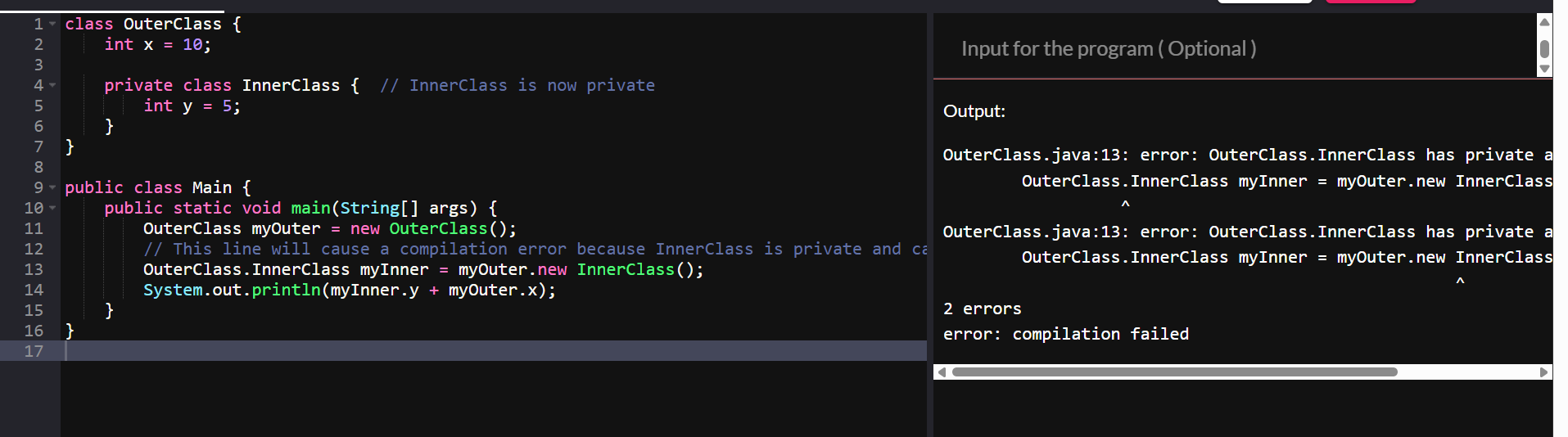
}



Task 016

Use the above code and make the inner class as private and see the output..

Ex: private  class InnerClass {



Task 017

Use the above code Task 015 and make the inner class static … see the output and explain..

Ex: static class InnerClass {

