9th Day of Study – 11/06/2025

**Task 1:**

**What do you understand by exceptions?**

It occurs during the execution of a program that disrupts the normal flow of instructions. When an error occurs that the program cannot handle in its current state, we will get an exception.

**Task 2:**

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

In Java, there are two kinds of exceptions:

* Unchecked exceptions (runtime exceptions)
* Checked exceptions (compile-time exceptions)

**Task 3:**

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

// 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!");

}

finally {

System.out.println(

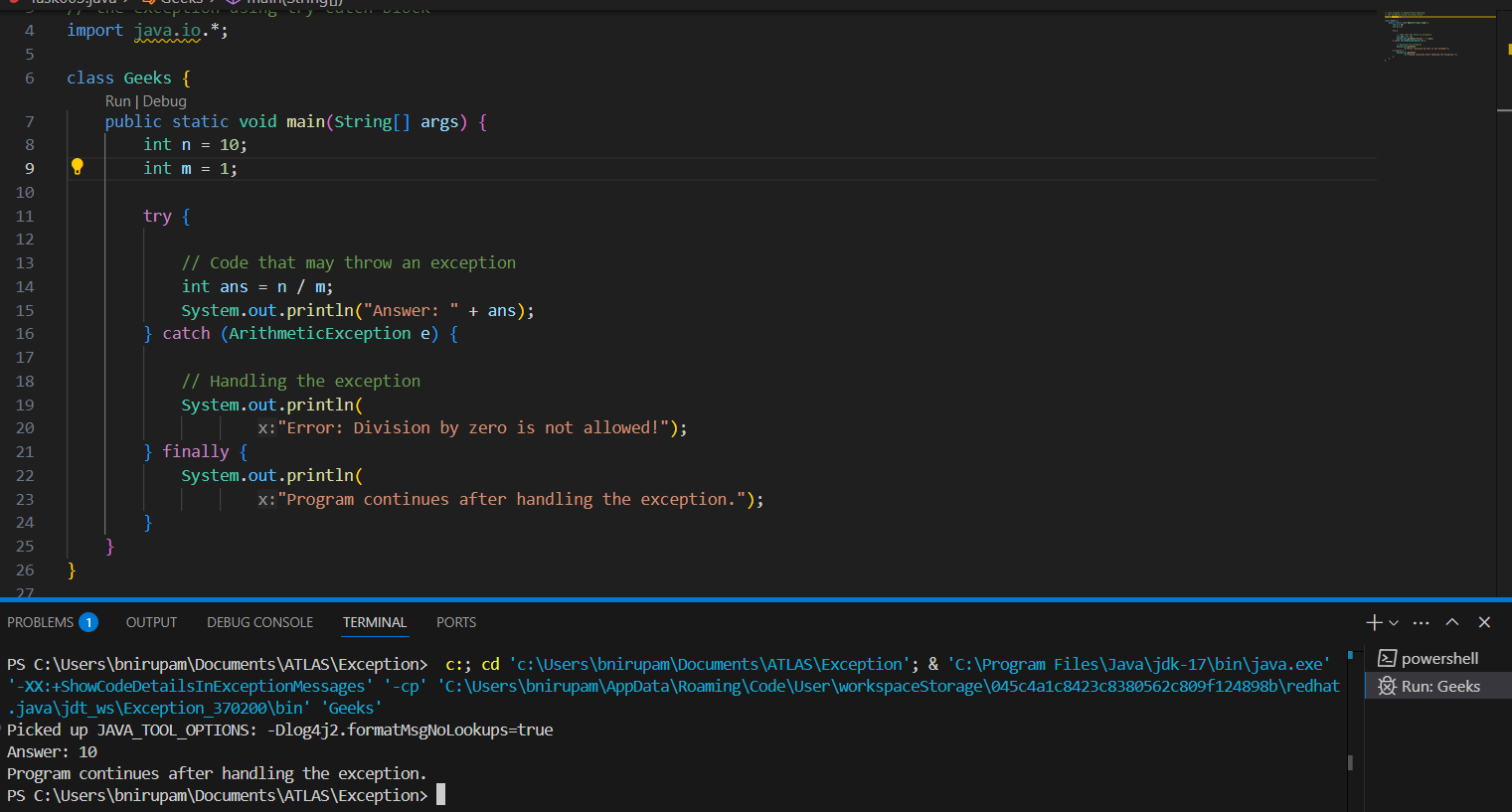
"Program continues after handling the exception.");

}

}

}

The specific exception type in this Java program is Arithmetic Exception. This exception occurs because the code attempts to perform division by zero (n / m where m is 0), which is an illegal arithmetic operation. Arithmetic Exception is an unchecked exception in Java, meaning the compiler doesn't force you to handle it, but it's good practice to do so when such a condition is anticipated.



**Task 4:**

**List of checked and unchecked exceptions.**

**Checked Exceptions**

*(The compiler forces you to handle these, typically for external, recoverable issues)*

* **IOException**: Problems during input/output operations (e.g., file not found, network error).
* **SQLException**: Errors encountered while interacting with a database.
* **ClassNotFoundException**: A class specified by name could not be found at runtime.
* **InterruptedException**: A thread that's sleeping, waiting, or otherwise blocked was interrupted.
* **CloneNotSupportedException**: An object cannot be cloned because its class doesn't implement Cloneable.

**Unchecked Exceptions**

*(The compiler doesn't force handling; these often indicate programming bugs)*

* **NullPointerException**: Attempting to use an object reference that currently points to null.
* **ArithmeticException**: An illegal mathematical operation occurred, like division by zero.
* **ArrayIndexOutOfBoundsException**: Accessing an array element using an invalid index (too small or too large).
* **ClassCastException**: Trying to convert an object to a type it's not actually an instance of.
* **IllegalArgumentException**: A method was called with an inappropriate or invalid argument.
* **IllegalStateException**: An object is in an inappropriate state for the requested operation.

**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) {

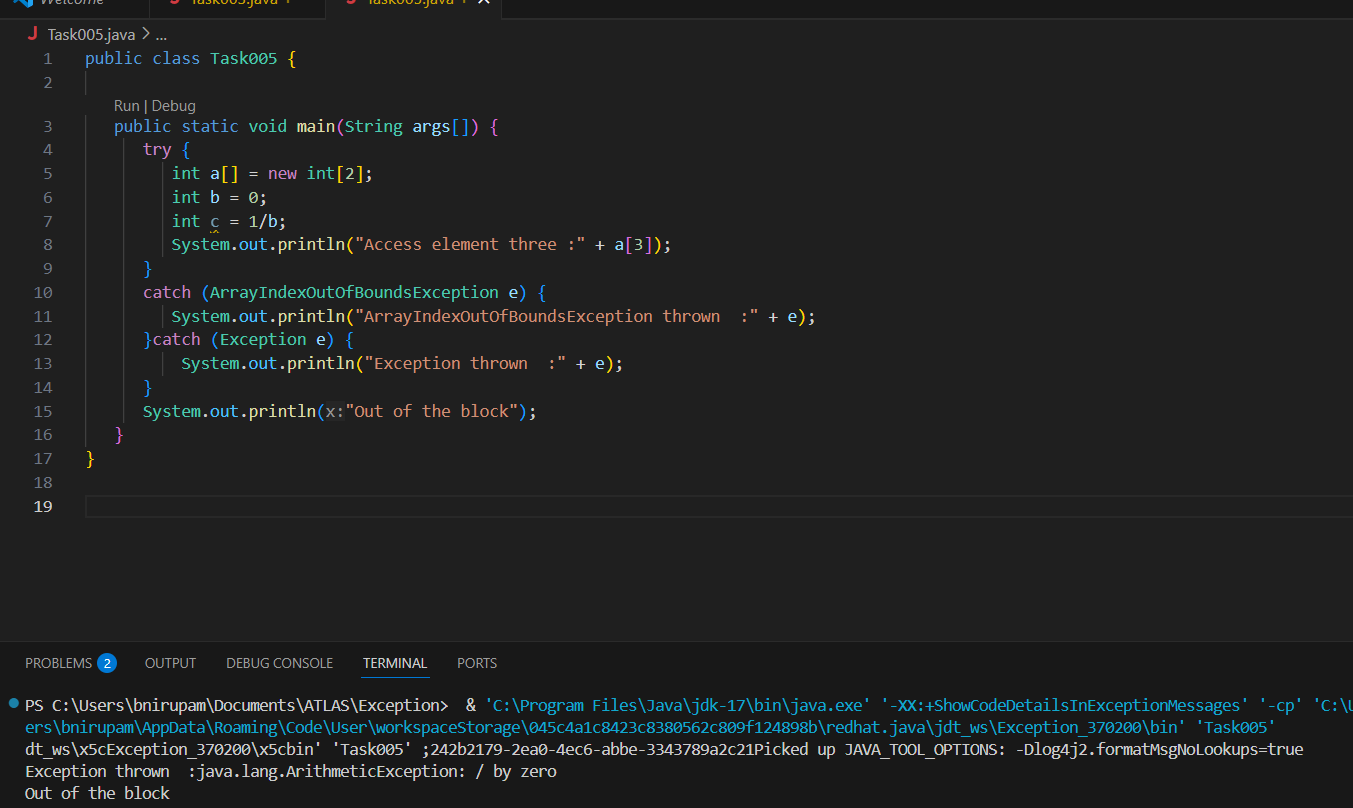
          System.out.println("Exception thrown  :" + e);

      }

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

   }

}



The specific exception type in this Java program is Arithmetic Exception. This exception occurs because the code attempts to perform division by zero (n / m where m is 0), which is an illegal arithmetic operation.

**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) {

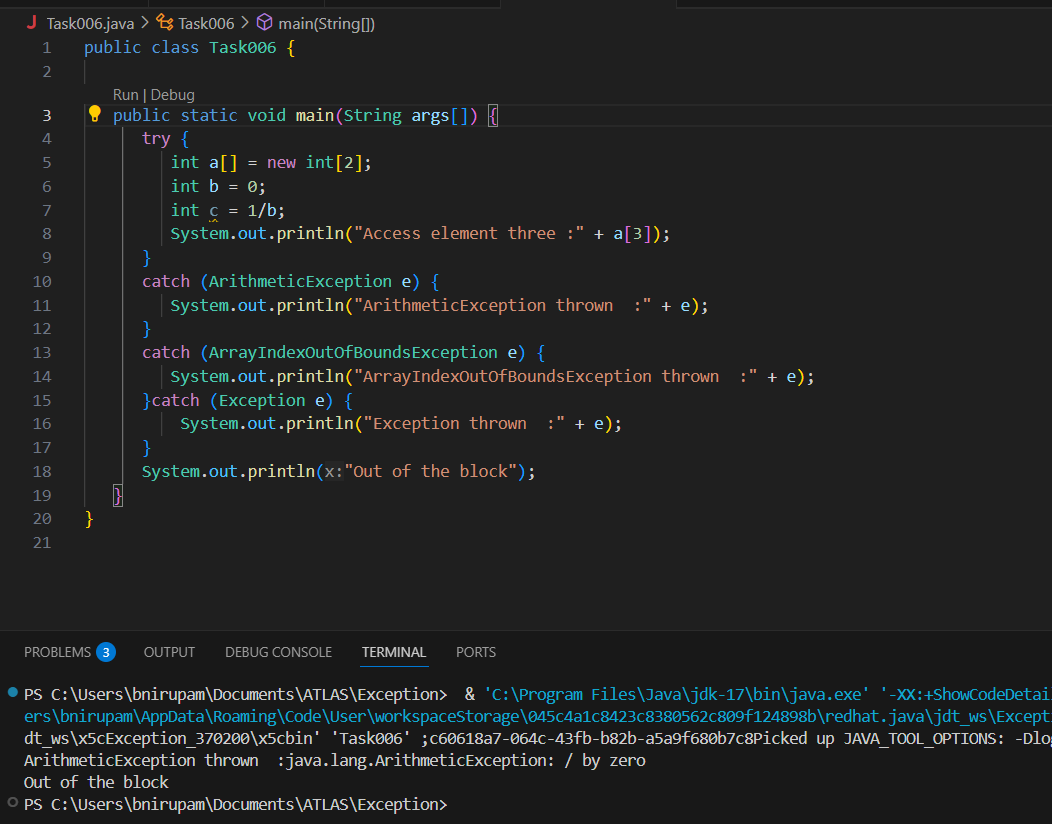
          System.out.println("Exception thrown  :" + e);

      }

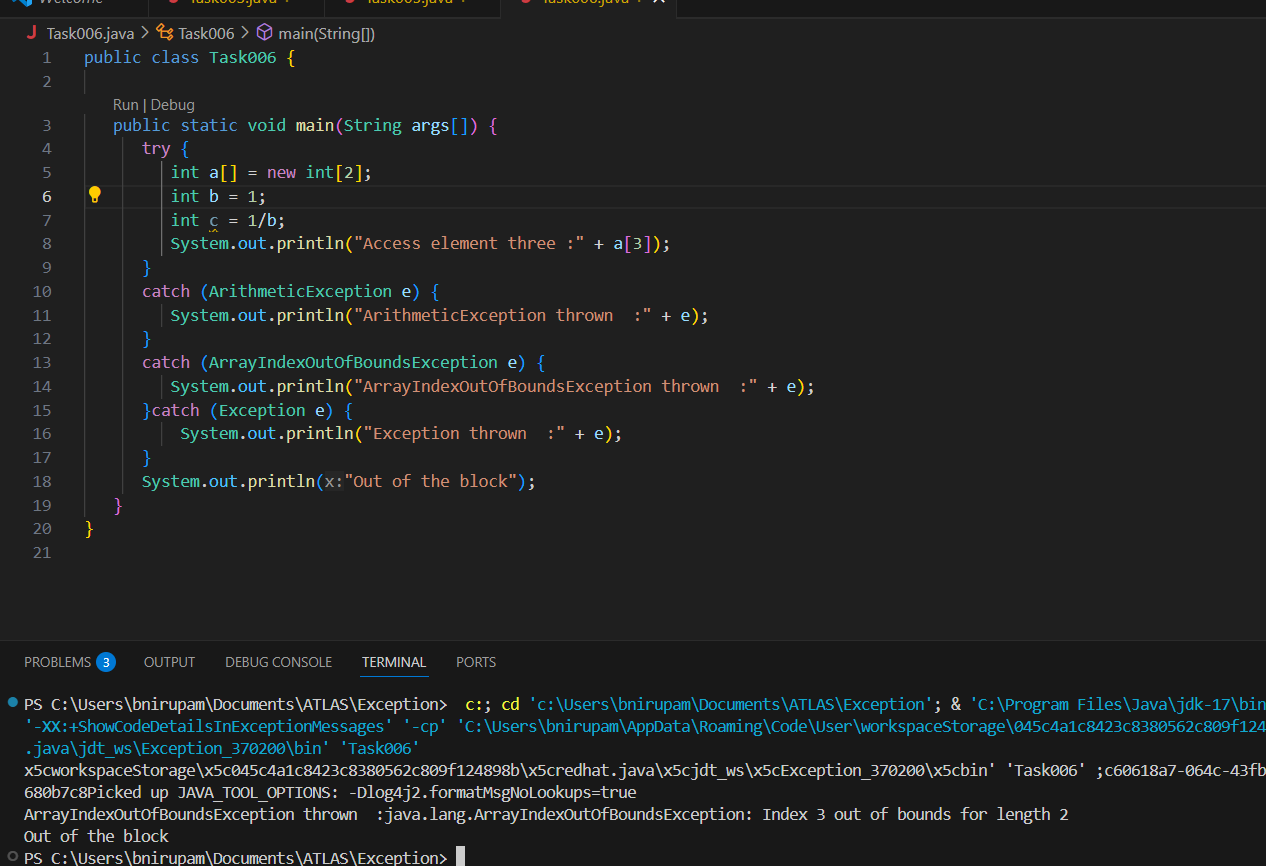
      System.out.println("Out of the block");

   }

}



The specific exception type in this Java program is Arithmetic Exception. This exception occurs because the code attempts to perform division by zero (n / m where m is 0), which is an illegal arithmetic operation.

****

(when replaced 1 with 0) - This is an unchecked exception in Java. It means your program tried to access an element of an array at an index that doesn't exist.

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) {

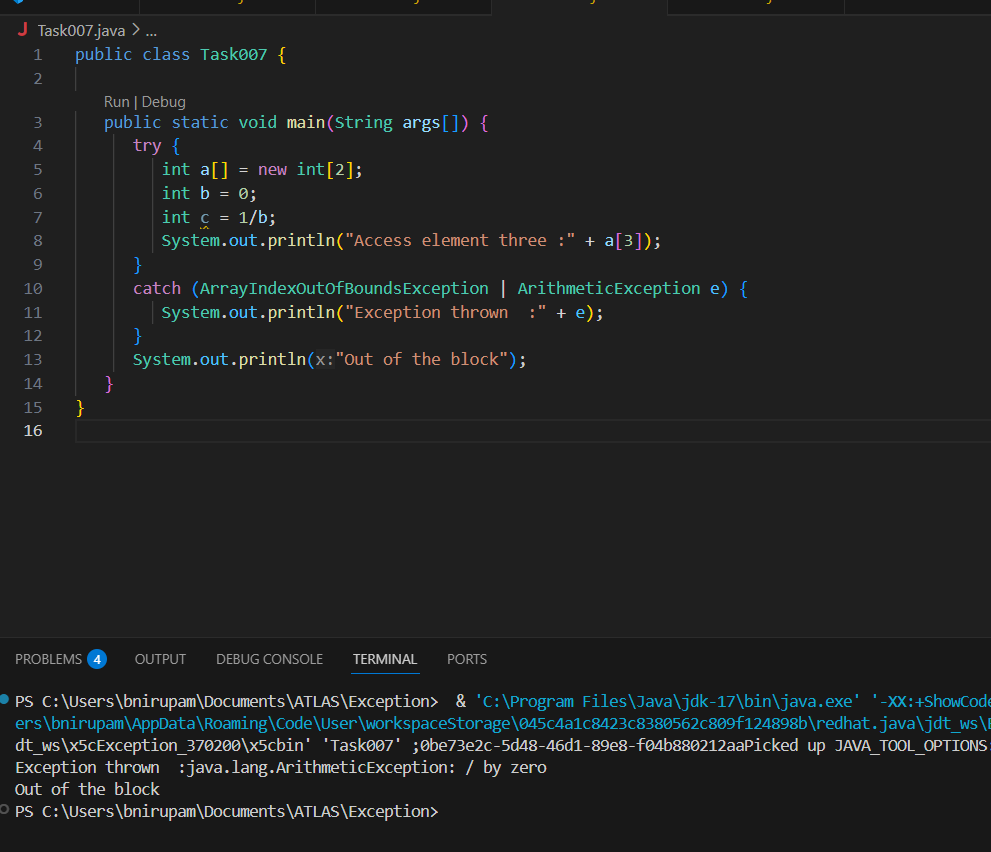
         System.out.println("Exception thrown  :" + e);

      }

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

   }

}



Task 008:

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) {

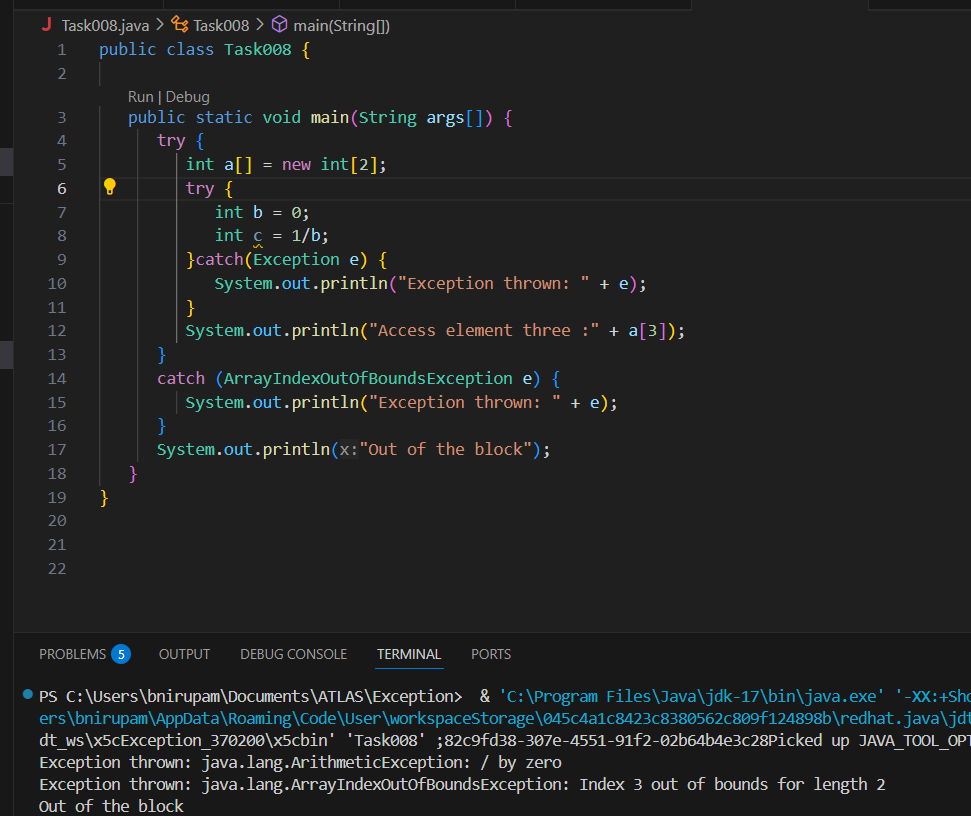
         System.out.println("Exception thrown: " + e);

      }

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

   }

}



**ArithmeticException: / by zero**: This happens first when you try to divide 1 by 0 in your inner try block. The inner catch(Exception e) handles it, prints the error, and allows the program to continue.

**ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 2**: After the first error is handled, execution continues to the line System.out.println("Access element three :" + a[3]);. This array a was created with a length of 2 (valid indices 0 and 1), so trying to access index 3 causes this second error. The outer catch(ArrayIndexOutOfBoundsException e) handles this one.

**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….

        }

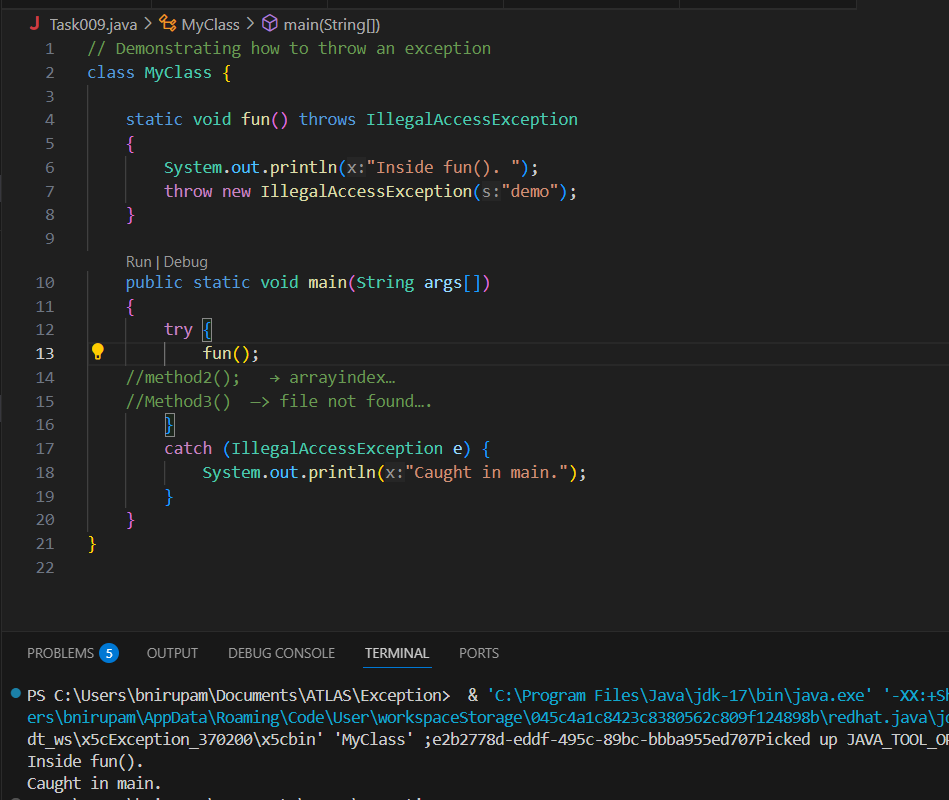
        catch (IllegalAccessException e) {

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

        }

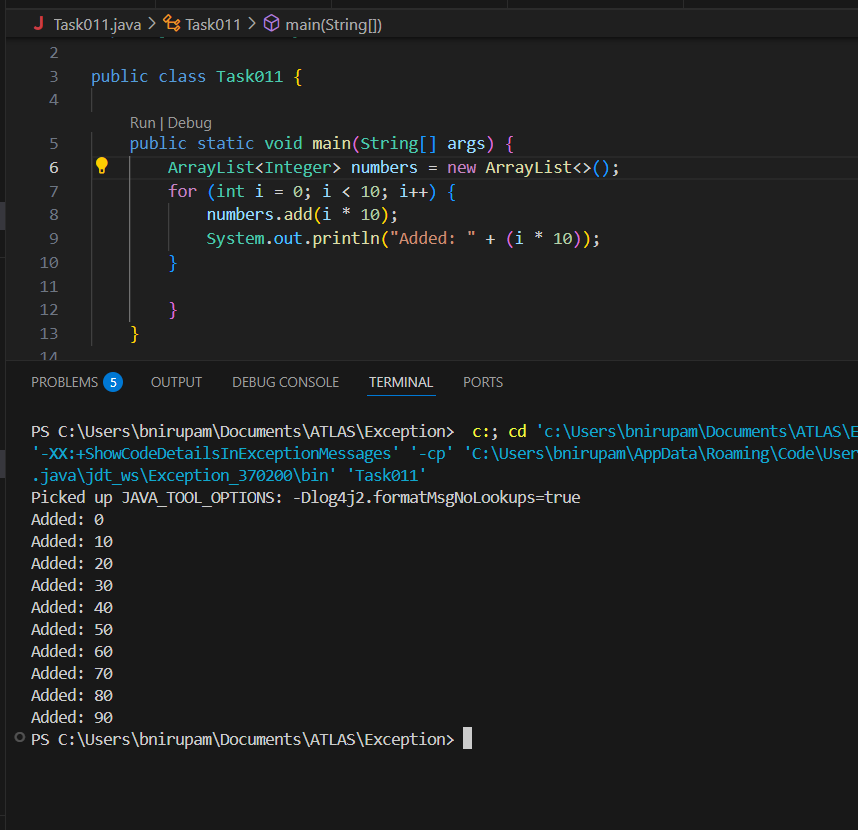
    }

}



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");

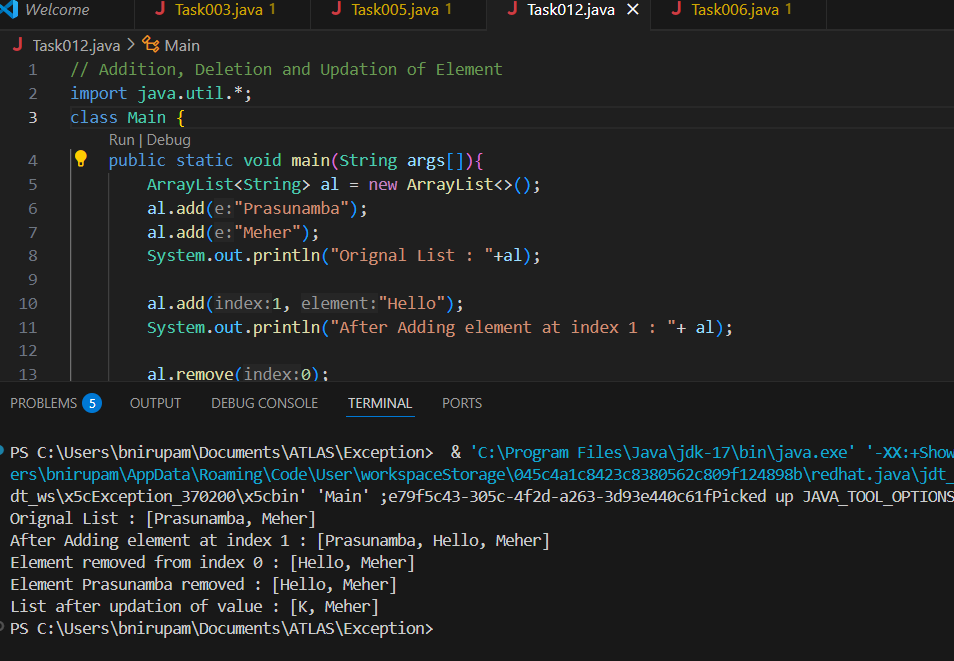
       System.out.println("Element Prasunamba removed : "+ al);

        al.set(0, "K");

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

    }

}



**Task 013**

Run the code and see hope 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) {

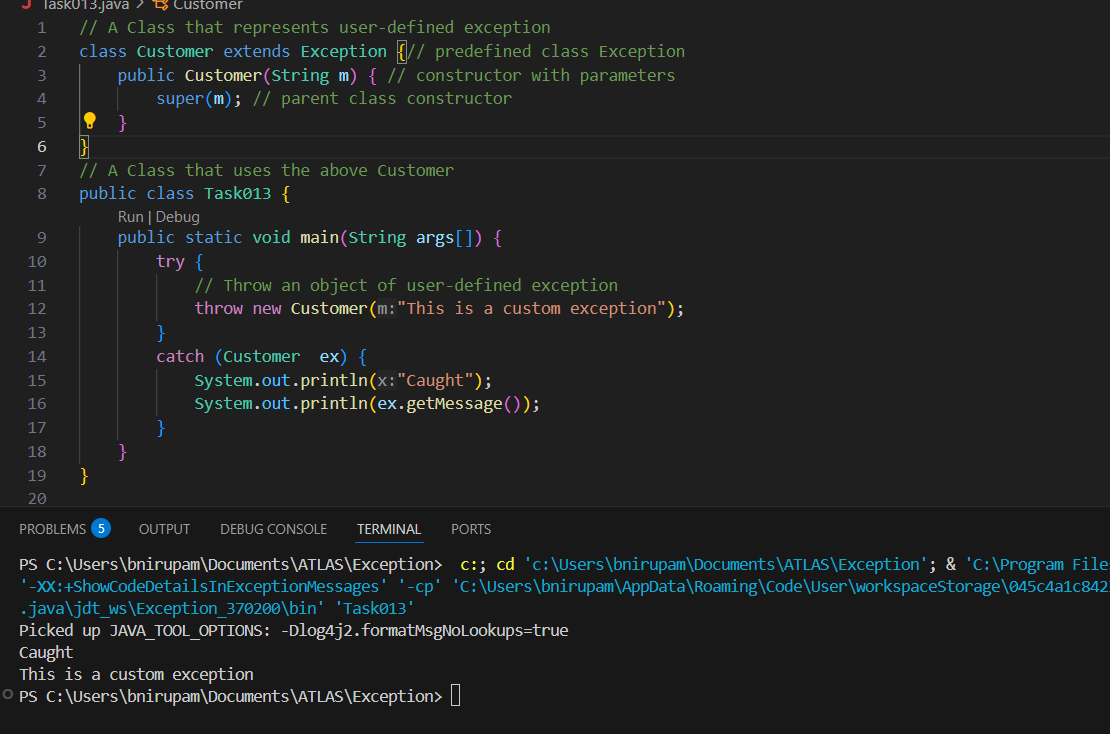
            System.out.println("Caught");

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

        }

    }

}



**Task 014**

Inheritance

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

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

Hint : use getter and setters..

public class Task014 {

    public static void main(String[] args) {

        Customer cust = new Customer("Nirupama", 30, "123");

        Employee emp = new Employee("Joseph", 28, "456");

        Manager manager = new Manager("Abhilash", 35, "789");

        System.out.println("Customer : " + cust);

        System.out.println("Employee : " + emp);

        System.out.println("Manager : " + manager);

    }

}

class Person {

    private String name;

    private int age;

    public Person(String name, int age) {

        this.name = name;

        this.age = age;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getAge() {

        return age;

    }

    public void setAge(int age) {

        this.age = age;

    }

    public String toString() {

        return "Name:" + name + " Age:" + age;

    }

}

class Customer extends Person {

    private String customerId;

    public Customer(String name, int age, String customerId) {

        super(name, age);

        this.customerId = customerId;

    }

    public String getCustomerId() {

        return customerId;

    }

    public void setCustomerId(String customerId) {

        this.customerId = customerId;

    }

    public String toString() {

        return super.toString() + " Customer ID: " + customerId;

    }

}

class Employee extends Person {

    private String employeeId;

    public Employee(String name, int age, String employeeId) {

        super(name, age);

        this.employeeId = employeeId;

    }

    public String getEmployeeId() {

        return employeeId;

    }

    public void setEmployeeId(String employeeId) {

        this.employeeId = employeeId;

    }

    public String toString() {

        return super.toString() + ", Employee ID: " + employeeId;

    }

}

class Manager extends Person {

    private String managerId;

    public Manager(String name, int age, String managerId) {

        super(name, age);

        this.managerId = managerId;

    }

    public String getManagerId() {

        return managerId;

    }

    public void setManagerId(String managerId) {

        this.managerId = managerId;

    }

    public String toString() {

        return super.toString() + ", Manager ID:" + managerId;

    }

}

**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 {

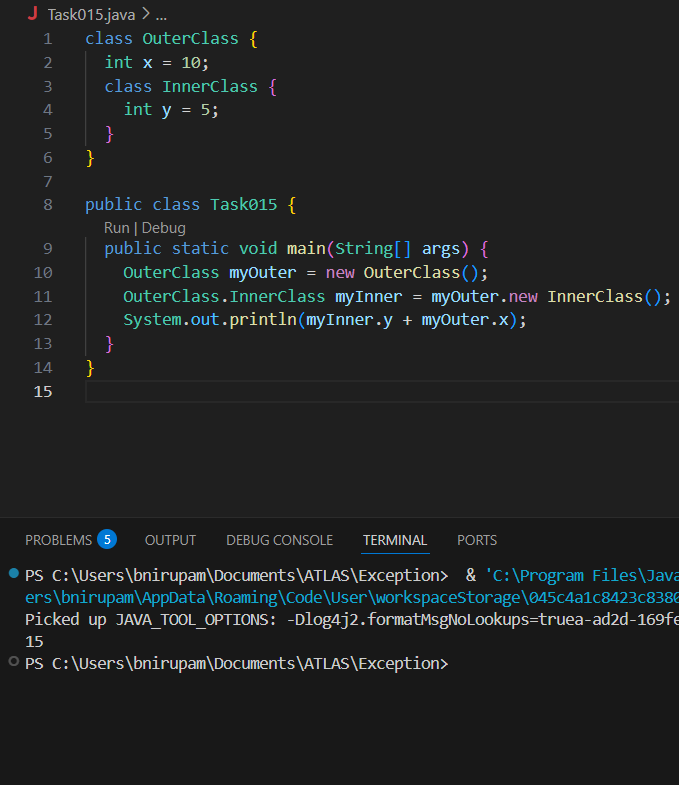
  public static void main(String[] args) {

    OuterClass myOuter = new OuterClass();

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

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

  }

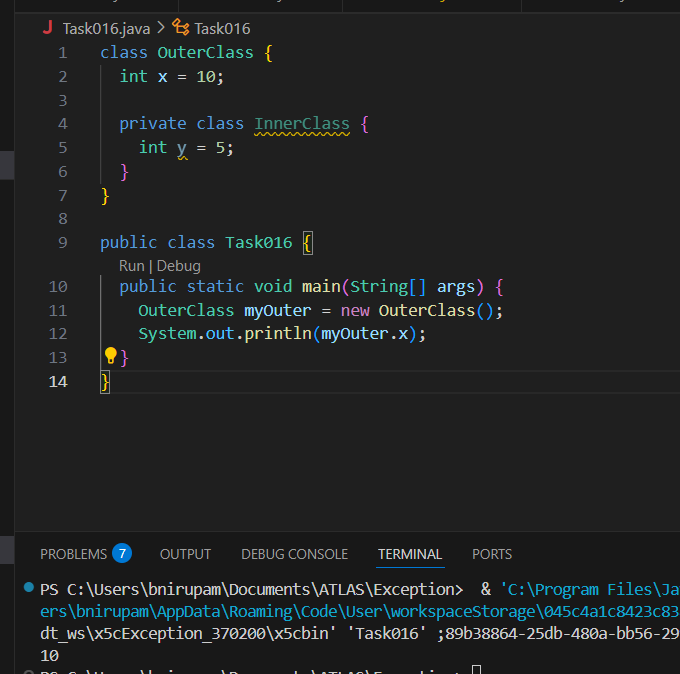
} 

The code creates an OuterClass object (myOuter) and then an InnerClass object (myInner) that is tied to myOuter. It then adds the y variable (value 5) from myInner and the x variable (value 10) from myOuter, resulting in 5 + 10 = 15 being printed.

Task 016

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

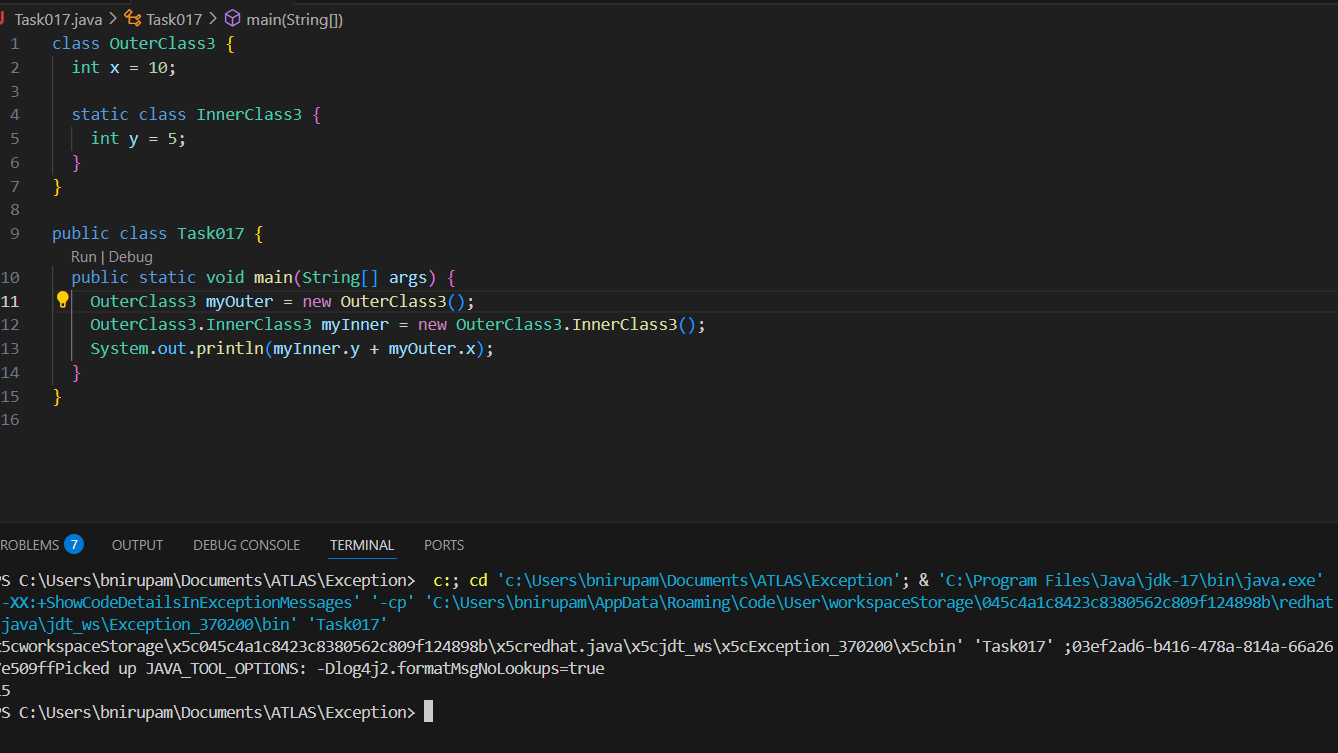
Ex: private  class InnerClass {



Task 017

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

Ex: static class InnerClass {



Task 018

Use the above code Task 015 and create a method in innerclass and return the outer class variable

class OuterClass{

Int x = 50;

Class InnerClass {

Public int innerMethod() {

Return x;

}

}

}

Public class DriverClass {

psvm(){

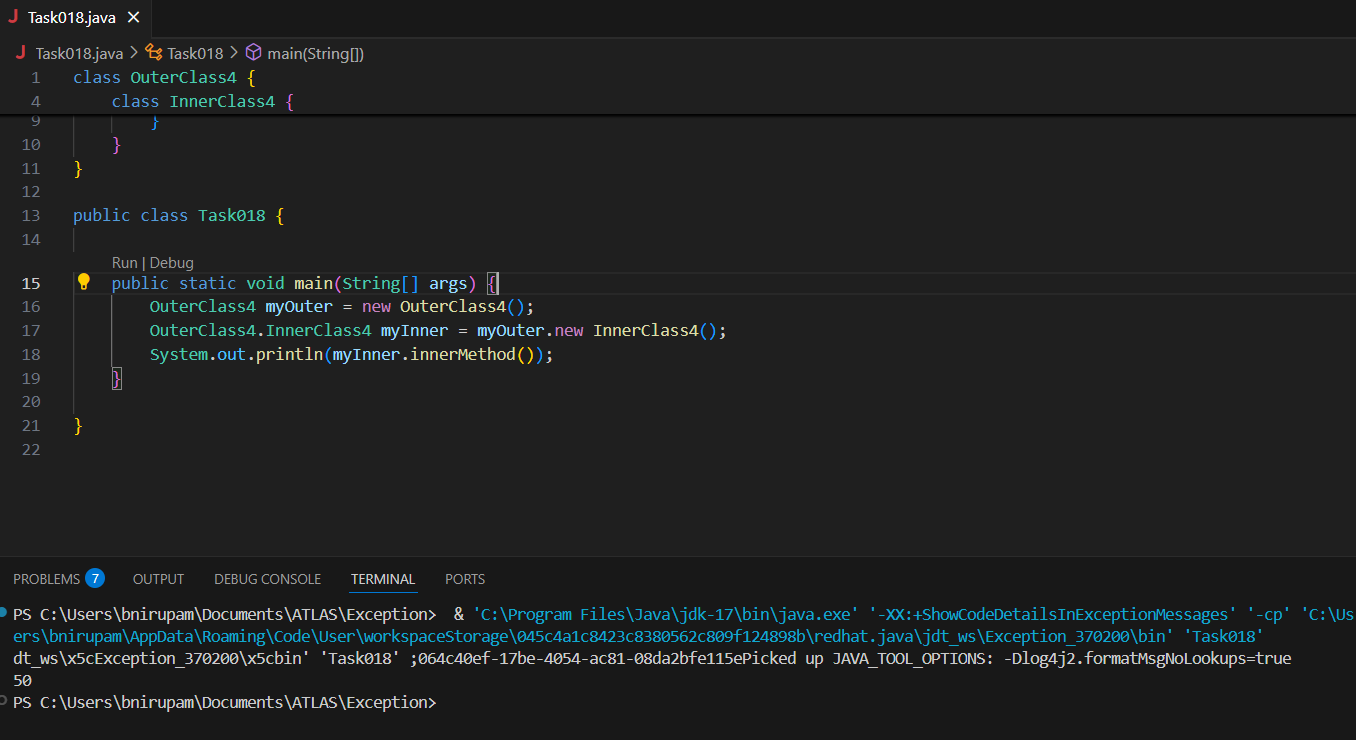
OuterClass myOuter = new OuterClass();

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

     System.out.println(myInner.innerMethod());

}

}



Task 019  —

class OuterClass {

  int x = 10;

  static class InnerClass {

    static int y = 5;

  }

}

public class Main {

  public static void main(String[] args) {

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

    System.out.println(myInner.y);

  }

}

