**ID: bvarshsp**

**Name: Varsha SP**

**Day 9 –11th June 2025**

**Task 1:**

What do you understand by exceptions?

3 min 10.20 to 10.23

Answer: Exceptions are unexpected events that interrupt the normal flow of a program like invalid input, missing files or divide-by-zero errors. They help us identify and fix problems during runtime and handling them prevents the program from crashing.

Task 2:

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

Answer: In Java, exceptions are mainly categorized into 2 types. There is also a broader group called Errors.

1. Checked Exceptions (compile-time exceptions)
2. Unchecked Exceptions (Runtime Exceptions)
3. Errors (Not Exceptions)

Plz refer the above image .. for more details.

Task 3:

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!");

        }

        finally {

            System.out.println(

                "Program continues after handling the exception.");

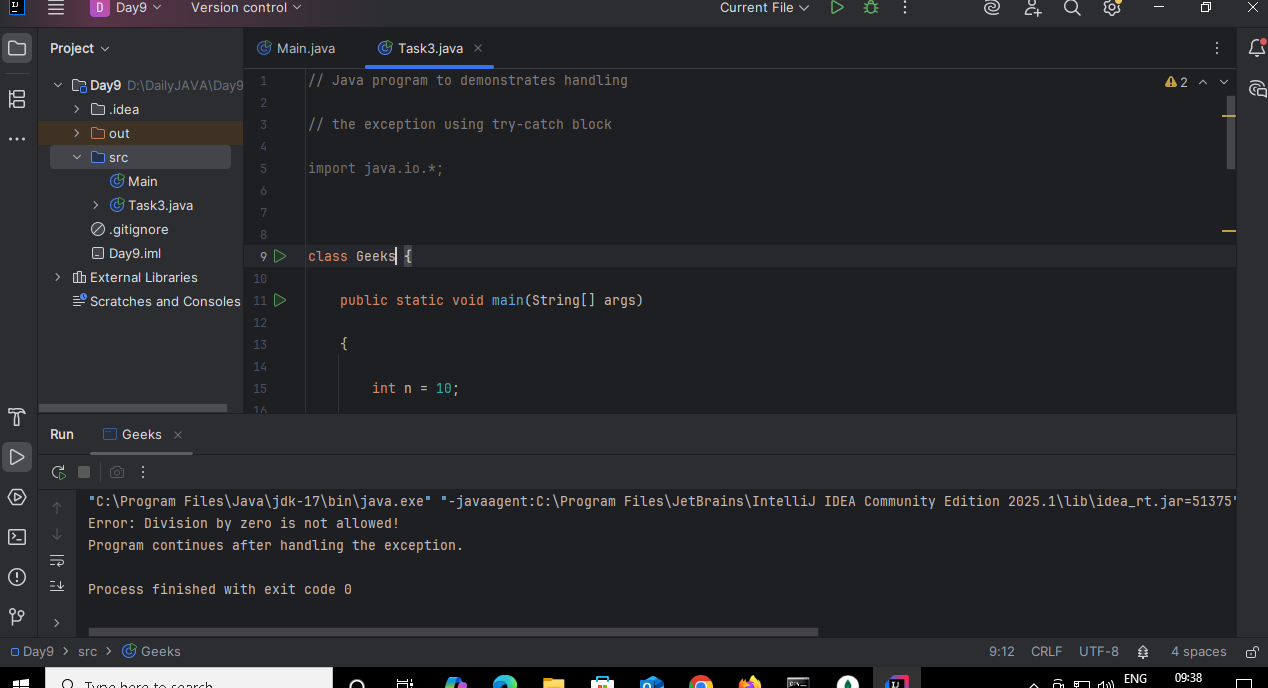
        }

    }

}

Answer:

Error: Division by zero is not allowed!  
Program continues after handling the exception.



Task 4:

List of checked and unchecked exceptions.

Answer:

Checked Exceptions: IOException, FileNotFoundException, SQLException, ClassNotFoundException, InterruptedException, ParseException, MalformedURLException, NoSuchMethodException.

Unchecked Exceptions: ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException, NumberFormatException, IllegalArgumentException, IllegalStateException, ClassCastException.

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);

      }

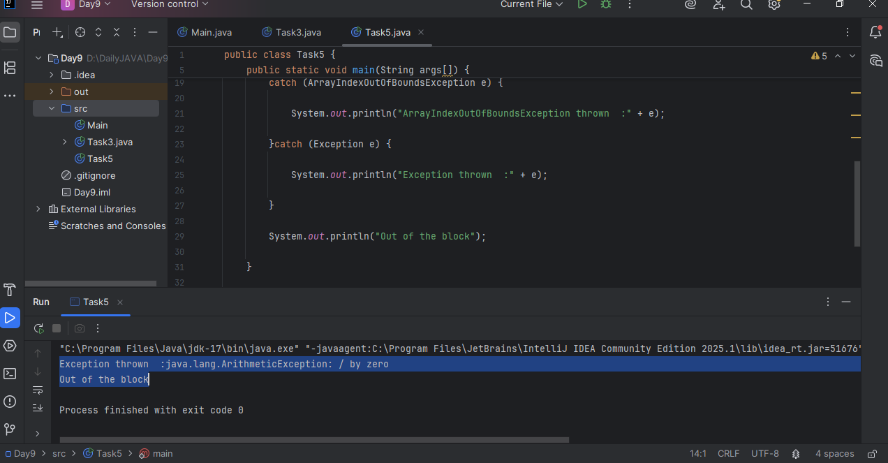
      System.out.println("Out of the block");

   }

}

Answer:

Exception thrown  :java.lang.ArithmeticException: / by zero  
Out of the block



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);

      }

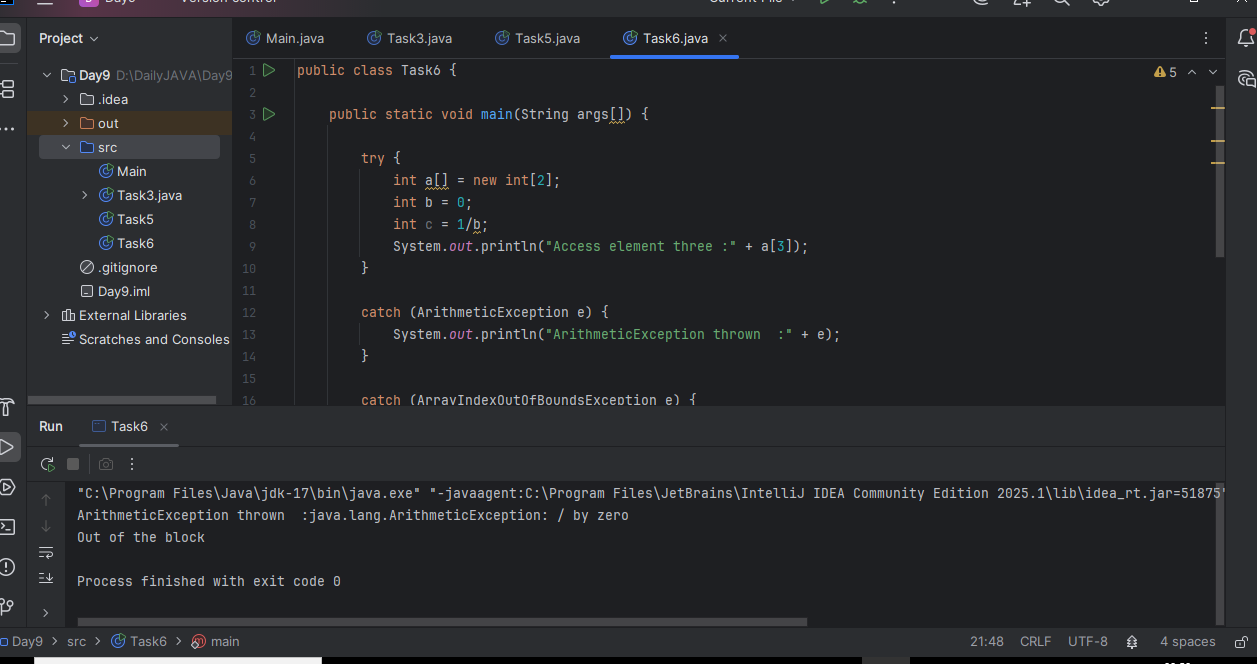
      System.out.println("Out of the block");

   }

}

Answer:

ArithmeticException thrown  :java.lang.ArithmeticException: / by zero  
Out of the block



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);

      }

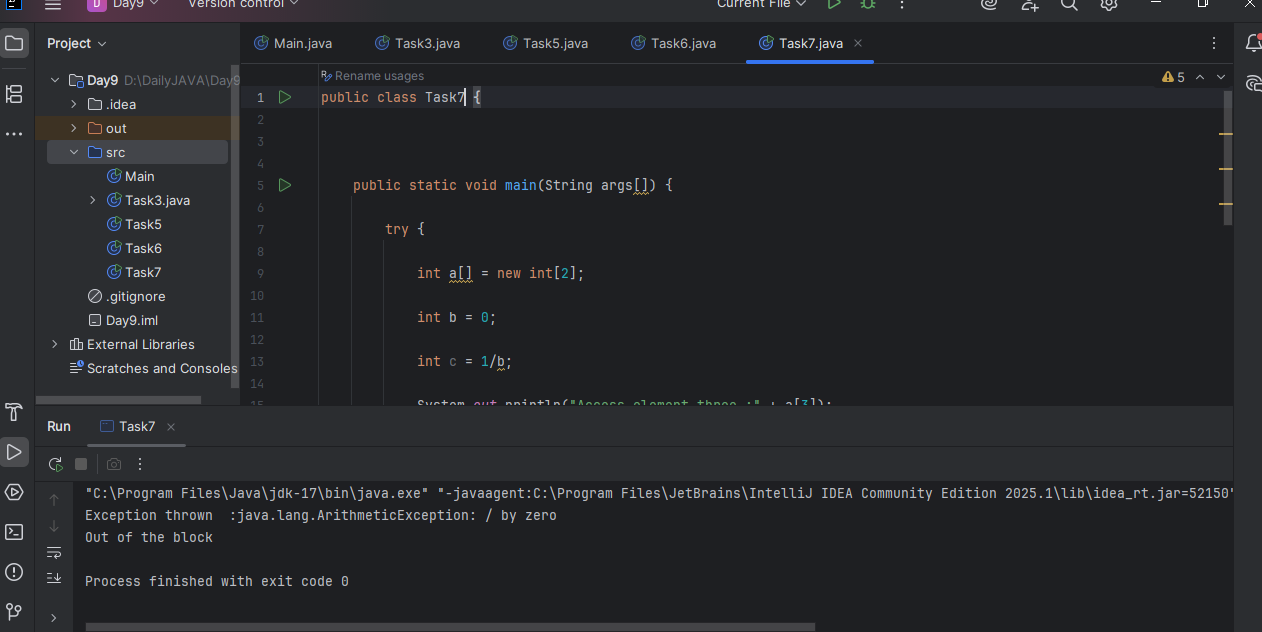
      System.out.println("Out of the block");

   }

}

Answer:

Exception thrown  :java.lang.ArithmeticException: / by zero  
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");

   }

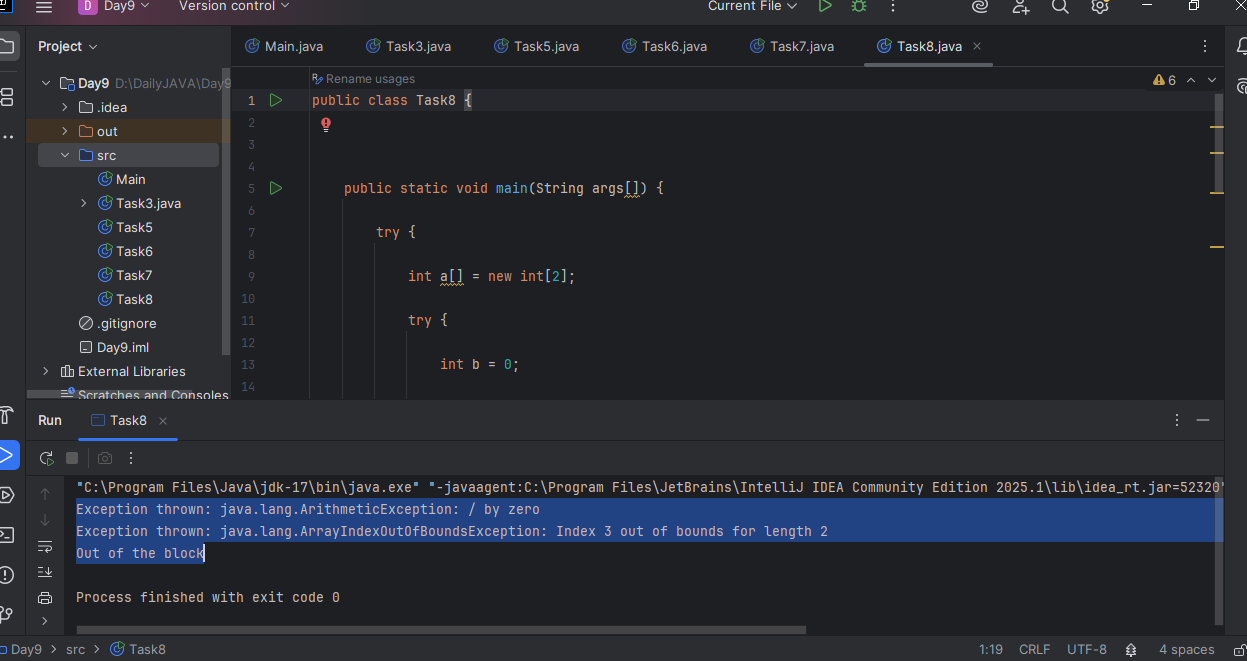
}

Answer:

Exception thrown: java.lang.ArithmeticException: / by zeroException thrown:

java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 2

Out of the block



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.");

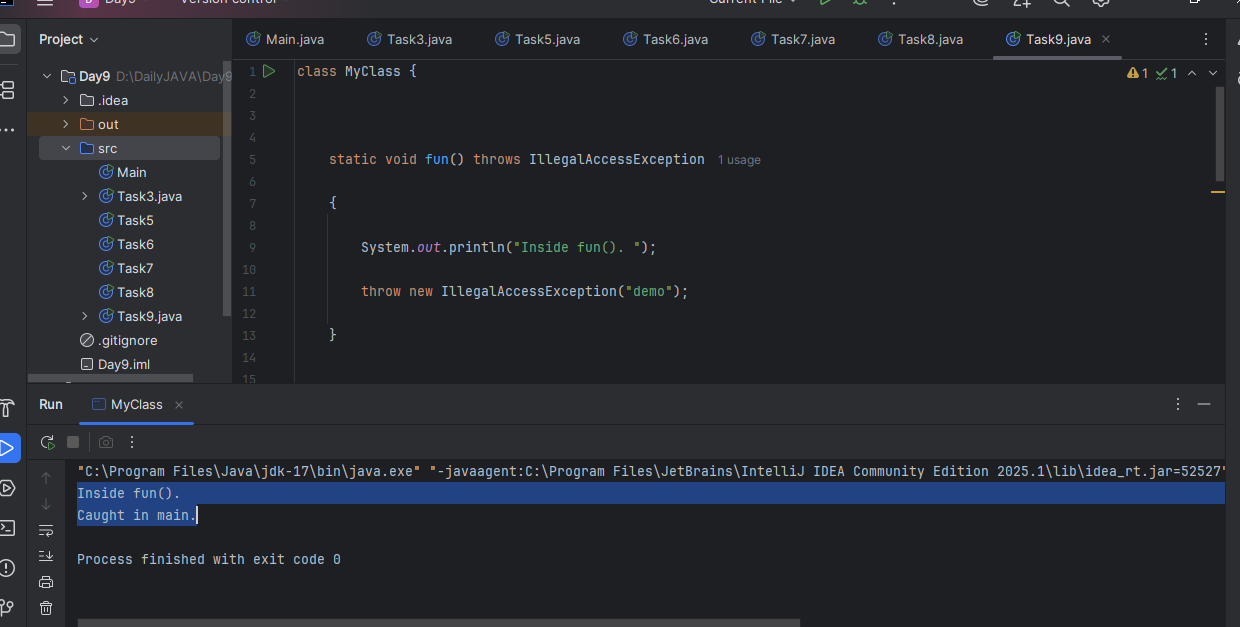
        }

    }

}

Answer:

Inside fun().   
Caught in main.



Task 10:

import java.util.ArrayList;

class Main {

    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

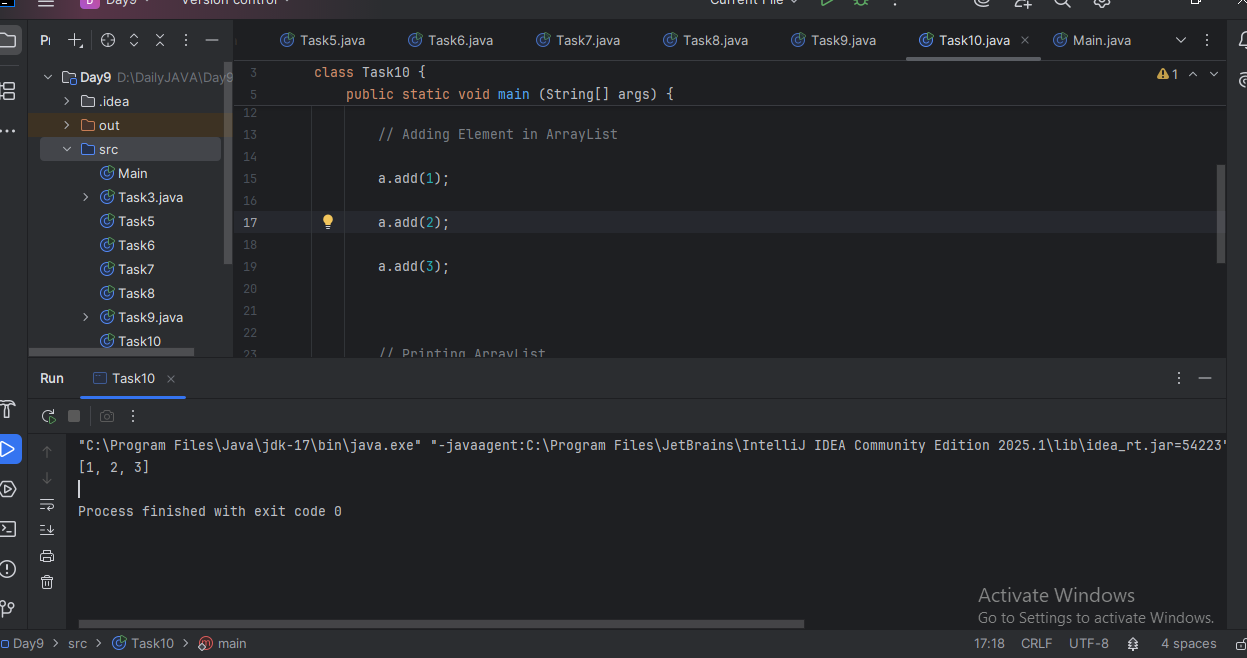
       System.out.println(a);

    }

}

Answer:

[1, 2, 3]



Task 011

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

Answer:

import java.util.ArrayList;

public class Task11 {

    public static void main(String[] args) {

        // Creating an ArrayList of Integers

        ArrayList<Integer> list = new ArrayList<>();

        // Adding 10 elements to the list using a for loop

        for (int i = 1; i <= 10; i++) {

            list.add(i); // Adds numbers 1 to 10

        }

        // Displaying the elements using a for loop

        System.out.println("ArrayList Elements:");

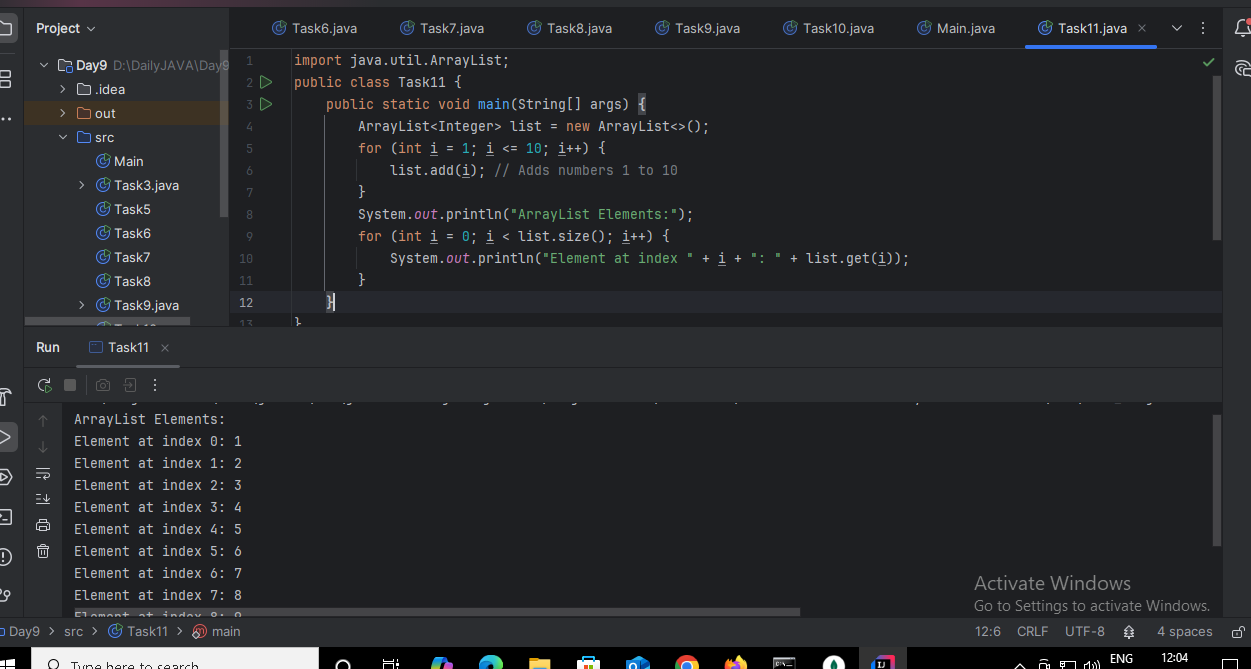
        for (int i = 0; i < list.size(); i++) {

            System.out.println("Element at index " + i + ": " + list.get(i));

        }

    }

}



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);

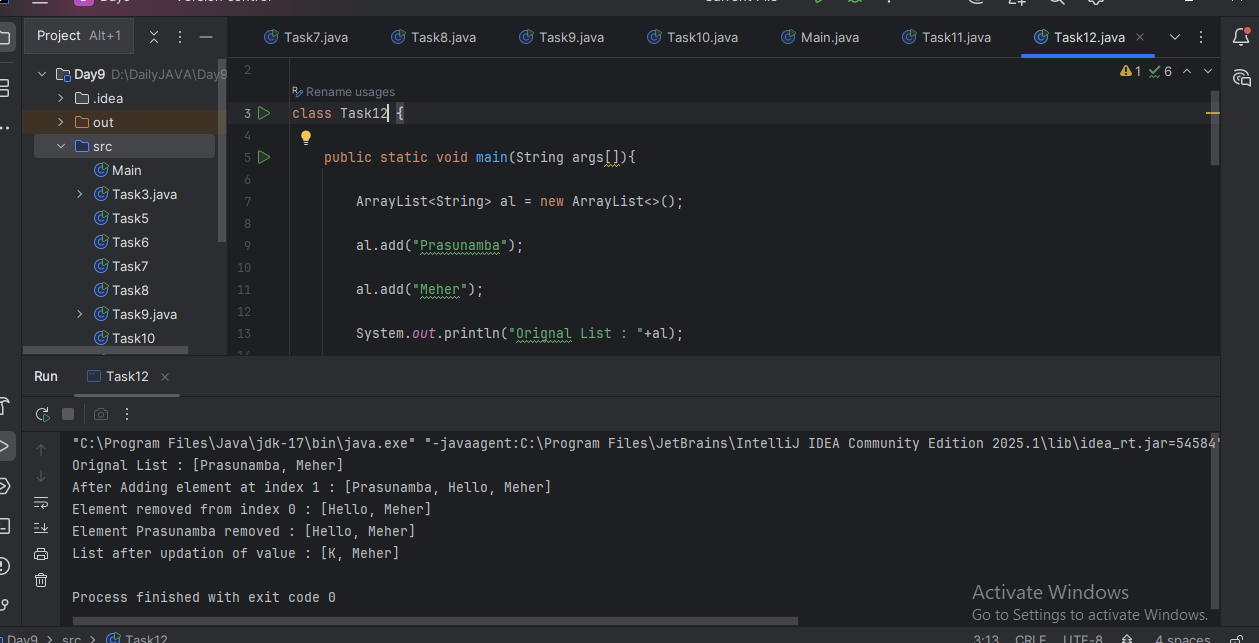
        al.set(0, "K");

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

    }

}

Answer:



Task 013

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 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");

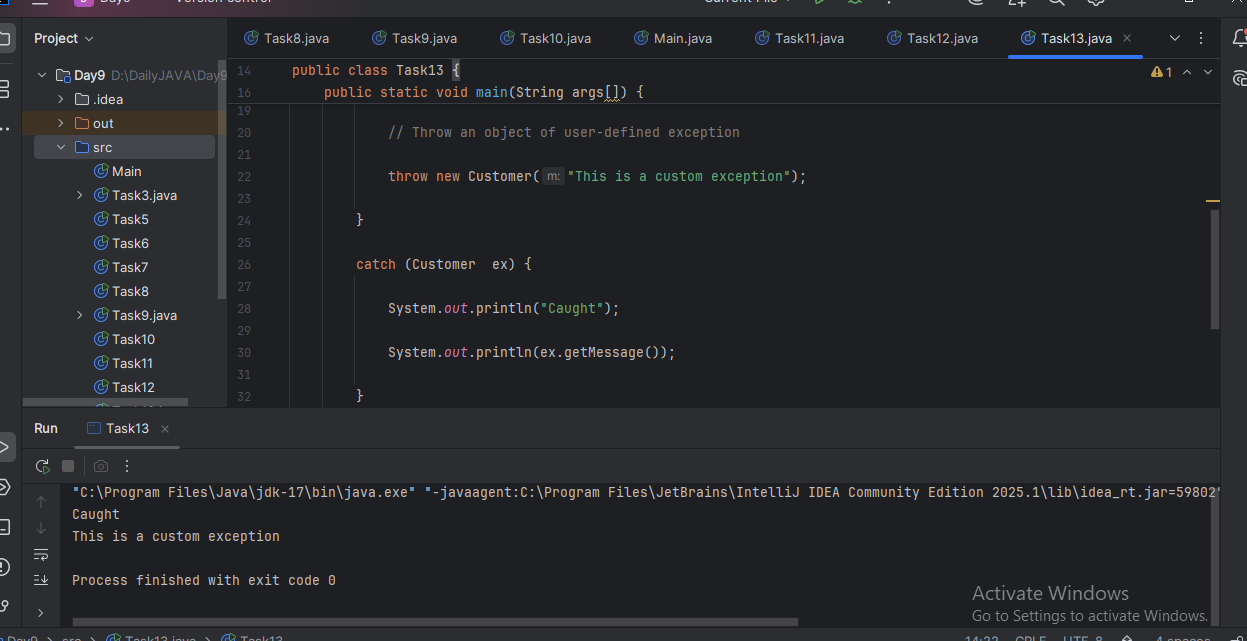
            System.out.println(ex.getMessage());

        }

    }

}

Answer:



Task 014

Inheritance

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

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

Hint : use getter and setters..

Answer:

class Person {

private String name;

private int id;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

@Override

public String toString() {

return "Name: " + name + ", ID: " + id;

}

}

class Employee extends Person {

private String department;

private double salary;

public String getDepartment() {

return department;

}

public void setDepartment(String dept) {

this.department = dept;

}

public double getSalary() {

return salary;

}

public void setSalary(double sal) {

this.salary = sal;

}

@Override

public String toString() {

return super.toString() + ", Department: " + department + ", Salary: " + salary;

}

}

class Manager extends Employee {

private int teamSize;

private String level;

public int getTeamSize() {

return teamSize;

}

public void setTeamSize(int size) {

this.teamSize = size;

}

public String getLevel() {

return level;

}

public void setLevel(String lvl) {

this.level = lvl;

}

@Override

public String toString() {

return super.toString() + ", Team Size: " + teamSize + ", Level: " + level;

}

}

public class Task14 {

public static void main(String[] args) {

Manager manager = new Manager();

manager.setName("Alice");

manager.setId(101);

manager.setDepartment("IT");

manager.setSalary(85000.50);

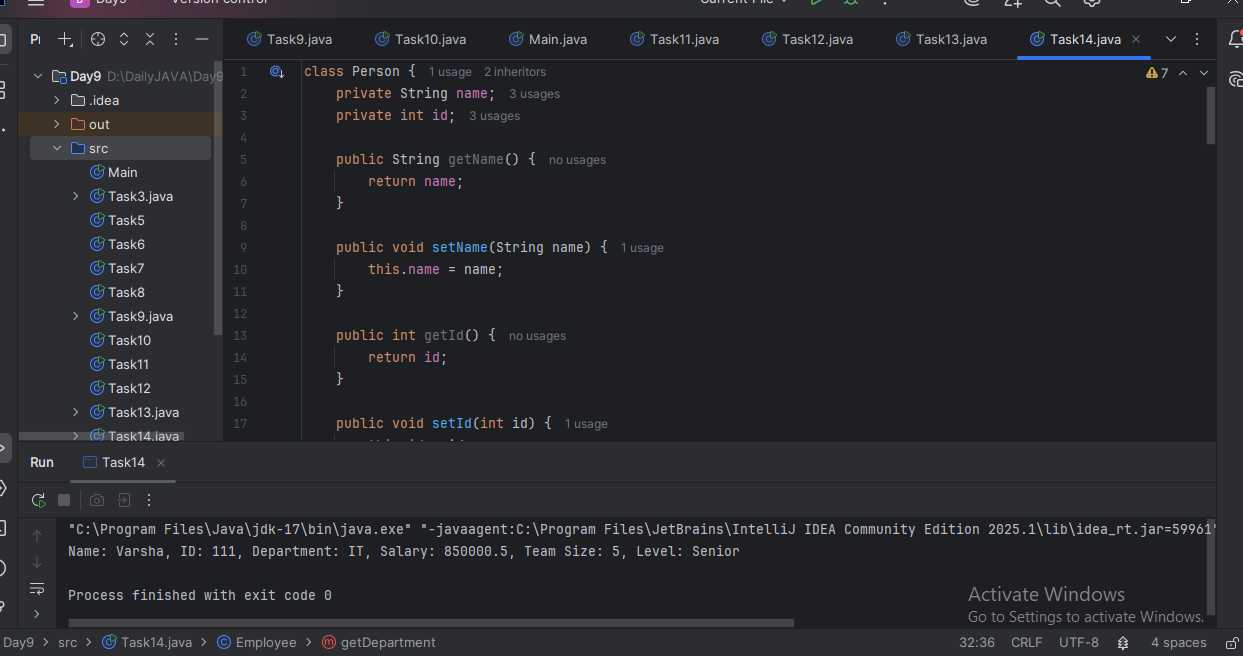
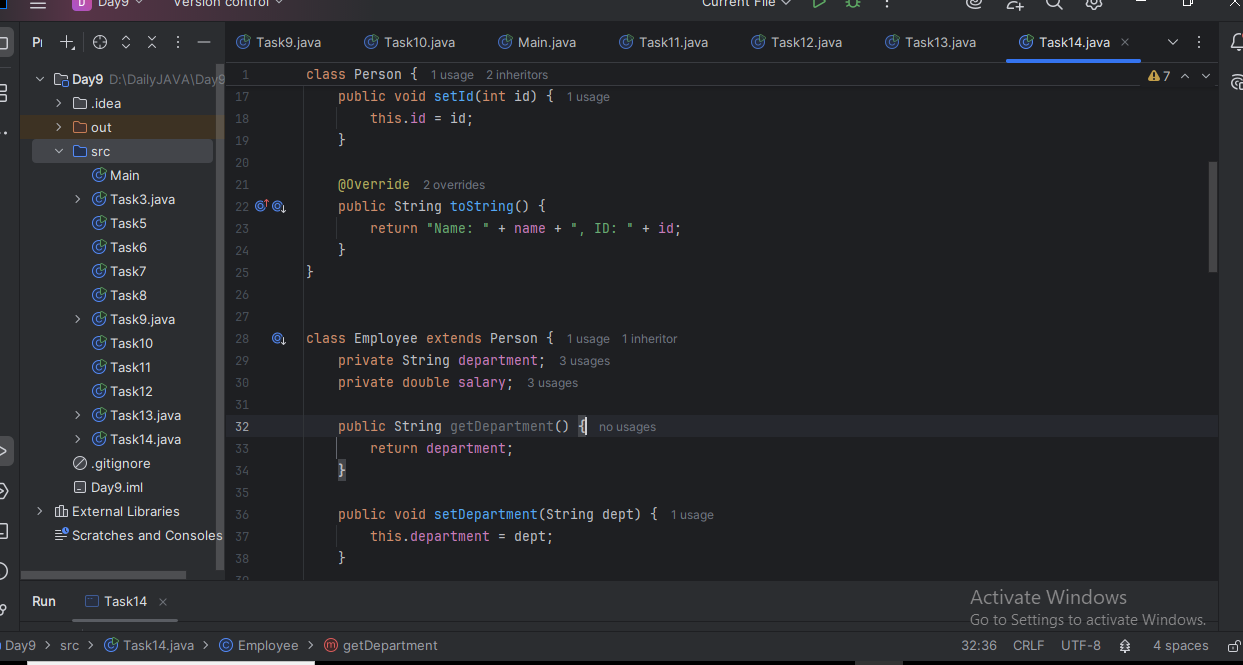
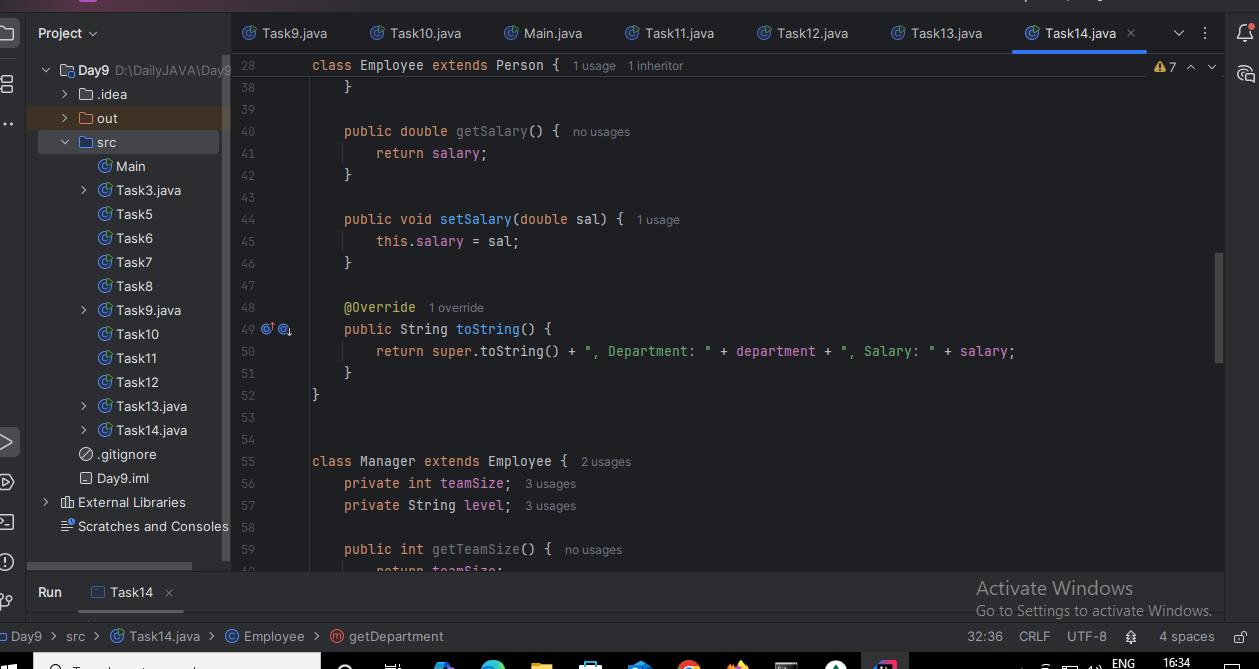
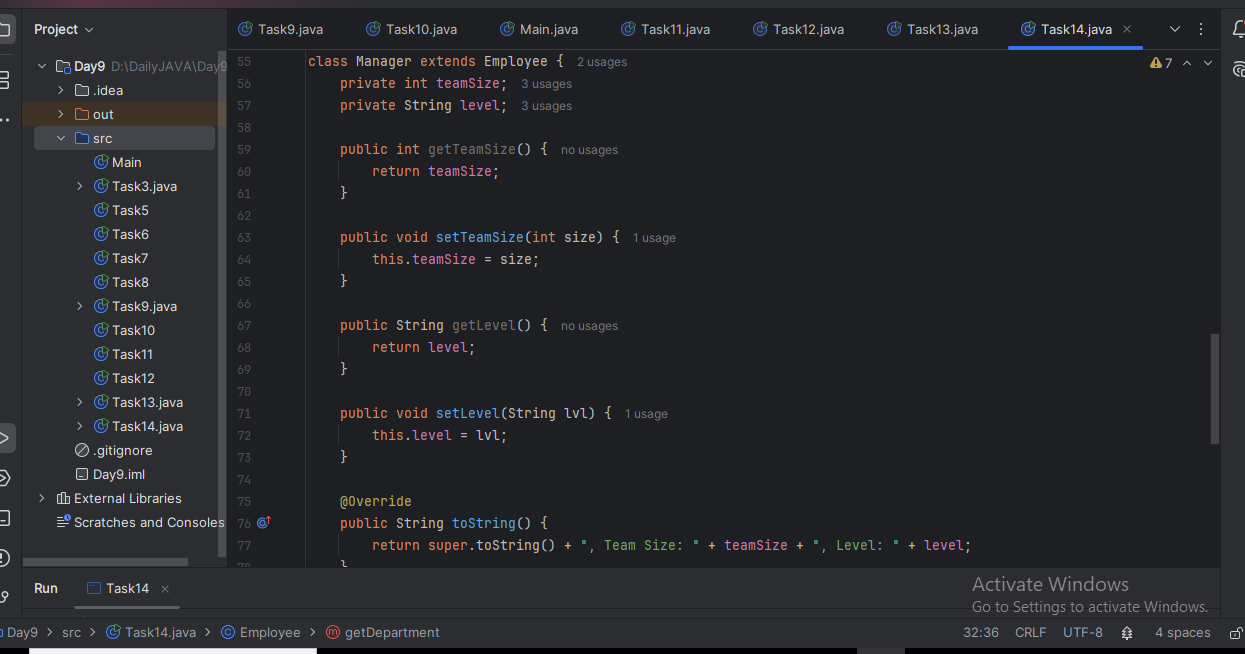
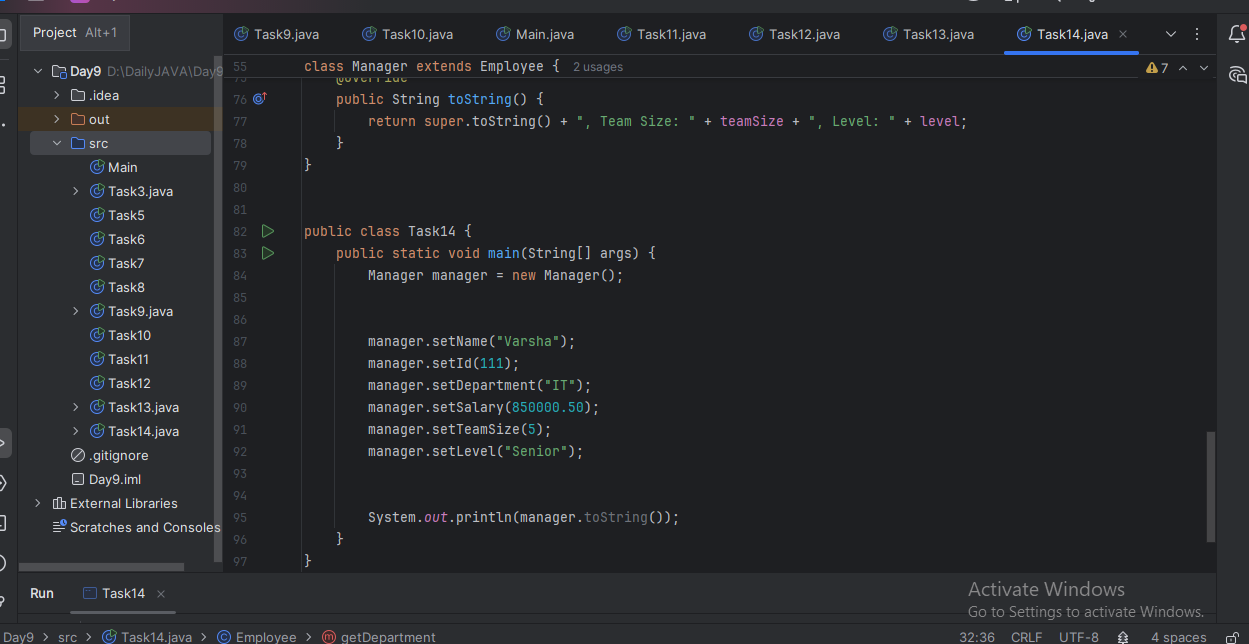
manager.setTeamSize(5);

manager.setLevel("Senior");

System.out.println(manager.toString());

}

}



Inner classes

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();

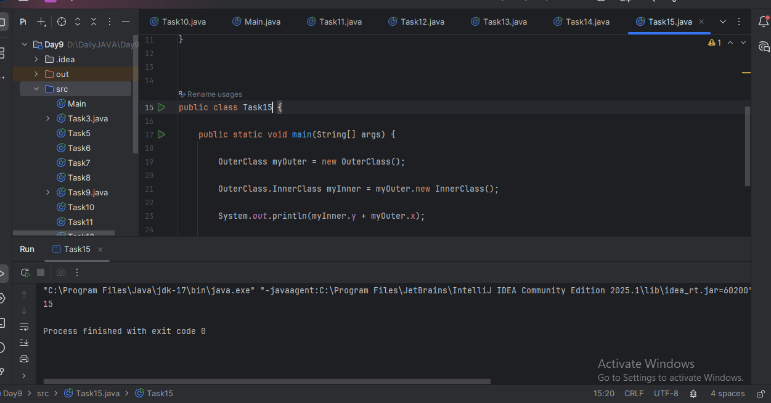
    System.out.println(myInner.y + myOuter.x);

  }

}

Answer: 15

Inner classes can access members of the outer class directly. We can only instantiate a non-static inner class using an instance of the outer class. This is also an example of member inner classes in Java.

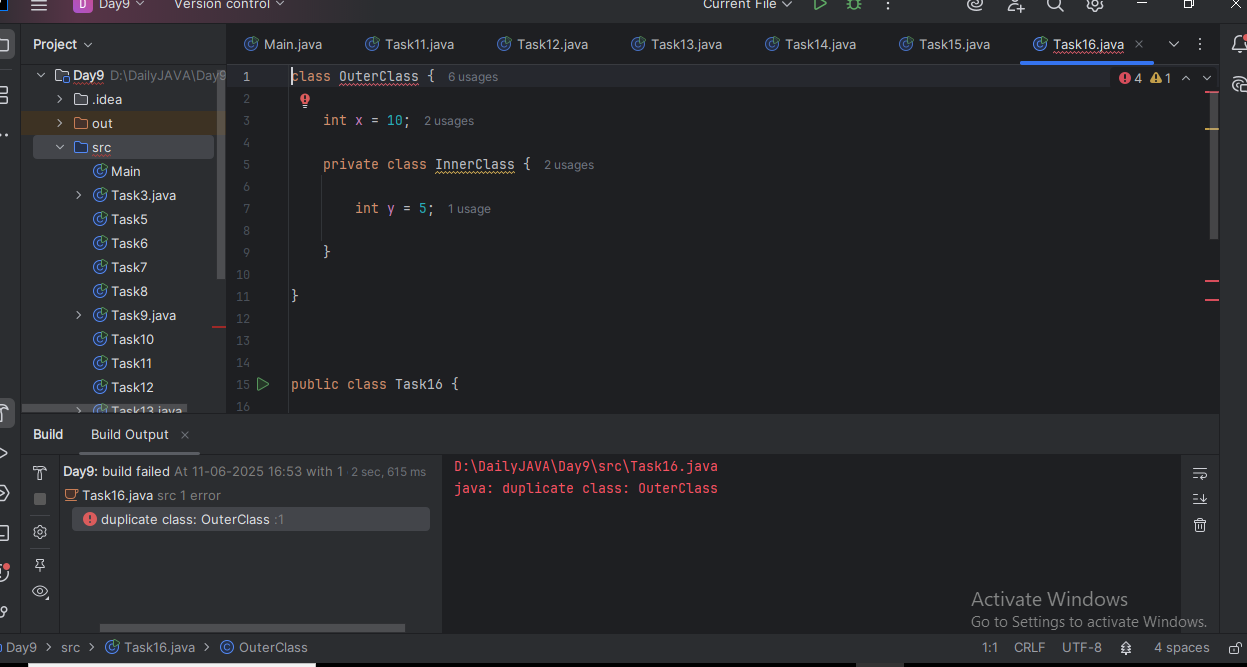


**Task 016**

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

Ex: Private  class InnerClass {

Answer:

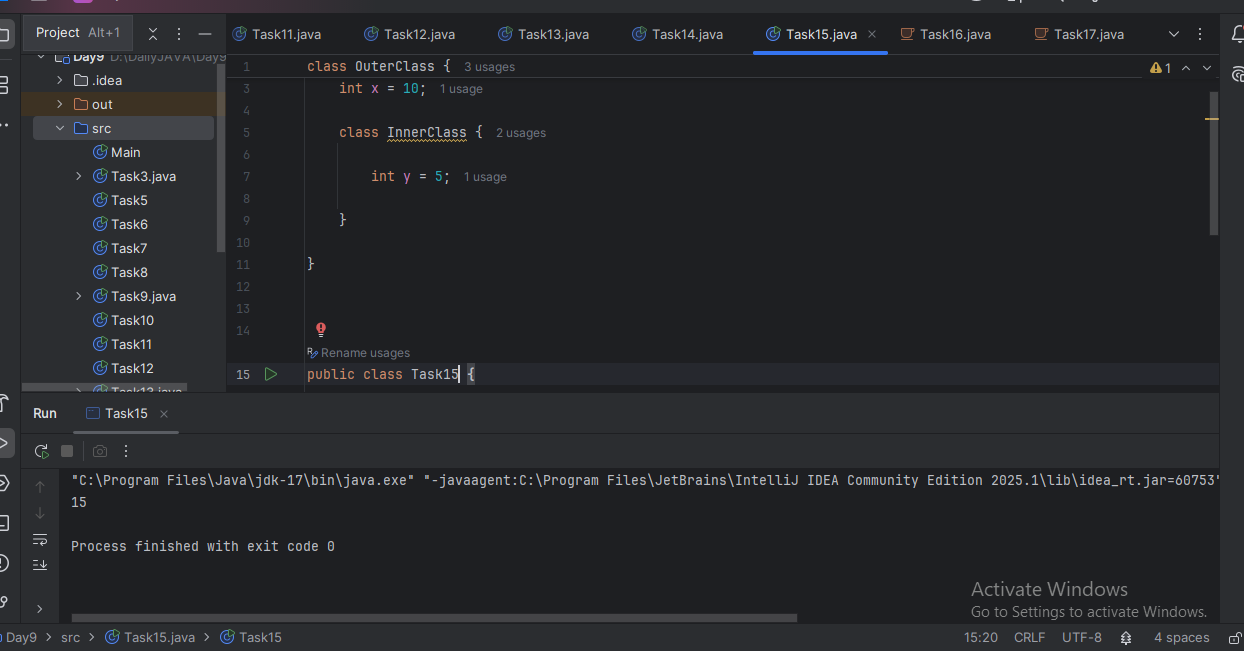


**Task 017**

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

Ex: static class InnerClass {

Answer:



**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() {

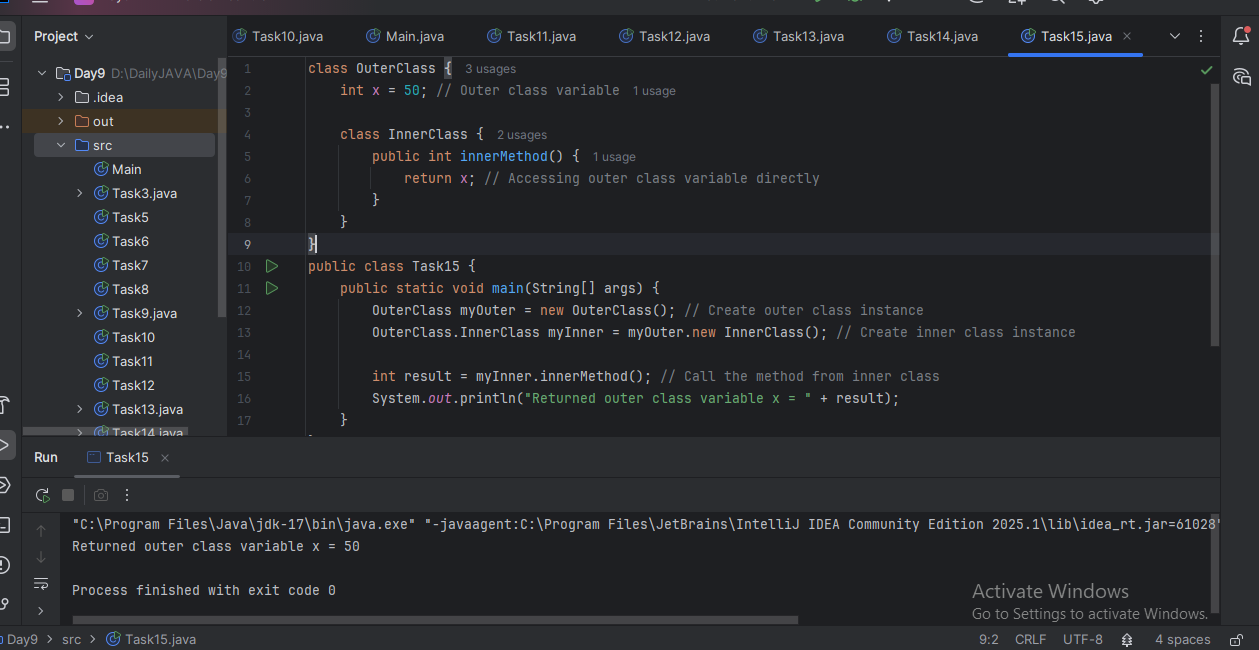
Return x;

}

}

}

Answer:



Task 019  — query by vivek

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);

  }

}

Answer:

