DAY-8(5/6/2025)

Emp ID : 110714877

NAME :Kinnera venkatamma

Task 18. Enums

//Attaching Multiple values

public enum Element {

    H("Hydrogen", 1, 1.008f),

    HE("Helium", 2, 4.0026f),

    // ...

    NE("Neon", 10, 20.180f);

    private static final Map<String, Element> BY\_LABEL = new HashMap<>();

    private static final Map<Integer, Element> BY\_ATOMIC\_NUMBER = new HashMap<>();

    private static final Map<Float, Element> BY\_ATOMIC\_WEIGHT = new HashMap<>();

    static {

        for (Element e : values()) {    //for each loop

            BY\_LABEL.put(e.label, e);

            BY\_ATOMIC\_NUMBER.put(e.atomicNumber, e);

            BY\_ATOMIC\_WEIGHT.put(e.atomicWeight, e);

        }

    }

    public final String label;

    public final int atomicNumber;

    public final float atomicWeight;

    private Element(String label, int atomicNumber, float atomicWeight) {

        this.label = label;

        this.atomicNumber = atomicNumber;

        this.atomicWeight = atomicWeight;

    }

    public static Element valueOfLabel(String label) {

        return BY\_LABEL.get(label);

    }

    public static Element valueOfAtomicNumber(int number) {

        return BY\_ATOMIC\_NUMBER.get(number);

    }

    public static Element valueOfAtomicWeight(float weight) {

        return BY\_ATOMIC\_WEIGHT.get(weight);

    }

}

Task20.

Create an array of your name

Hint : use

Char[] Name = {‘P’, “r’, ….}; // initializing an array

sout(Name);

Int n = Name.length; // size of your name

sout(“there are “+ n +”letters in my name”);

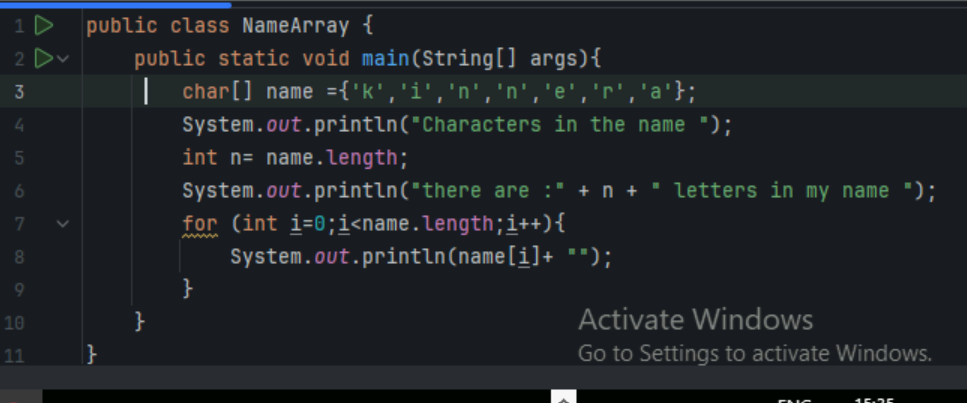
Use for loop to display each letter..

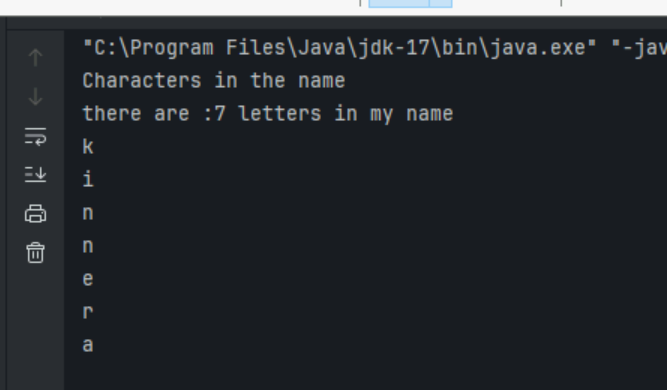
HInt: use ghe below code snippet…

// traversing array

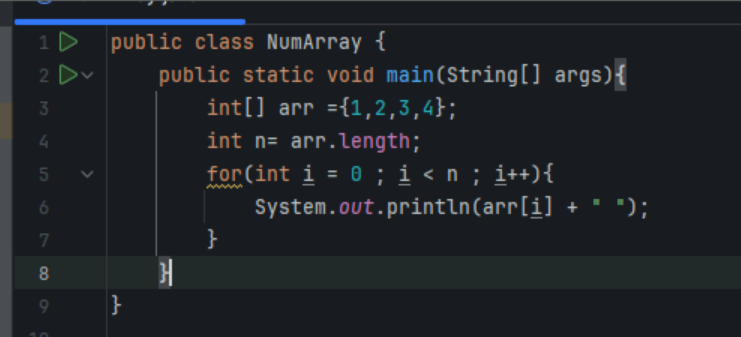
        for (int i = 0; i < n; i++)

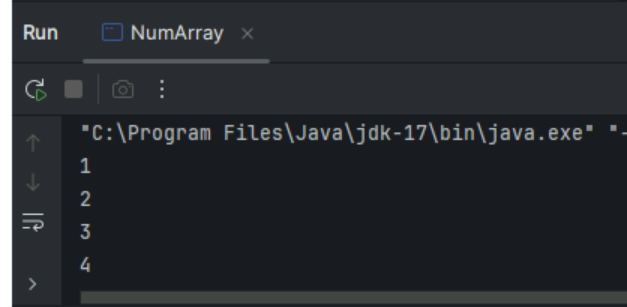
            System.out.print(Name[i] + " ");





Task21. This example demonstrates how to initialize an array and traverse it using a for loop to print each element.



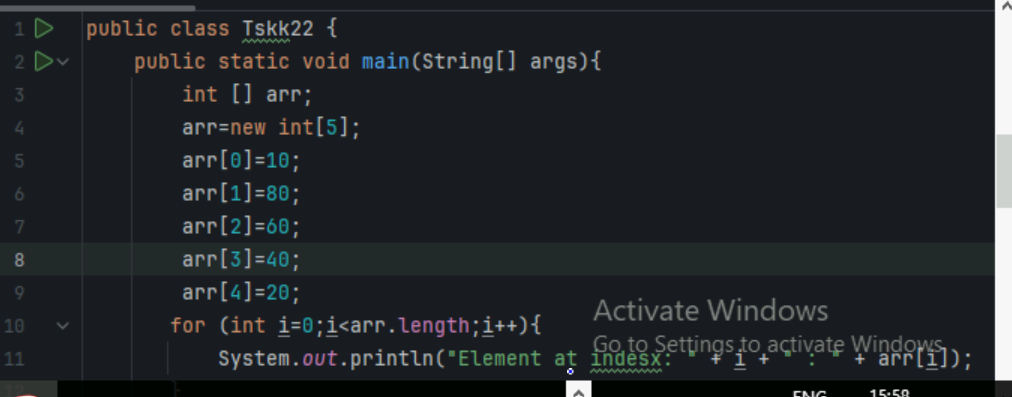


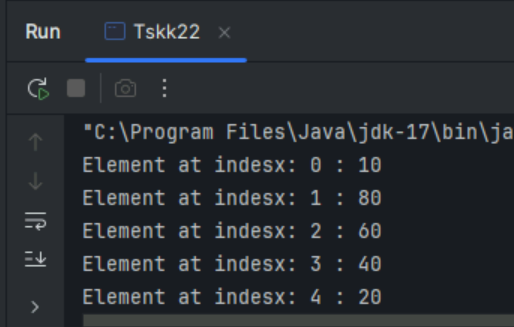
#### Task 22. Implementation:

// Java program to illustrate creating an array

// of integers,  puts some values in the array,

// and prints each value to standard output.

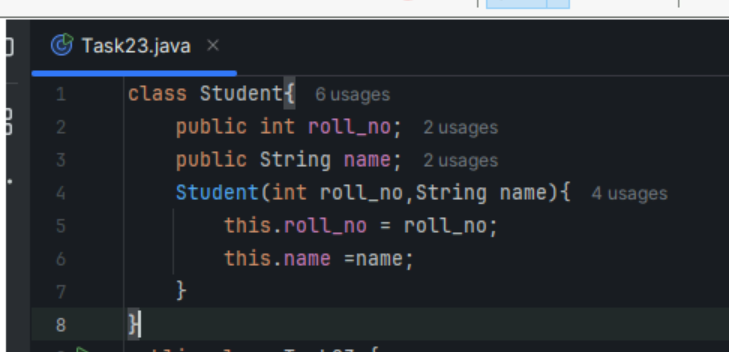




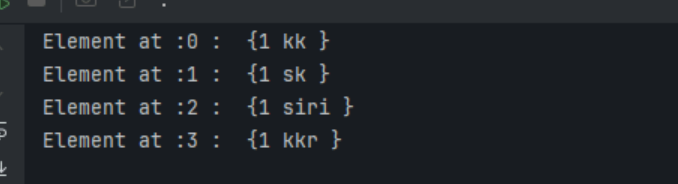
Task23. Here we are taking a student class and creating an array of Student with five Student objects stored in the array. The Student objects have to be instantiated using the constructor of the Student class, and their references should be assigned to the array elements.

// Java program to illustrate creating

// an array of objects



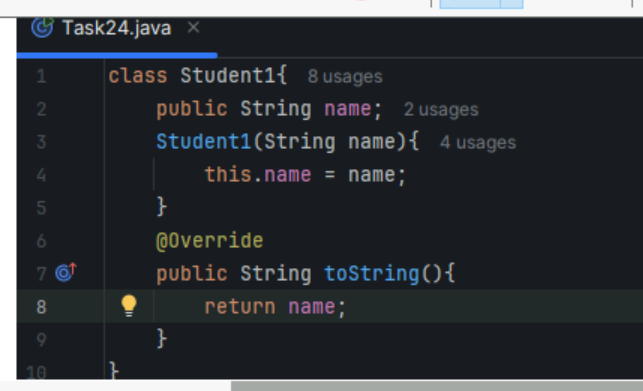


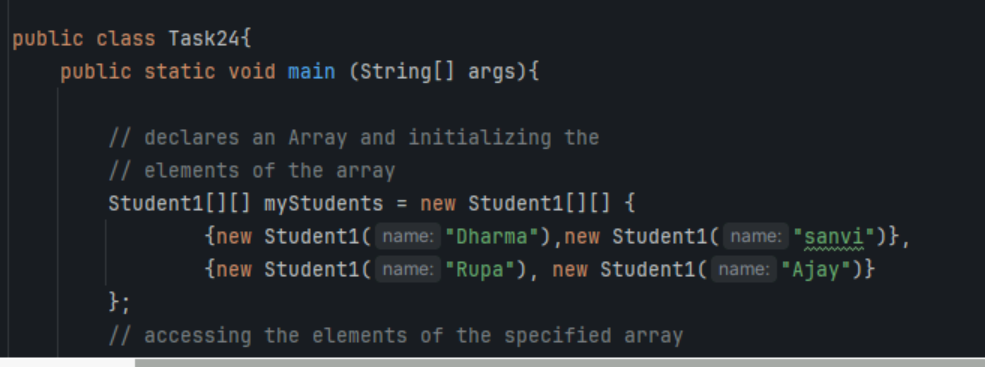


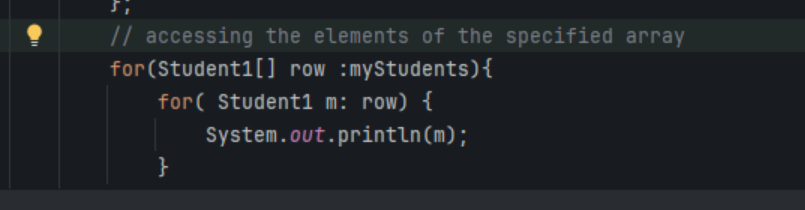
Task24. An array of objects is also created like

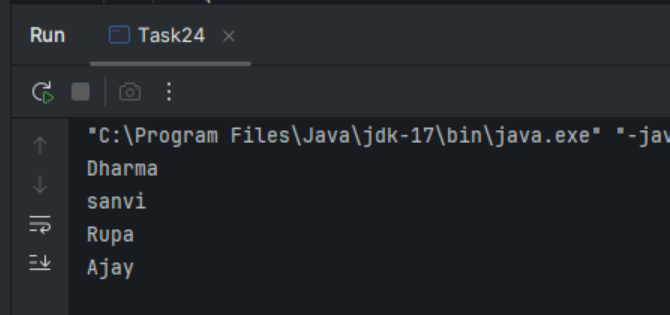
// Java program to illustrate creating

//  an array of objects

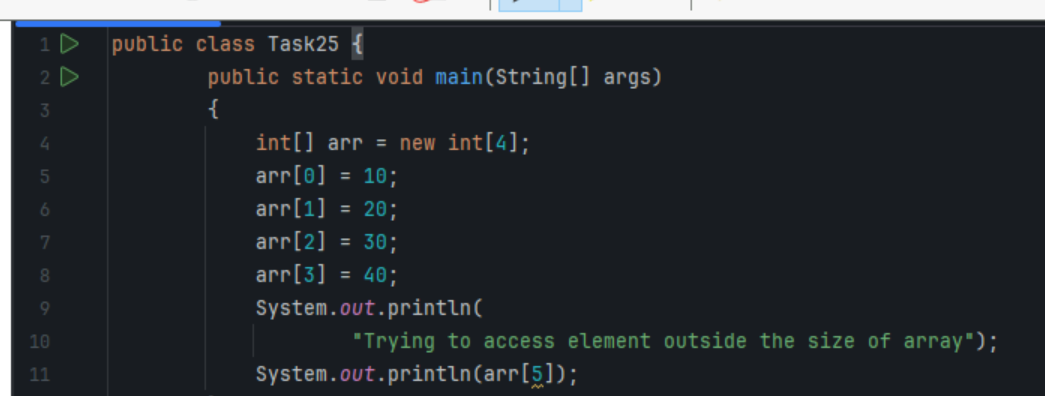


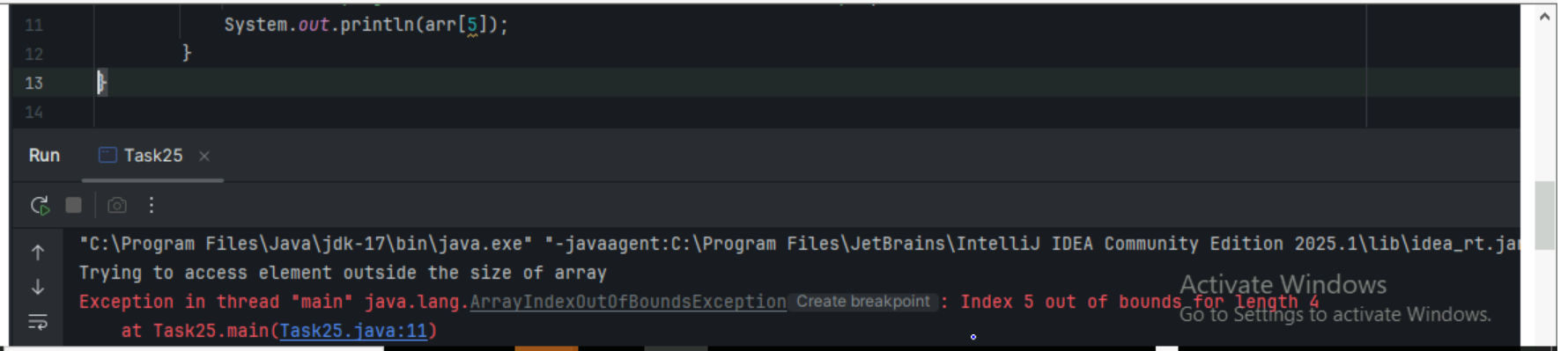






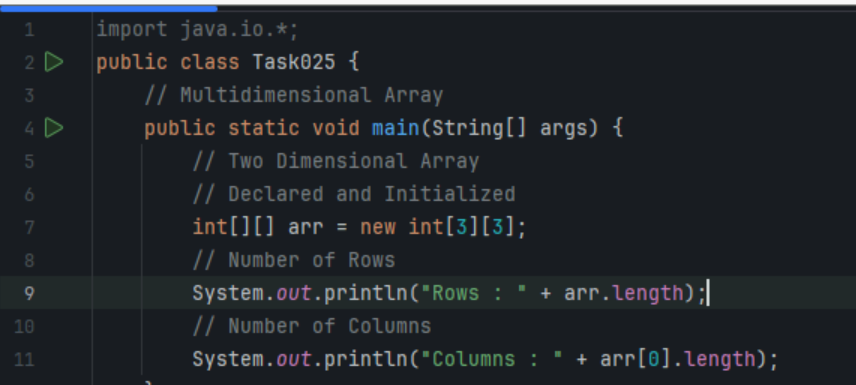
Task 25. Code for showing error "ArrayIndexOutOfBoundsException"

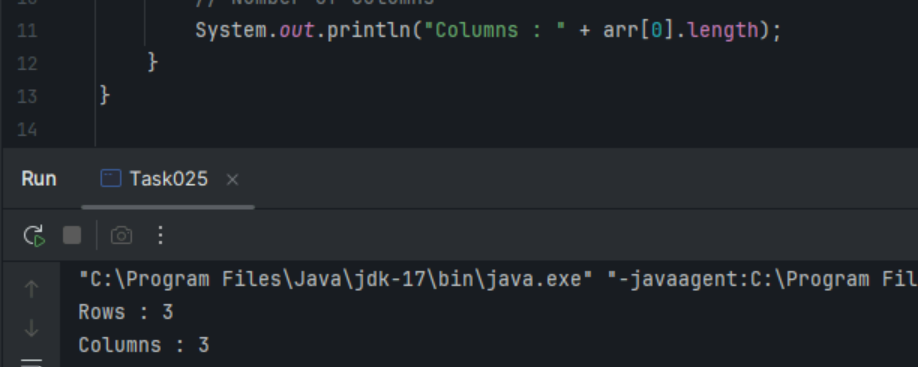




Task 025.  Let us start with basic two dimensional Array declared and initialized.

// Java Program to demonstrate

// Multidimensional Array 

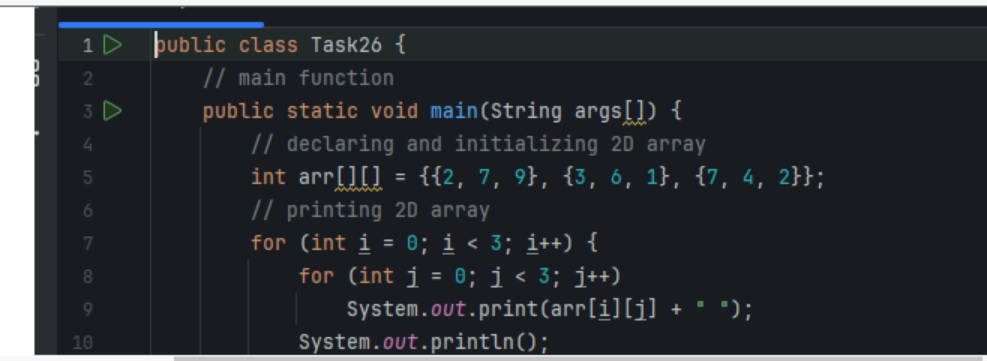


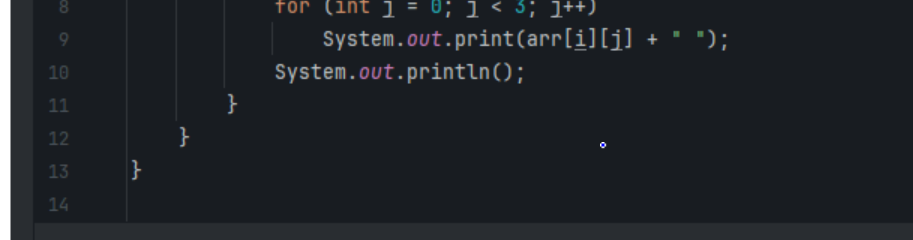
Task 26. Now, after declaring and initializing the array we will check how to Traverse the Multidimensional Array using for loop.

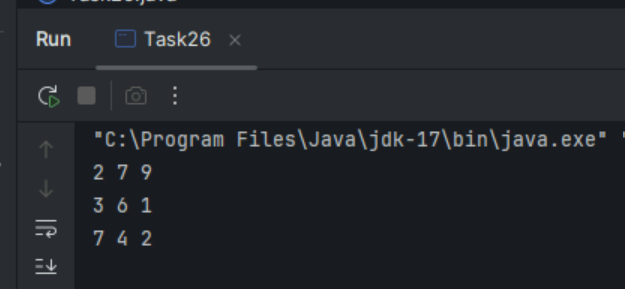
// Java Program to Multidimensional Array

​

// Driver Class

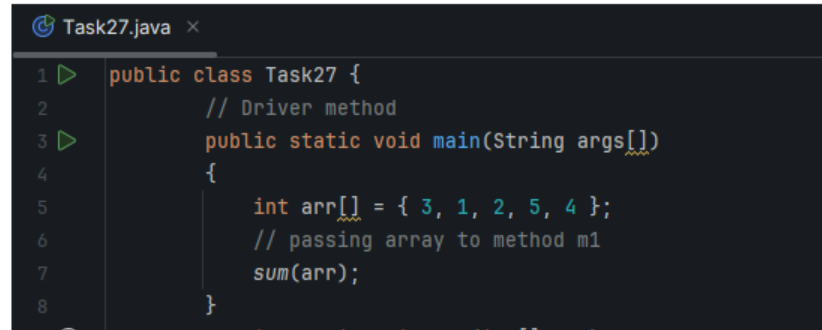
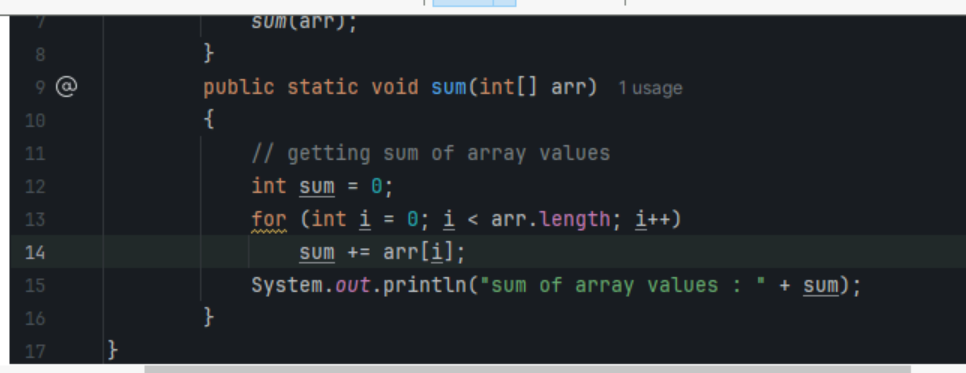


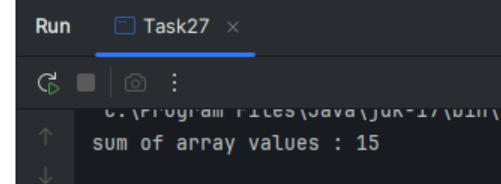


****

**Task 27.** // Java program to demonstrate

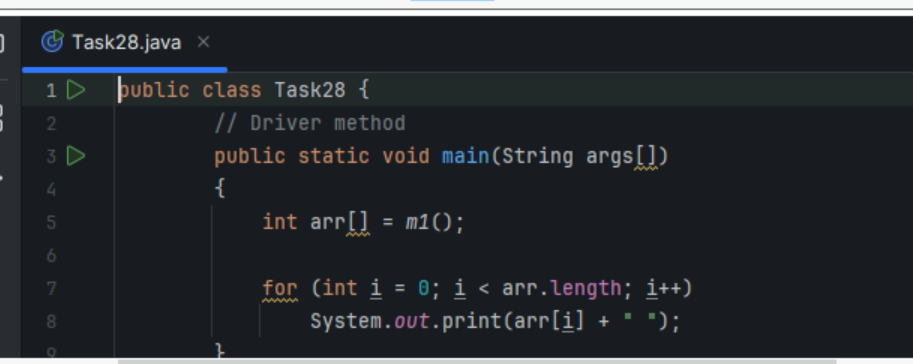
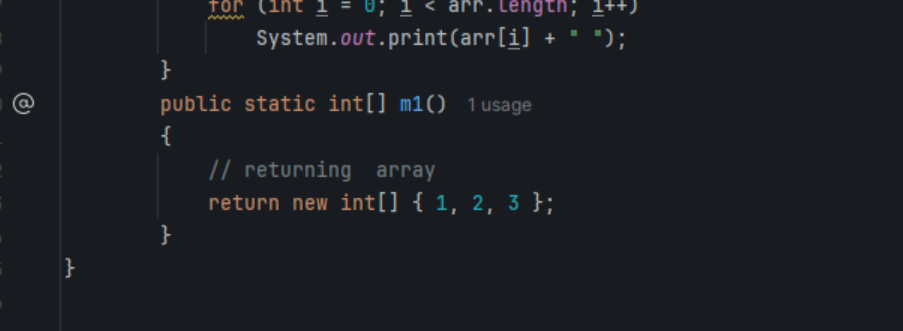
// passing of array to method

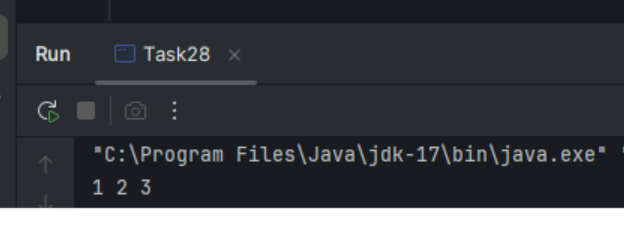
**** ****

****

**Task 28.** // Java program to demonstrate

// return of array from method

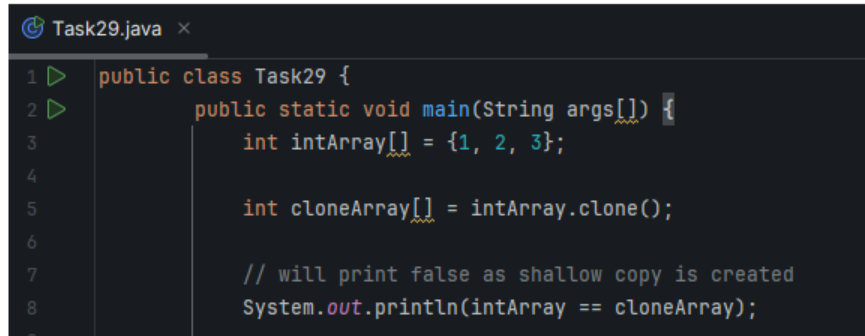
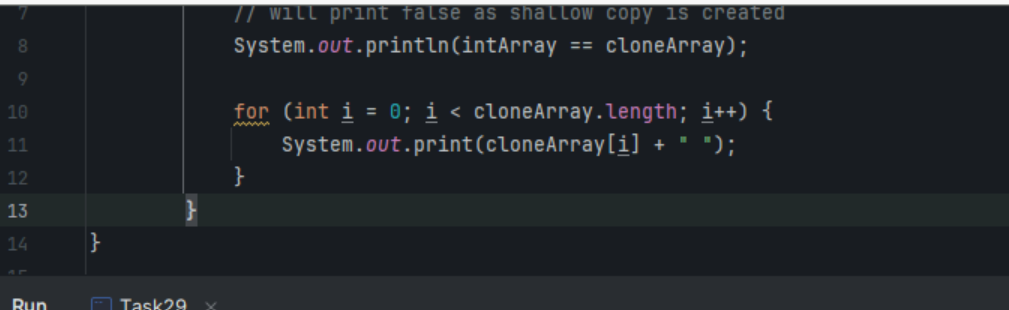
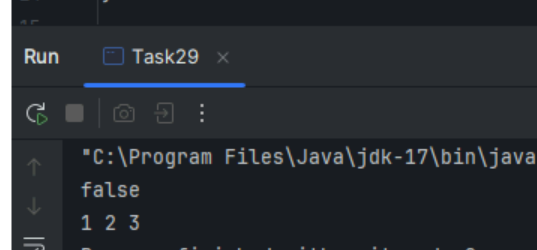
**** ****

****

**Task.** 29

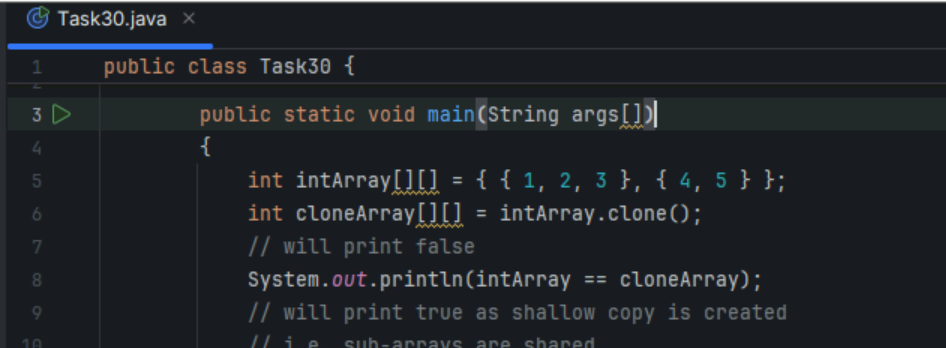
// Java program to demonstrate

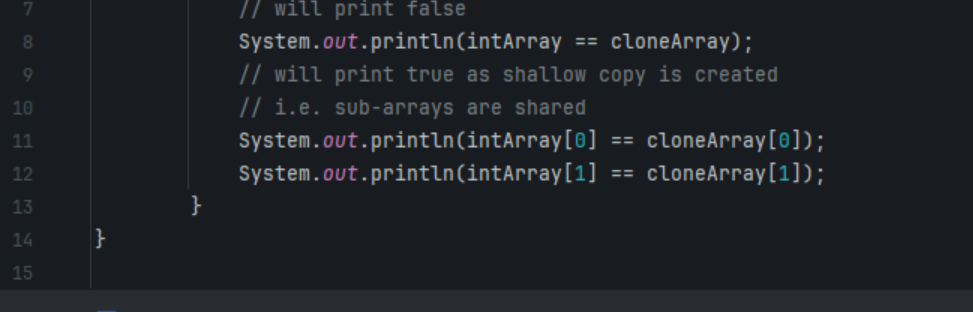
// cloning of one-dimensional arrays

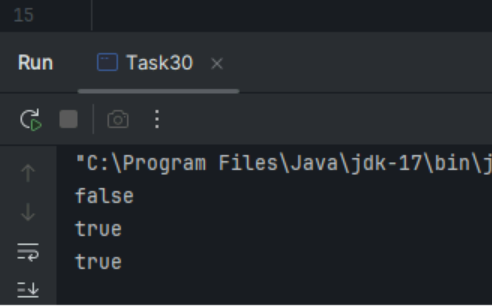
  

**Task 30.** // Java program to demonstrate

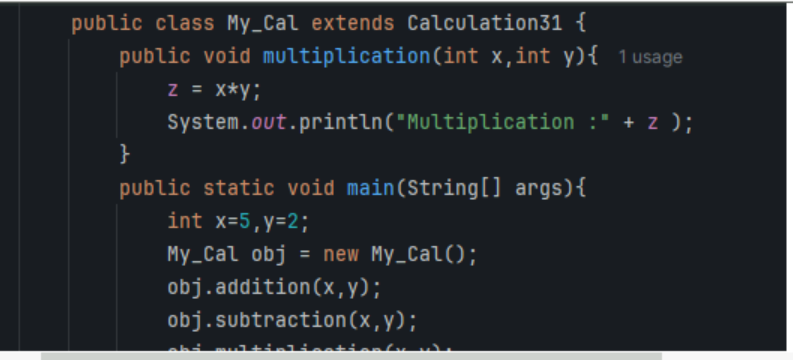
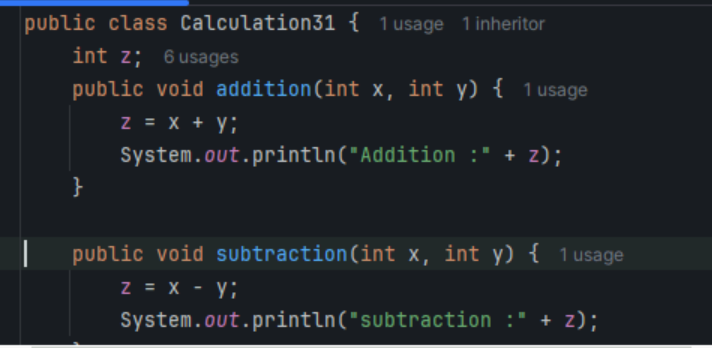
// cloning of multi-dimensional arrays

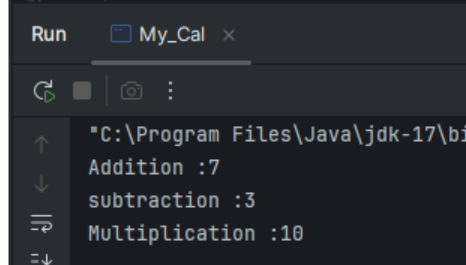






Task 31. Calculation program (Operator using)





Task 32. In the above code add a class clock — and try to extend calculation and clock in the my calculation class..   Is it possible ???? give reason.

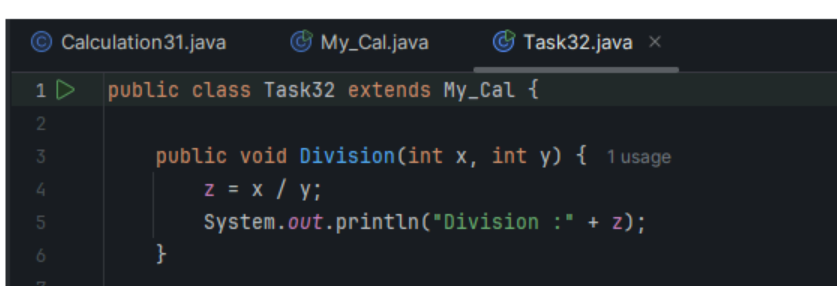
class clock {

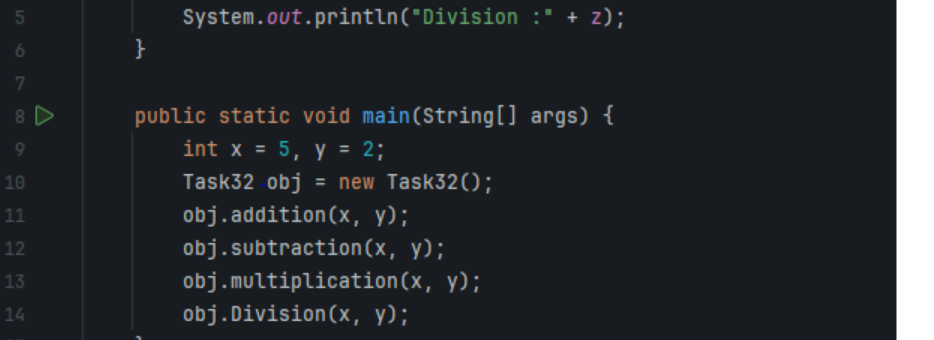
—--

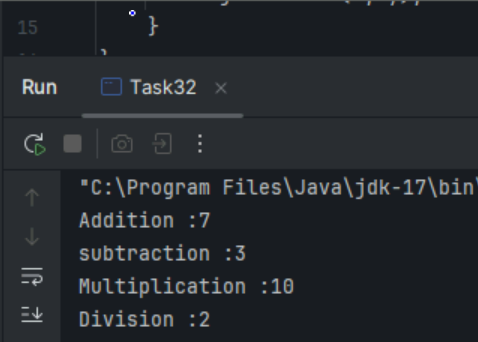
—--

}

class my\_calculation extends calculation , clock{  // multiple inheritance







Task 33. class Customer {

Void purchage\_list{

Int cos = 40t;

String items = “Tomatoes”;

}

}

public class Mart extends Customer {

Void billing(){

String items = “onions”;

Int cost = 30;

}

Psvm (String[] args) {

Super.items = “Potatoes”

Super.cost = 50;

       Sout(items);

sout(cost);

sout”(%%%%%%%%%%%%%%”);

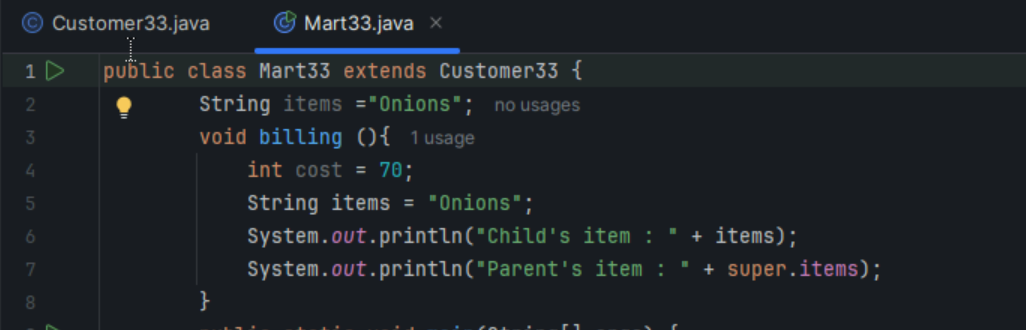
Sout(super.items);

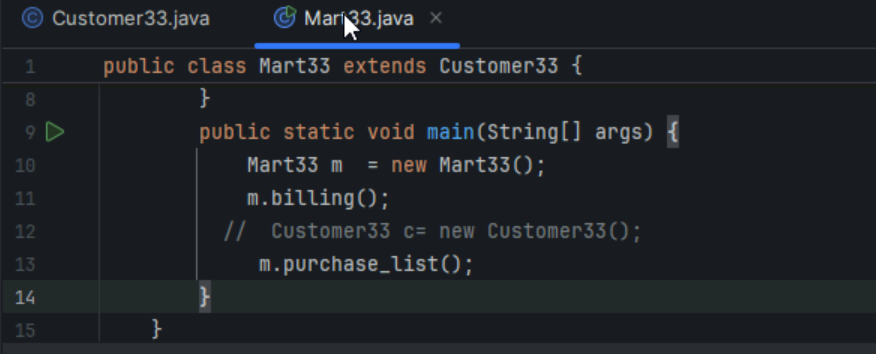
sout(suer.cost);

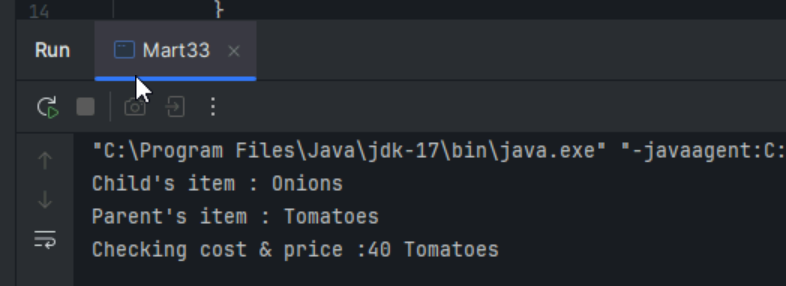
}

}









Task 34. No of parameters:

Void add(int x, int y){

Sout —> x and y values

}

Void add(int x, int y, int z){

Sout —-> x, y, z values

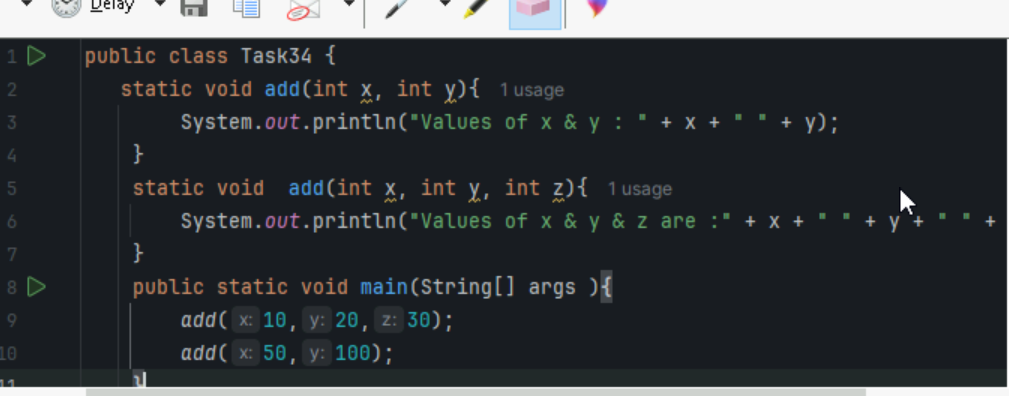
}

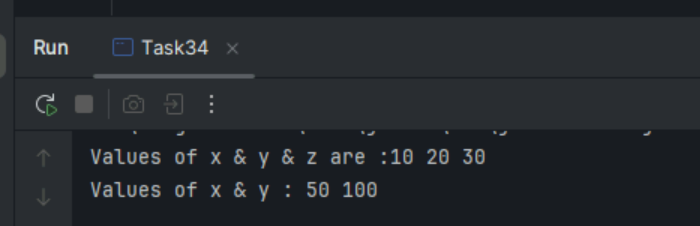
psvm(){

add(10,20,30);

add(50,100);

}





Task 035. Type of parameters

Void add(char x, char y){

Sout —-> x, y values

}

Void add(int x, int y) {

Sout —> x, y values

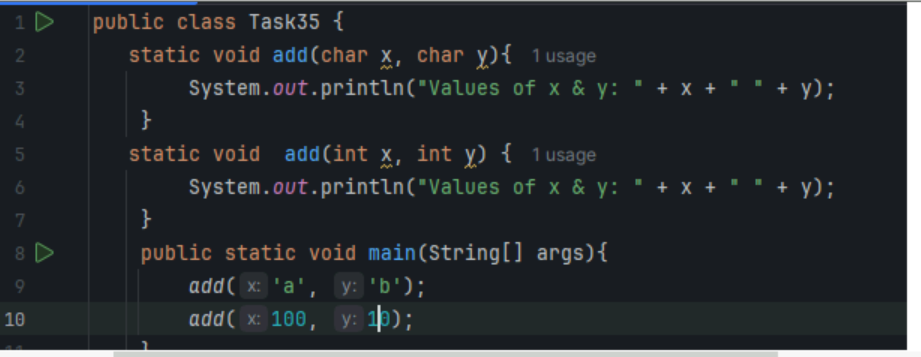
}

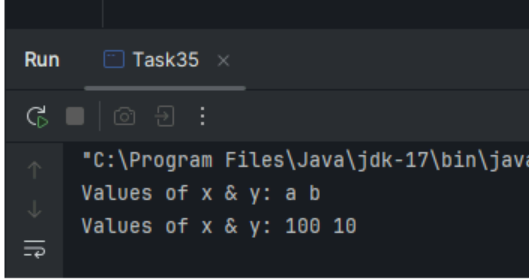
psvm(){

add(‘d’, ‘a’);

add(100, 100);

}





Task 036 : Sequence of Parameters

Void add(int x, float y){

Sout → x, y values

}

Void add(float x, int y){

Sout  → x, y

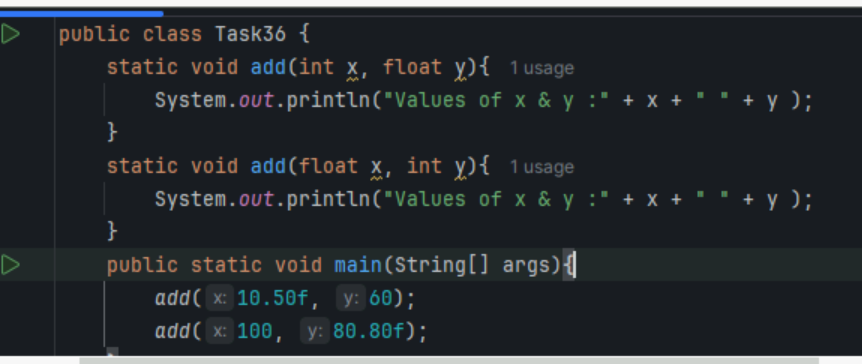
}

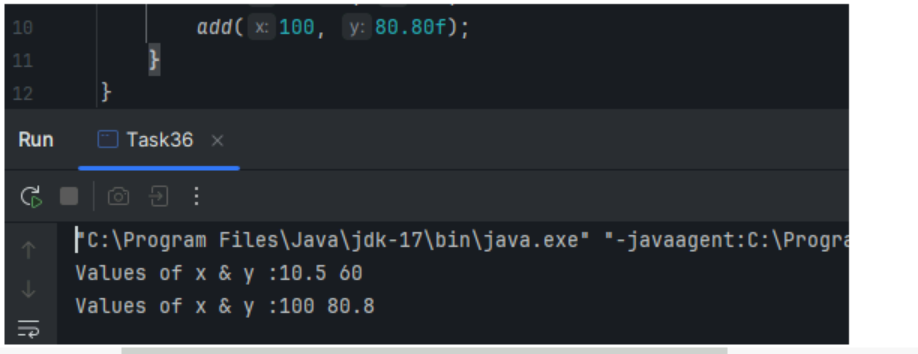
psvm(){

add(10.50f, 60);

add(100, 80.80f)

}





Task 37. Class Employee{

Private int pwd;

Protected int Salary;

Public int empid:

employee(){ // constructors are methods having same name as class name  (we have in c++)

}

~employee(){// destructors used in c++ but not in java

}

}

Class Hr extends Employee {

super.pwd = 1254; //===============>  ??????

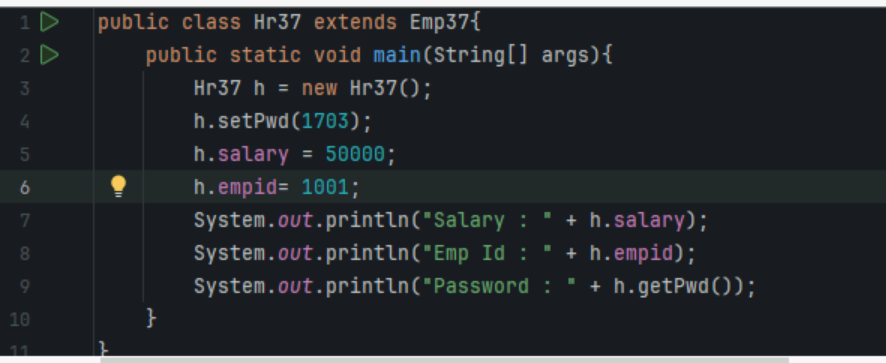
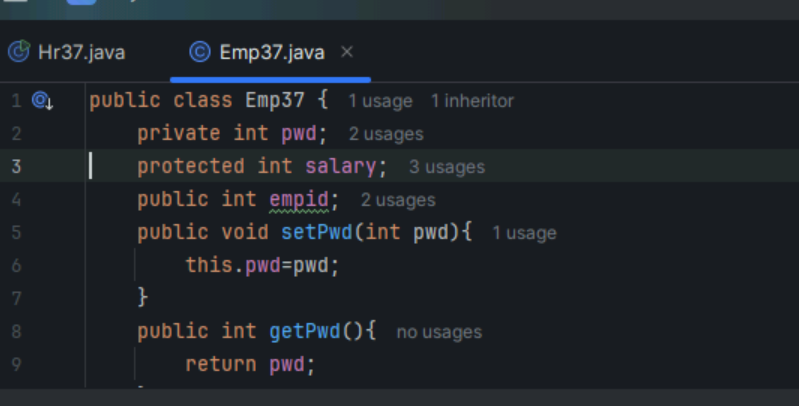
super.Salary = 50000; //==================>  ?

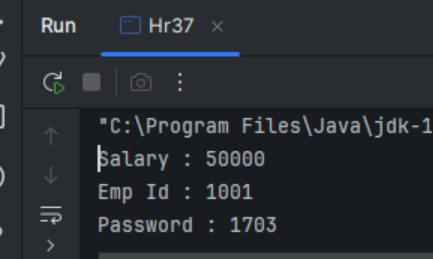
Super.empid = 10001; // ======================>?

psvm(){

}

}





Task 038

/\* File name : AbstractDemo.java \*/

Public class AbstractDemo {

   public static void main(String [] args) {

      /\* Following is not allowed and would raise error \*/

      Employee e = new Employee("George W.", "Houston, TX", 43);

      System.out.println("\n Call mailCheck using Employee reference--");

      e.mailCheck();

   }

}

abstract class Employee {

   private String name;

   private String address;

   private int number;

   public Employee(String name, String address, int number) {

      System.out.println("Constructing an Employee");

      this.name = name;

      this.address = address;

      this.number = number;

   }

   public double computePay() {

     System.out.println("Inside Employee computePay");

     return 0.0;

   }

   public void mailCheck() {

      System.out.println("Mailing a check to " + this.name + " " + this.address);

   }

   public String toString() {

      return name + " " + address + " " + number;

   }

   public String getName() {

      return name;

   }

   public String getAddress() {

      return address;

   }

   public void setAddress(String newAddress) {

      address = newAddress;

   }

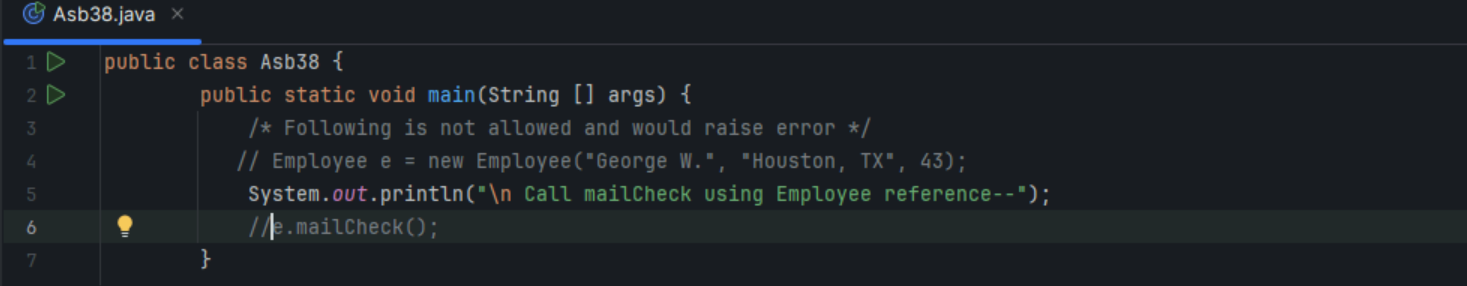
   public int getNumber() {

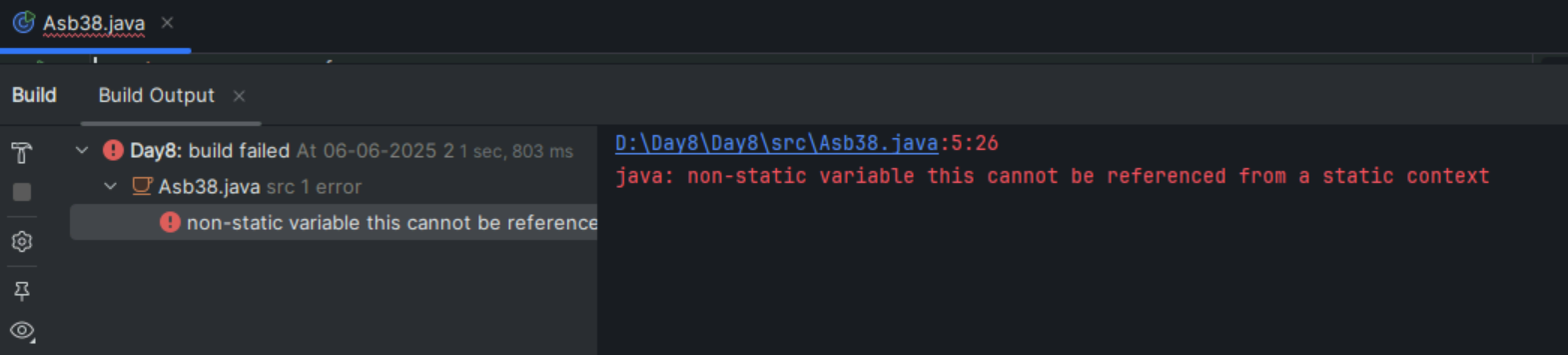
      return number;

   }

}

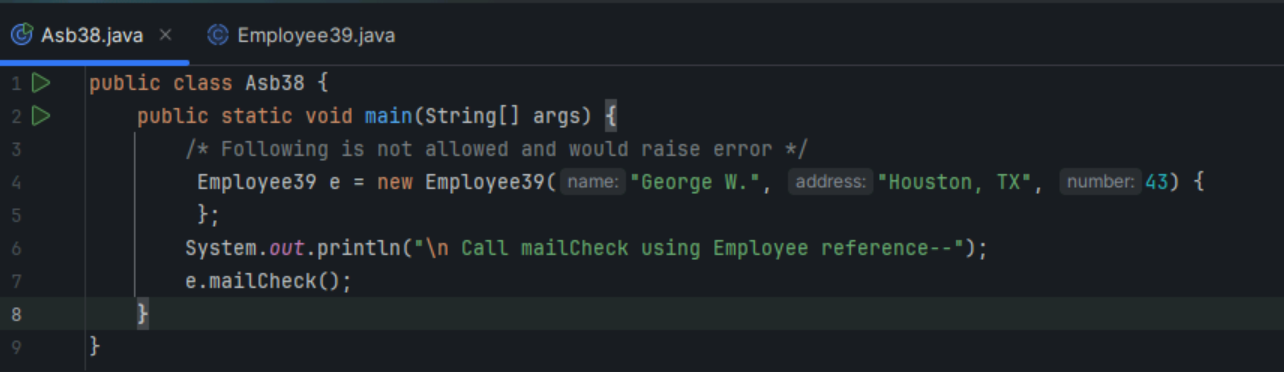
ERROR : Because of line no 4 (creating obj) & 6(using that object) it giving error

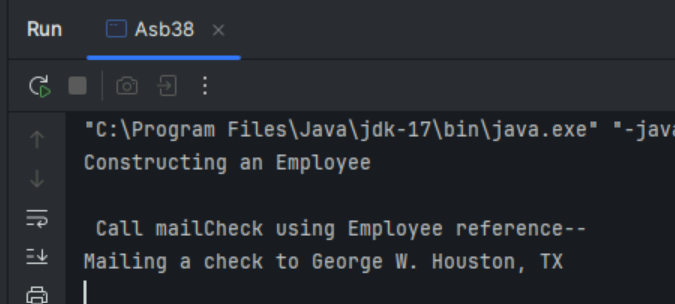




Task 39.Re-write above code without Errors

Here above code divided into 2 classes so that code running without any errors





Task 040

// Working of Abstraction in Java

abstract class Gadgets {

    abstract void turnOn();

    abstract void turnOff();

}

// Concrete class implementing the abstract methods

class TVRemote extends Gadgets {

    @Override

    void turnOn() {

        System.out.println("TV is turned ON.");

    }

    @Override

    void turnOff() {

        System.out.println("TV is turned OFF.");

    }

}

class ACRemote extends Gadgets {

    @Override

    void turnOn() {

        System.out.println("AC is turned ON.");

    }

    @Override

    void turnOff() {

        System.out.println("AC is turned OFF.");

    }

}

// Main class to demonstrate abstraction

public class Main {

    public static void main(String[] args) {

        Gadgets remote = new TVRemote();

        Gadgets remote = new ACRemote();

       remote.turnOn();

        remote.turnOff();

        Gadgets remote = new FanRemote();

        Gadgets remote = new CoolerRemote();

        remote.turnOn();

        remote.turnOff();

    }

}

