Day 8 - 05th June 2025

Enum , gettters and setters (encapsulation),  Arrays, (pending from yesterday)

OOP's concepts Inheritance, Polymorphism, Encapsulation, Abstraction, Interfaces, Exception Handling,

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) {

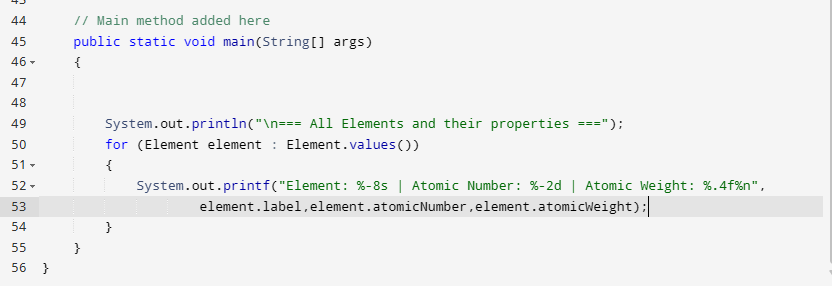
        return BY\_ATOMIC\_WEIGHT.get(weight);

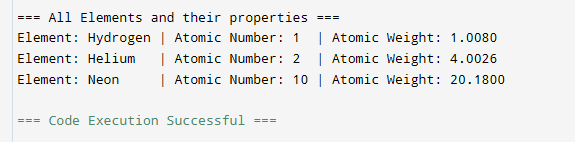
    }

}

Task 19:

Wap to display the content of the above enum.. (main  needs to be added)





Task 017:

Getter and setter

Create a program name Person.java

public class Person {

   private String name;

   // Getter

   public String getName() {

     return name;

   }

   // Setter

   public void setName(String newName) {

     this.name = newName;

   }

}

Create another program named Task017.java

public class Task017{

  public static void main(String[] args) {

    Person myObj = new Person();

    myObj.name = "John";

    System.out.println(myObj.name);

  }

}

**Explanation: Trying to directly access the private field name in the Task017 class, which is not allowed. Private members can only be accessed within the same class where they are declared**.

Task 018

Now create one more program named Task018.java

public class Main {

  public static void main(String[] args) {

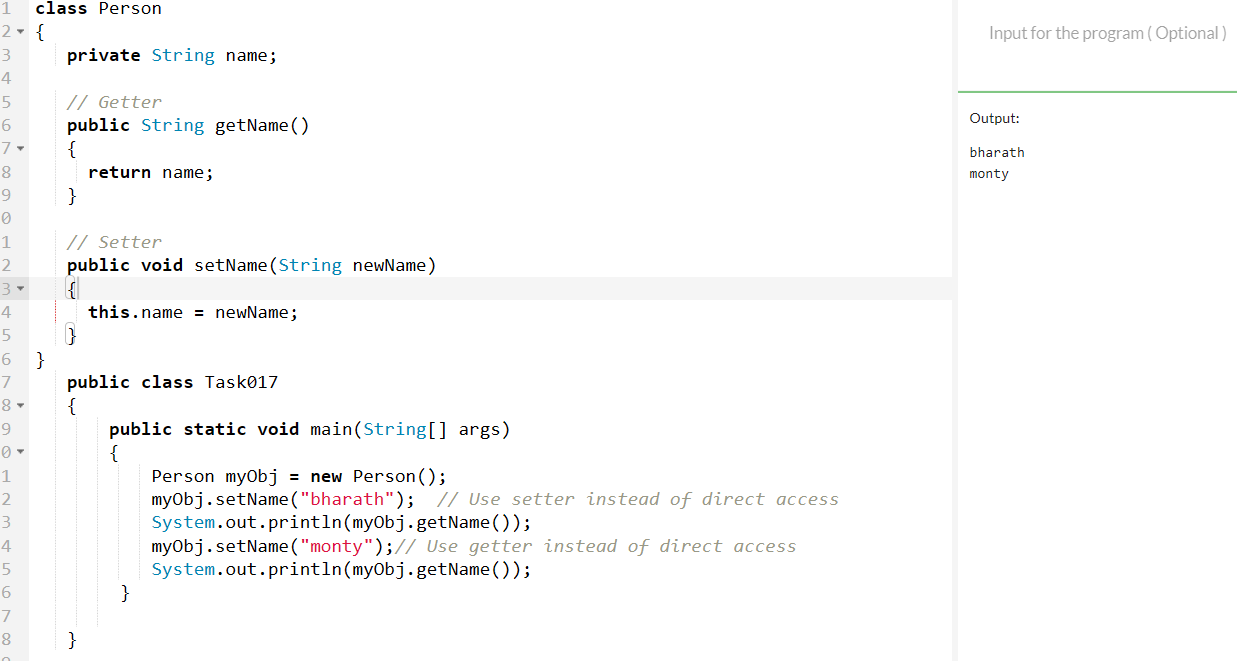
    Person myObj = new Person();

    myObj.setName("John");

    System.out.println(myObj.getName());

  }

}



Task 020:

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”);

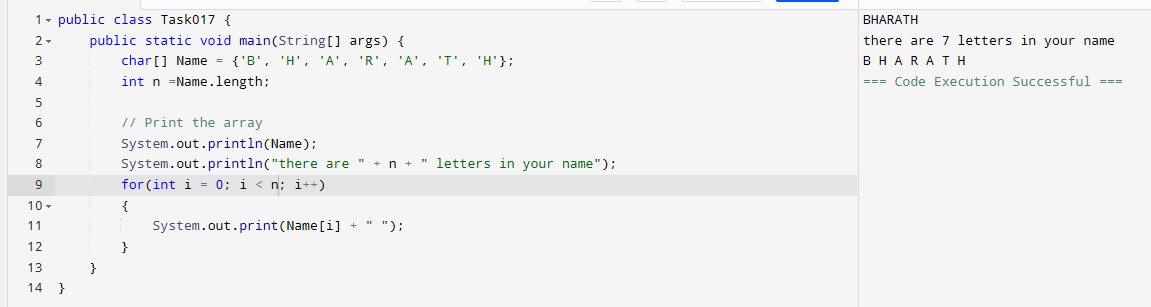
Use for loop to display each letter..

HInt: use ghe below code snippet…

// traversing array

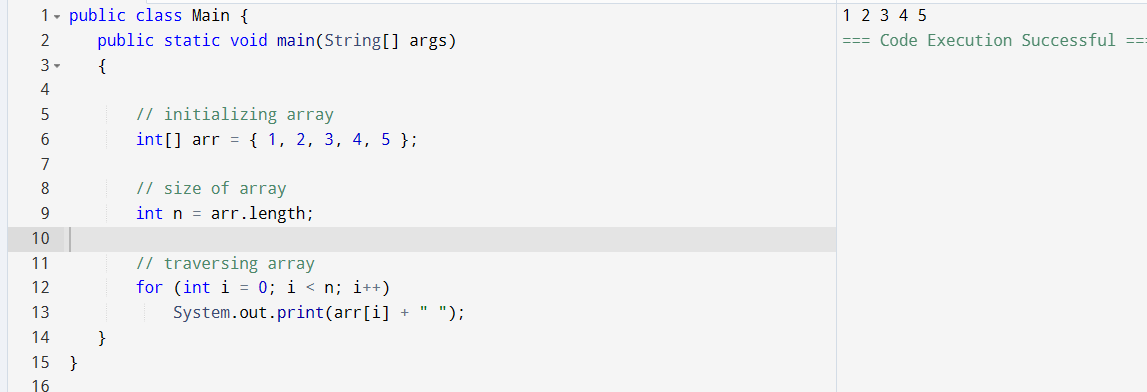
        for (int i = 0; i < n; i++)

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

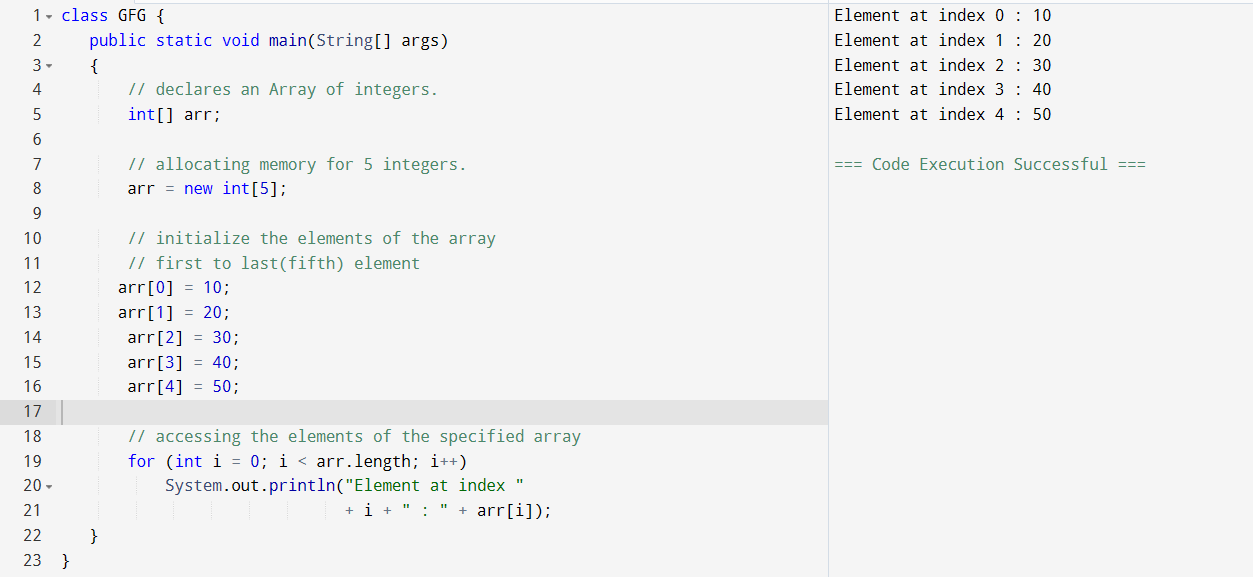


Task 21: Home Task

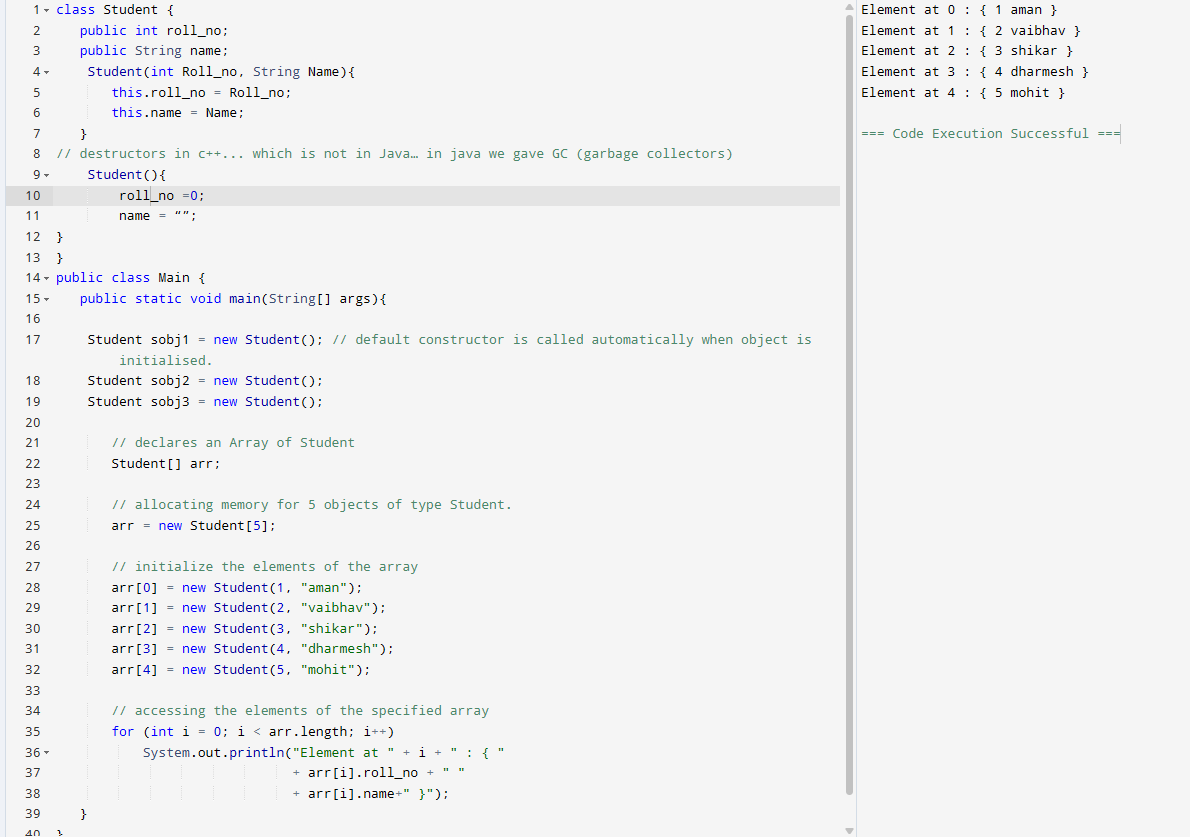
**Example:** This example demonstrates how to initialize an array and traverse it using a for loop to print each element.



*Task 022 - home task*



Task 023 - home task



Task 024 Home task



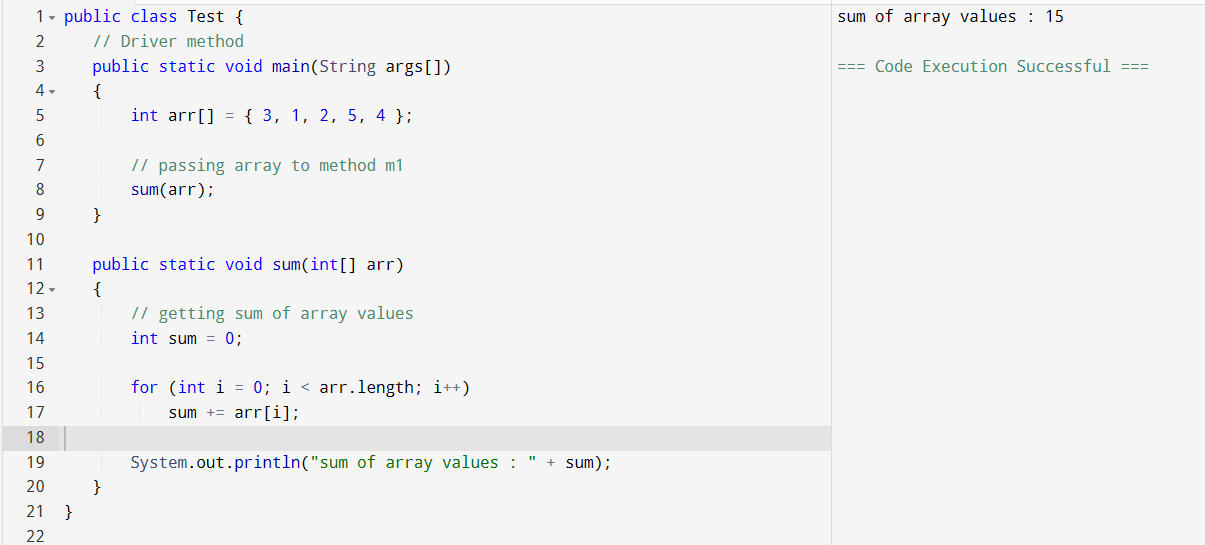
Task 025  - home task

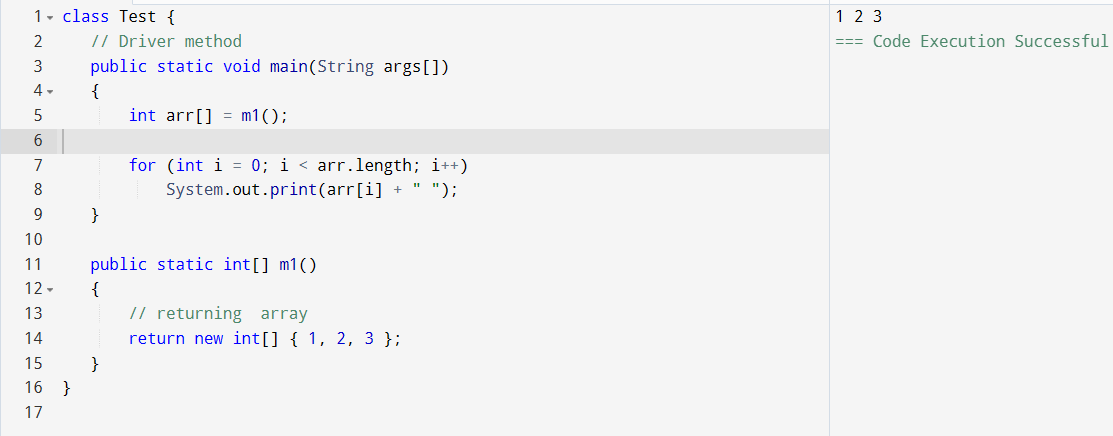


Task 026 - Home Task

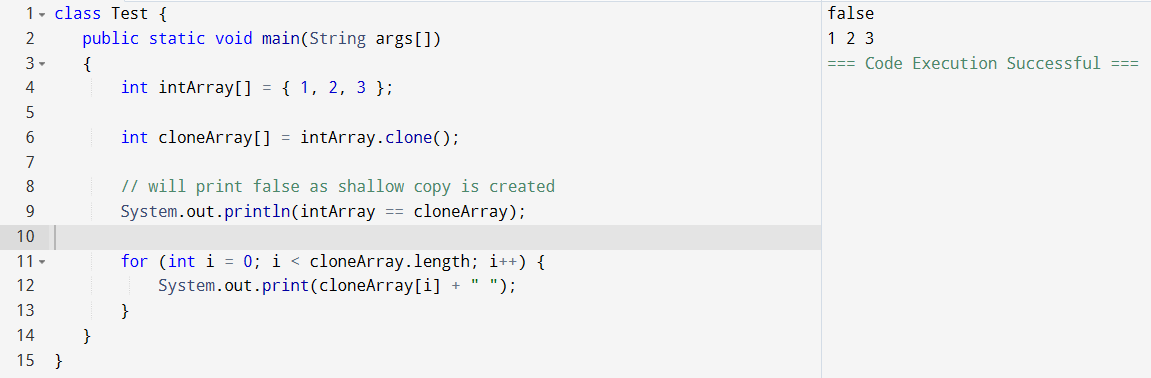


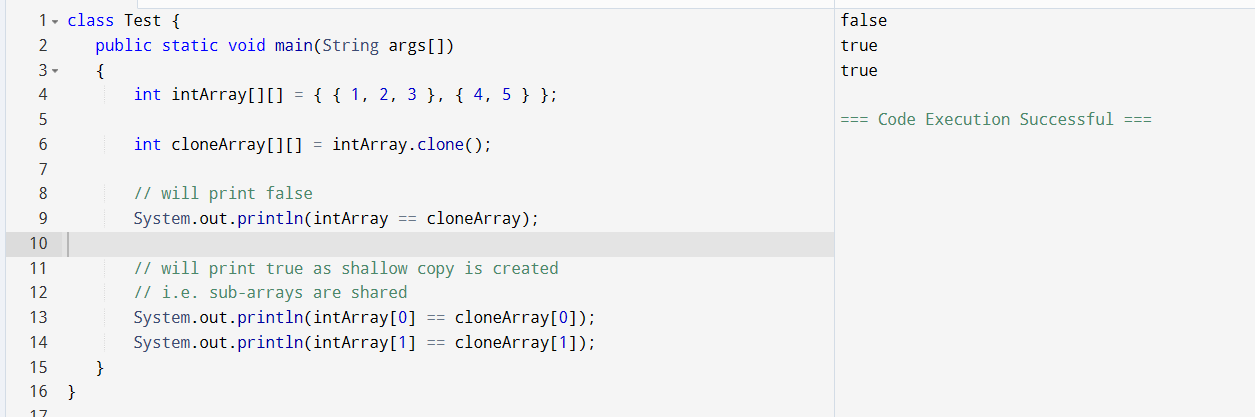
Task 27 - Home task

Task 28 - Home Task



Task 029 home Task

Task 030 Home Task



Task 031

class Calculation {

   int z;

   public void addition(int x, int y) {

      z = x + y;

      System.out.println("The sum of the given numbers:"+z);

   }

   public void Subtraction(int x, int y) {

      z = x - y;

      System.out.println("The difference between the given numbers:"+z);

   }

}

public class My\_Calculation extends Calculation {

   public void multiplication(int x, int y) {

      z = x \* y;

      System.out.println("The product of the given numbers:"+z);

   }

   public static void main(String args[]) {

      int a = 20, b = 10;

      My\_Calculation demo = new My\_Calculation();

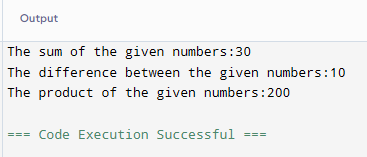
      demo.addition(a, b);

      demo.Subtraction(a, b);

      demo.multiplication(a, b);

   }

}



—--Task 032 ------------------------------------------------------------------------------

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.

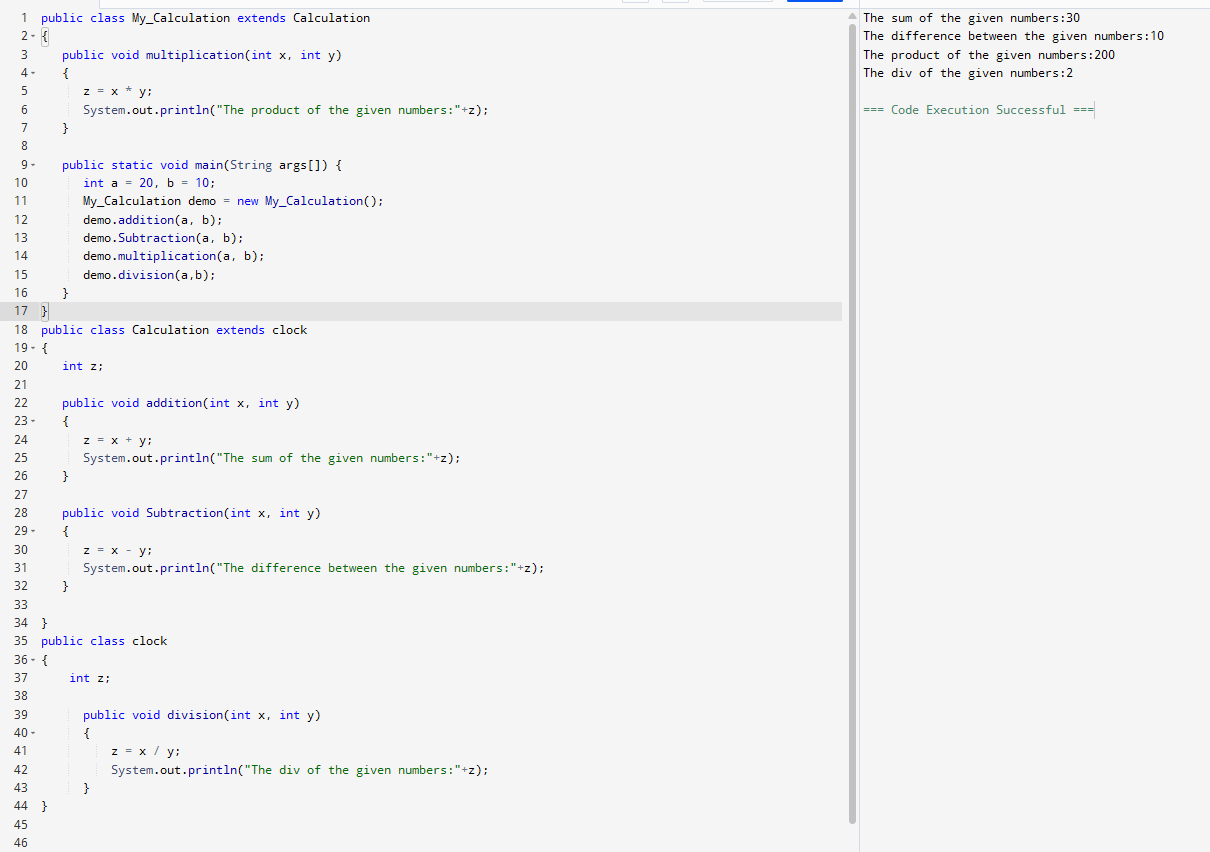
class clock {

—--

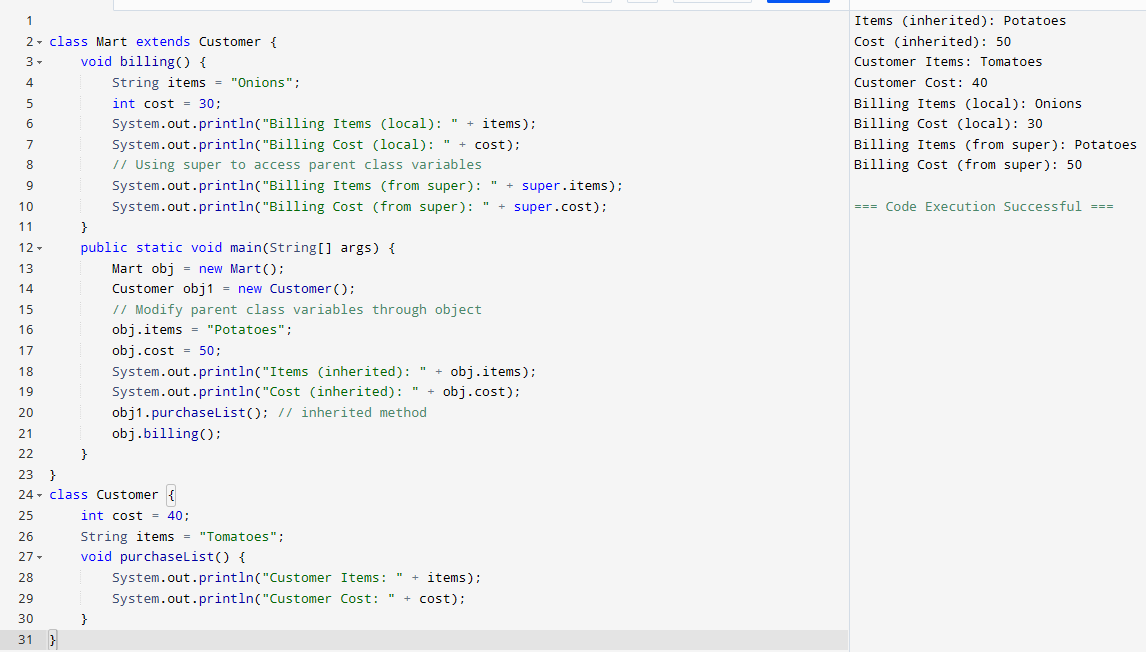
—--

}

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



Task 033

Ex:

Task 034

Void add(int x, int y){

Sout —> x and y values

}

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

Sout —-> x, y, z values

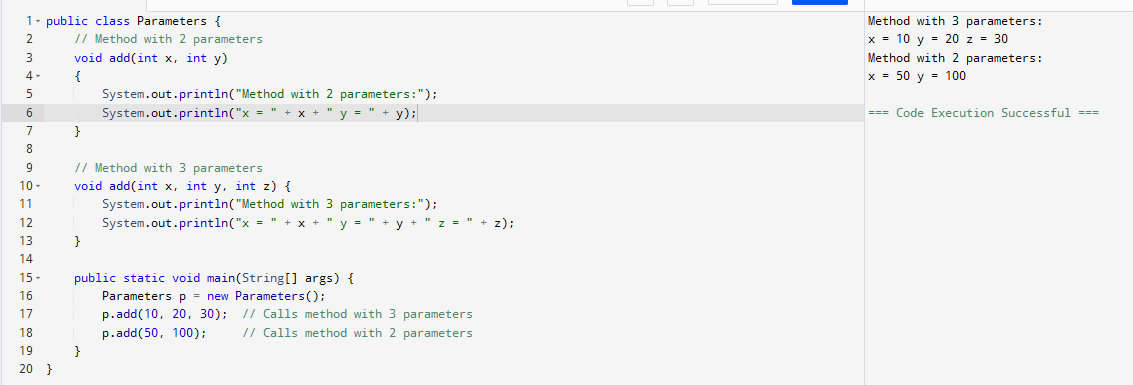
}

psvm(){

add(10,20,30);

add(50,100);

}



Type of parameters

Task 035

Void add(char x, char y){

Sout —-> x, y values

}

Void add(int x, int y) {

Sout —> x, y values

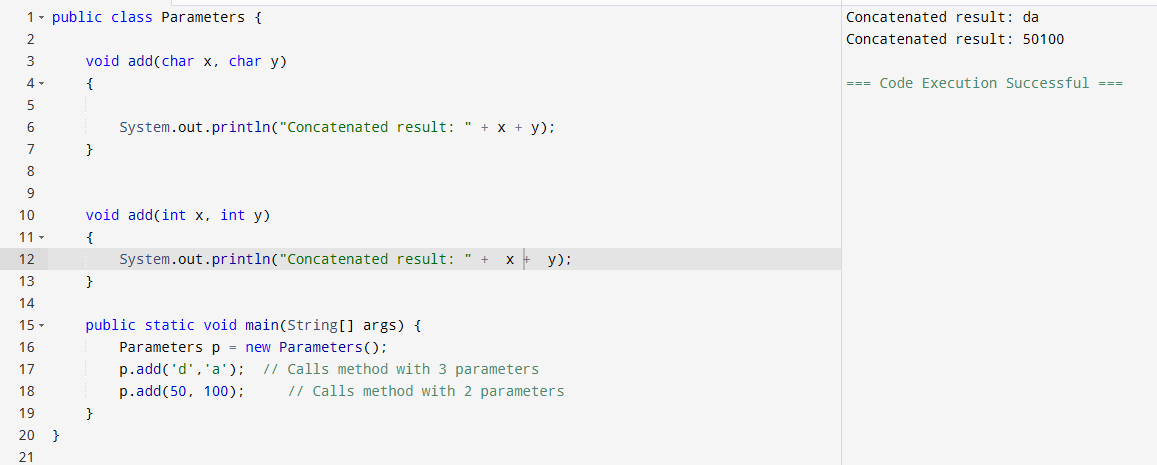
}

psvm(){

add(‘d’, ‘a’);

add(100, 100);

}



Task 036

Void add(int x, float y){

Sout → x, y values

}

Void add(float x, int y){

Sout  → x, y

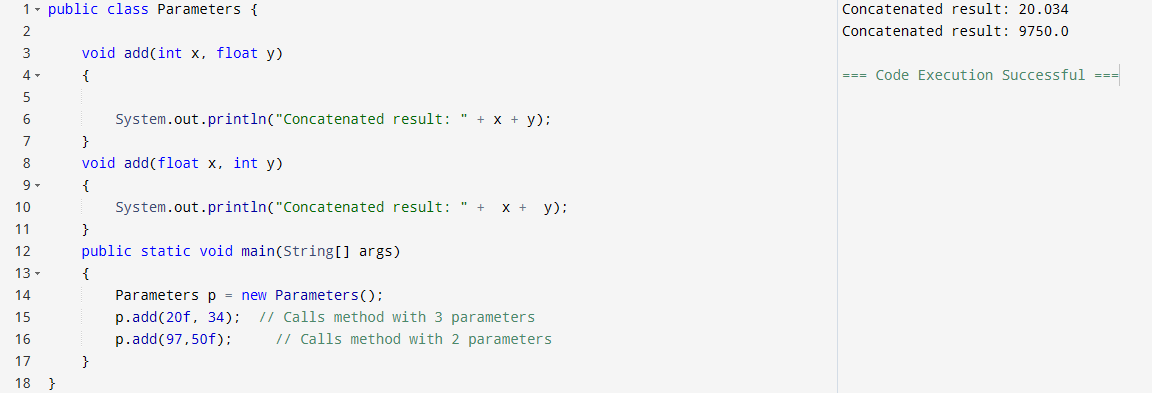
}

psvm(){

add(10.50f, 60);

add(100, 80.80f)

}



Task 037:

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; //===============>  ??????

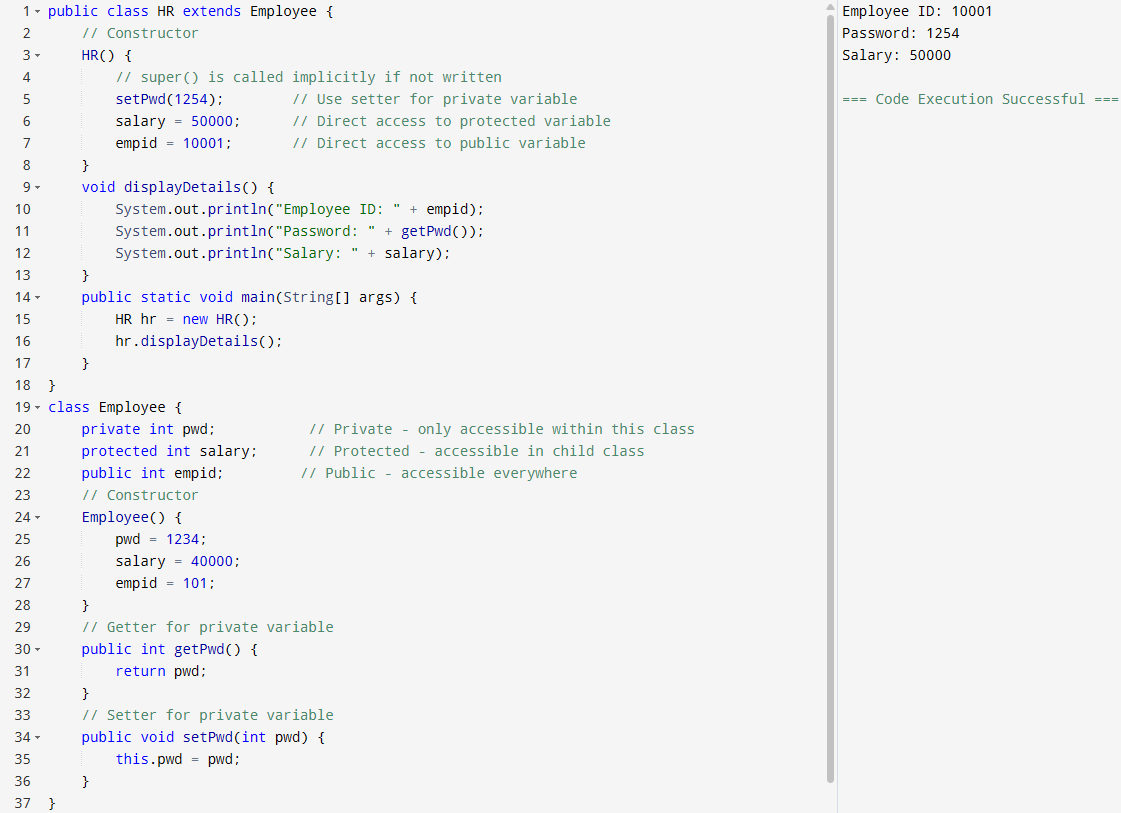
super.Salary = 50000; //==================>  ?

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;

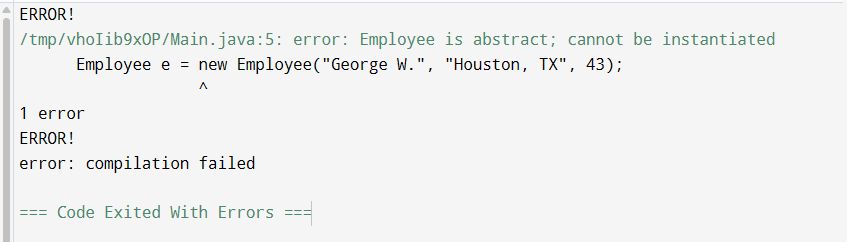
   }

   public int getNumber() {

      return number;

   }

}



Task 39

Above code without errors

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);

Salary s = new Salary("George W.", "Houston, TX", 43, 500000);

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

s.mailCheck();

System.out.println("Monthly Pay: " + s.computePay());

}

}

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;

}

}

class Salary extends Employee {

private double salary;

public Salary(String name, String address, int number, double salary) {

super(name, address, number);

this.salary = salary;

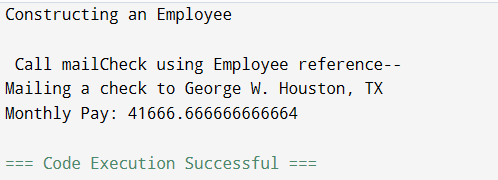
}

public double computePay() {

return salary/12;

}

}



Task 040:

