**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)

import java.util.HashMap;

import java.util.Map;

public class Task019 {

    public enum Element {

        H("Hydrogen", 1, 1.008f),

        HE("Helium", 2, 4.0026f),

        LI("Lithium", 3, 6.94f);

        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()) {

                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);

        }

    }

    public static void main(String[] args) {

        System.out.println("Element Details for all elemnents:");

        for (Element e : Element.values()) {

            System.out.println("Symbol: " + e.name() + ", Name: " + e.label +

                    ", Atomic Number: " + e.atomicNumber +

                    ", Atomic Weight: " + e.atomicWeight);

        }

}

}

Arrays

**Task 020:**

Create an array of your name

Hint : use

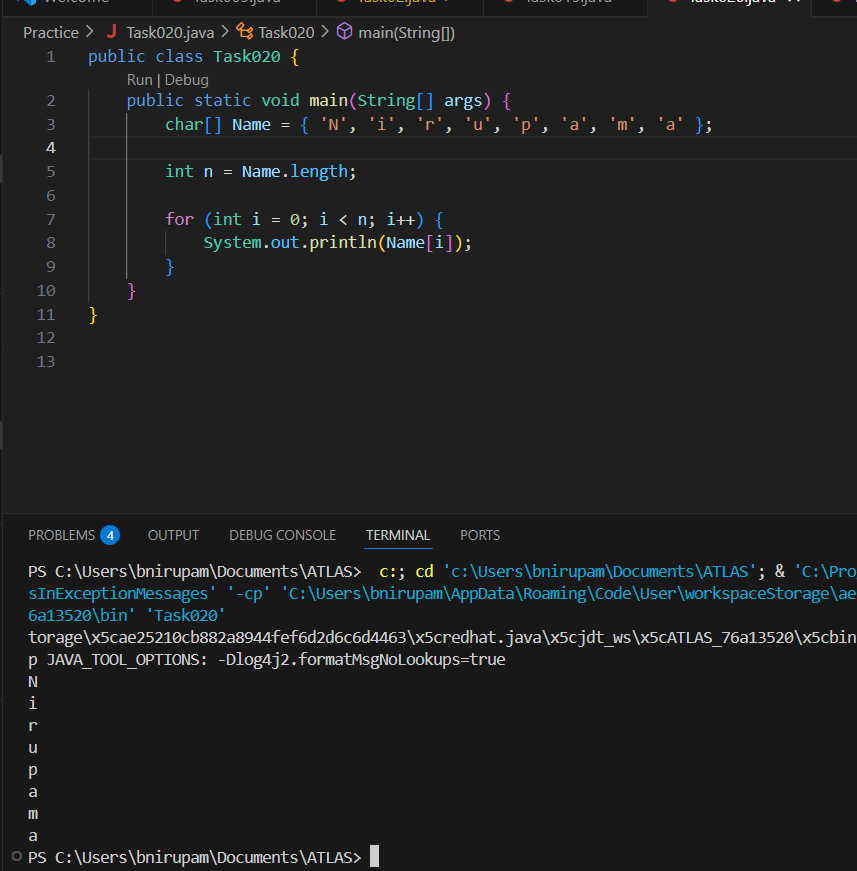
Char[] Name = {‘P’, “r’, ….}; // initializing an array

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..



Task 21: Home Task

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

public class Main {

   public static void main(String[] args)

   {

       // initializing array

       int[] arr = { 1, 2, 3, 4, 5 };

       // size of array

       int n = arr.length;

       // traversing array

       for (int i = 0; i < n; i++)

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

   }

}

**Output**



***Task 022 - home task***

**Implementation:**

// Java program to illustrate creating an array

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

// and prints each value to standard output.

​

class GFG {

   public static void main(String[] args)

   {

       // declares an Array of integers.

       int[] arr;

​

       // allocating memory for 5 integers.

       arr = new int[5];

​

       // initialize the elements of the array

       // first to last(fifth) element

      arr[0] = 10;

      arr[1] = 20;

       arr[2] = 30;

       arr[3] = 40;

       arr[4] = 50;

​

       // accessing the elements of the specified array

       for (int i = 0; i < arr.length; i++)

           System.out.println("Element at index "

                              + i + " : " + arr[i]);

   }

}

**Output**

Element at index 0 : 10

Element at index 1 : 20

Element at index 2 : 30

Element at index 3 : 40

Element at index 4 : 50

public class task22 {

  public static void main(String[] args){

       int[] arr = new int[5];

      arr[0] = 10;arr[1] = 20;

      arr[2] = 30;

      arr[3] = 40;

      arr[4] = 50;

       for (int i = 0; i < arr.length; i++)

           System.out.println("Element at index " + i + " = " + arr[i]);

   }

}

**Task 023 - home task**

**Example:** 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

​

class Student {

   public int roll\_no;

   public String name;

    Student(int roll\_no, String name){

       this.roll\_no = roll\_no;

       this.name = name;

   }

}

​

public class Main {

   public static void main(String[] args){

       // declares an Array of Student

       Student[] arr;

​

       // allocating memory for 5 objects of type Student.

       arr = new Student[5];

​

       // initialize the elements of the array

       arr[0] = new Student(1, "aman");

       arr[1] = new Student(2, "vaibhav");

       arr[2] = new Student(3, "shikar");

       arr[3] = new Student(4, "dharmesh");

       arr[4] = new Student(5, "mohit");

​

       // accessing the elements of the specified array

       for (int i = 0; i < arr.length; i++)

           System.out.println("Element at " + i + " : { "

                              + arr[i].roll\_no + " "

                              + arr[i].name+" }");

   }

}

**Output**

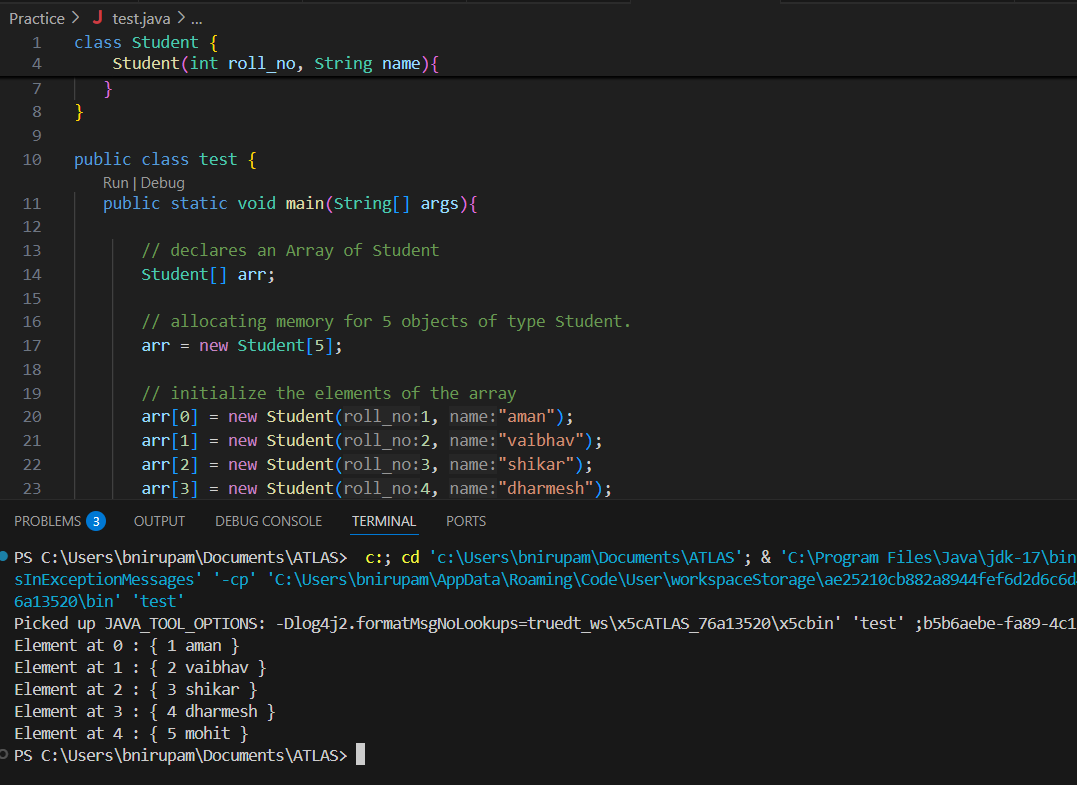
Element at 0 : { 1 aman }

Element at 1 : { 2 vaibhav }

Element at 2 : { 3 shikar }

Element at 3 : { 4 dharmesh }

Element at 4 : { 5 mohit }



**Task 024 Home task**

**Example:** An array of objects is also created like

// Java program to illustrate creating

//  an array of objects

 class Student{

   public String name;

    Student(String name){

       this.name = name;

   }

  @Override

   public String toString(){

       return name;

   }

}

 ​

public class Main{

   public static void main (String[] args){

       // declares an Array and initializing the

      // elements of the array

       Student[] myStudents = new Student[]{

         new Student("Dharma"),new Student("sanvi"),

         new Student("Rupa"),new Student("Ajay")

       };

        // accessing the elements of the specified array

       for(Student m:myStudents){

           System.out.println(m);

       }

   }

}

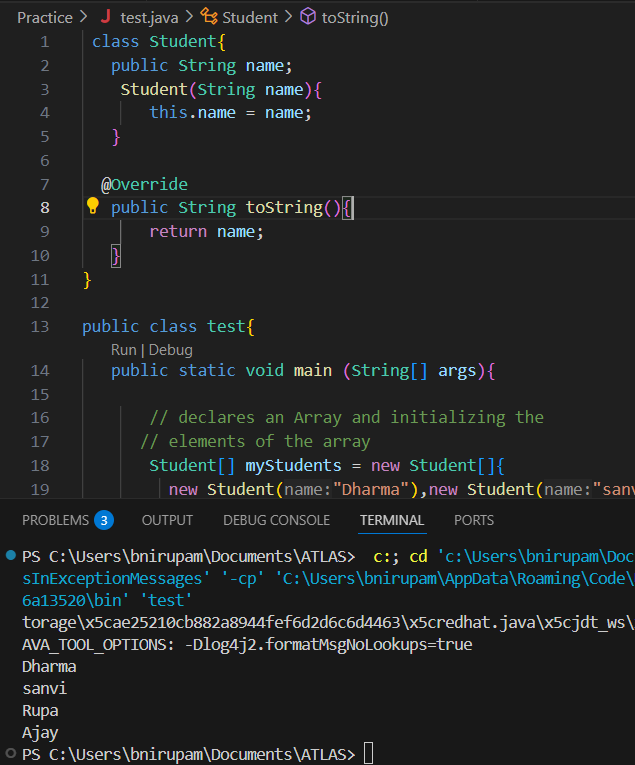
**Output**

Dharma

sanvi

Rupa

Ajay



**Task 025 - home Task**

// Code for showing error "ArrayIndexOutOfBoundsException"

​

public class GFG {

   public static void main(String[] args)

   {

       int[] arr = new int[4];

       arr[0] = 10;

       arr[1] = 20;

       arr[2] = 30;

       arr[3] = 40;

​

       System.out.println(

           "Trying to access element outside the size of array");

       System.out.println(arr[5]);

   }

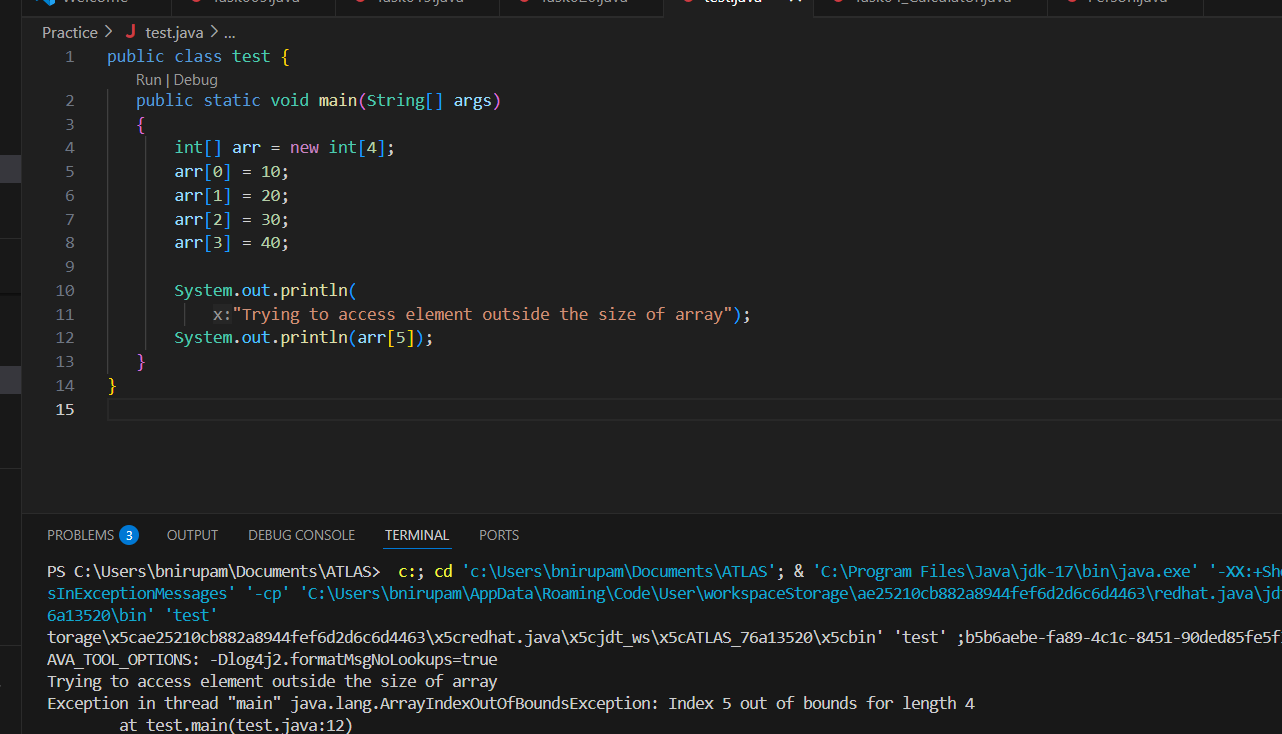
}

**Output**

Trying to access element outside the size of array

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 4

at GFG.main(GFG.java:13)



Task 025  - home task

**Example:** Let us start with basic two dimensional Array declared and initialized.

// Java Program to demonstrate

// Multidimensional Array

import java.io.\*;

​

class GFG {

   public static void main(String[] args){

       // Two Dimensional Array

      // Declared and Initialized

      int[][] arr = new int[3][3];

​

       // Number of Rows

       System.out.println("Rows : " + arr.length);

       // Number of Columns

       System.out.println("Columns : " + arr[0].length);

   }

}

**Output**

Rows:3

Columns:3



**Task 026 - Home Task**

**Example:** 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

public class multiDimensional {

     // main function

   public static void main(String args[])

   {

       // declaring and initializing 2D array

       int arr[][] = { { 2, 7, 9 }, { 3, 6, 1 }, { 7, 4, 2 } };

​

       // printing 2D array

       for (int i = 0; i < 3; i++) {

           for (int j = 0; j < 3; j++)

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

​

           System.out.println();

       }

   }

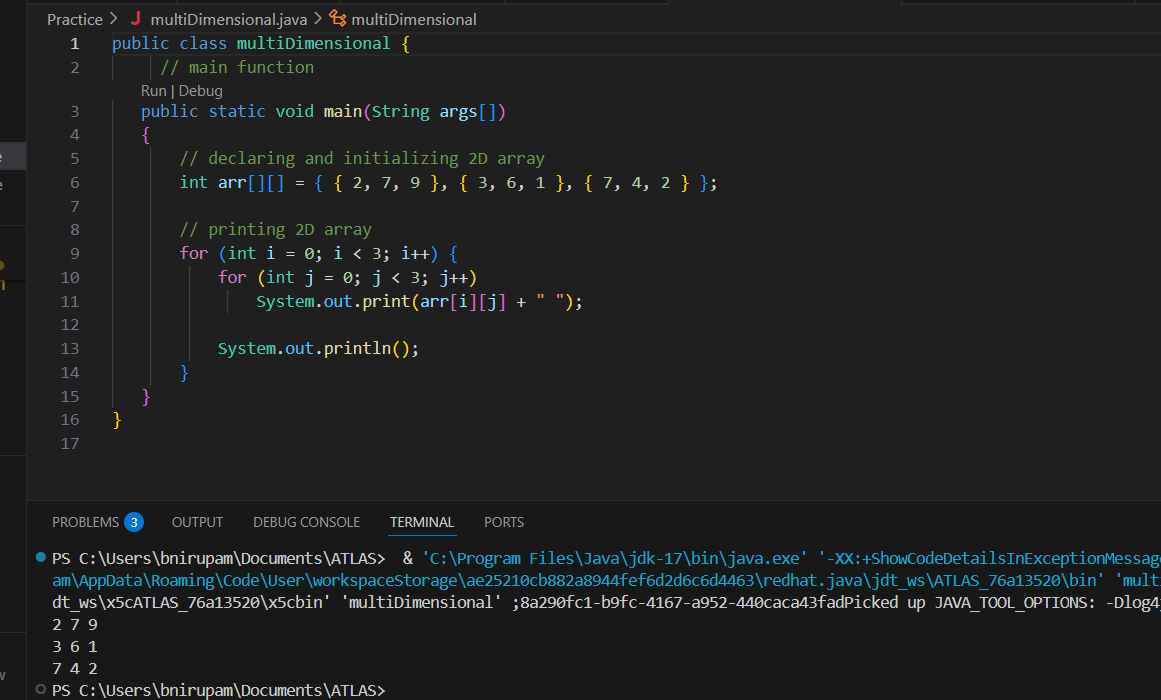
}

**Output**

2 7 9

3 6 1

7 4 2



Task 27 - Home task

// Java program to demonstrate

// passing of array to method

​

public class Test {

   // Driver method

   public static void main(String args[])

   {

       int arr[] = { 3, 1, 2, 5, 4 };

​

       // passing array to method m1

       sum(arr);

   }

​

   public static void sum(int[] arr)

   {

       // getting sum of array values

       int sum = 0;

​

       for (int i = 0; i < arr.length; i++)

           sum += arr[i];

​

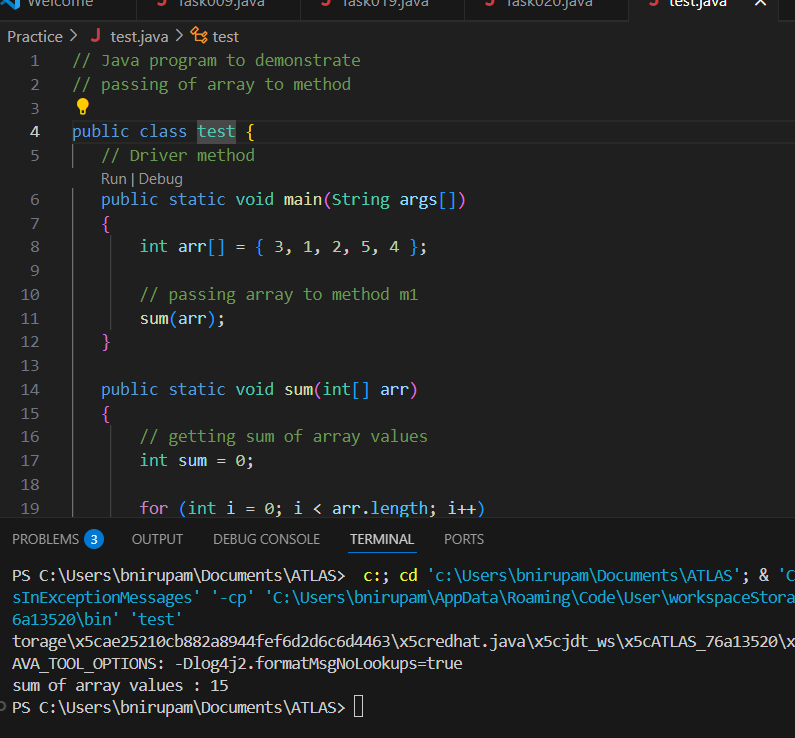
       System.out.println("sum of array values : " + sum);

   }

}

**Output**

sum of array values : 15



**Task 28 - Home Task**

// Java program to demonstrate

// return of array from method

​

class Test {

   // Driver method

   public static void main(String args[])

   {

       int arr[] = m1();

       for (int i = 0; i < arr.length; i++)

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

   }

   public static int[] m1()

   {

       // returning  array

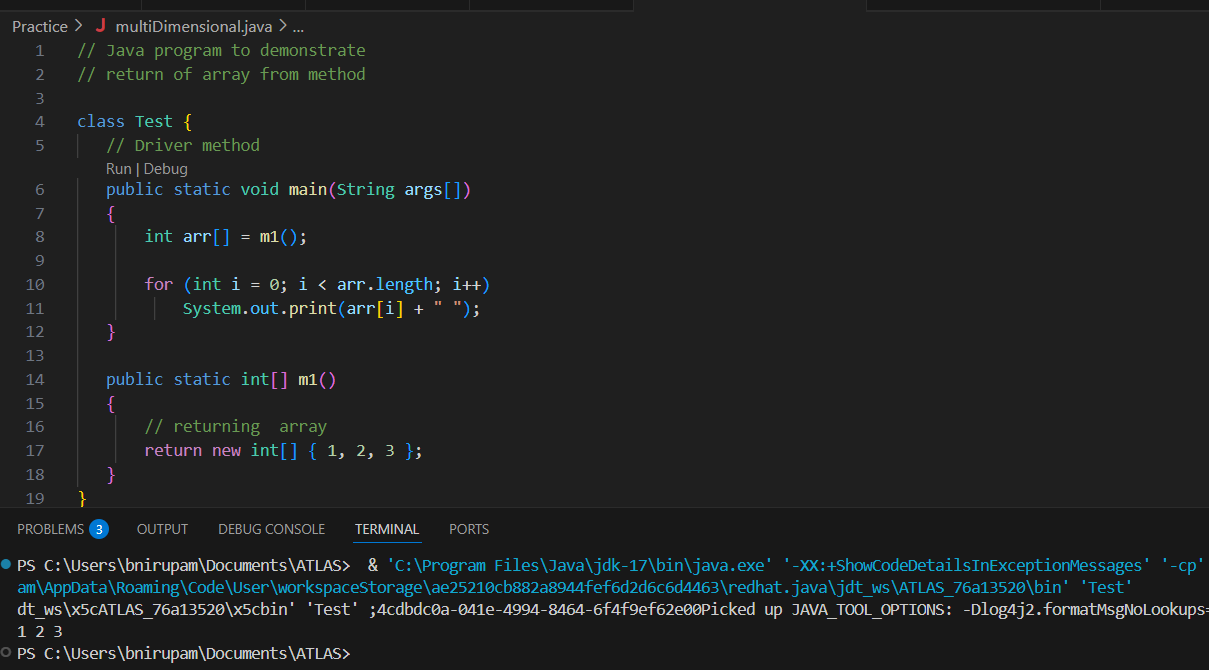
       return new int[] { 1, 2, 3 };

   }

}

**Output**

1 2 3



Task 029 home Task

// Java program to demonstrate

// cloning of one-dimensional arrays

​

class Test {

   public static void main(String args[])

   {

       int intArray[] = { 1, 2, 3 };

​

       int cloneArray[] = intArray.clone();

​

       // will print false as shallow copy is created

       System.out.println(intArray == cloneArray);

​

       for (int i = 0; i < cloneArray.length; i++) {

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

       }

   }

}

**Output**

false

1 2 3



**Below is the implementation of the above method:**

**Task 030** Home Task

// Java program to demonstrate

// cloning of multi-dimensional arrays

​

class Test {

   public static void main(String args[])

   {

       int intArray[][] = { { 1, 2, 3 }, { 4, 5 } };

​

       int cloneArray[][] = intArray.clone();

​

       // will print false

       System.out.println(intArray == cloneArray);

​

       // will print true as shallow copy is created

       // i.e. sub-arrays are shared

       System.out.println(intArray[0] == cloneArray[0]);

       System.out.println(intArray[1] == cloneArray[1]);

   }

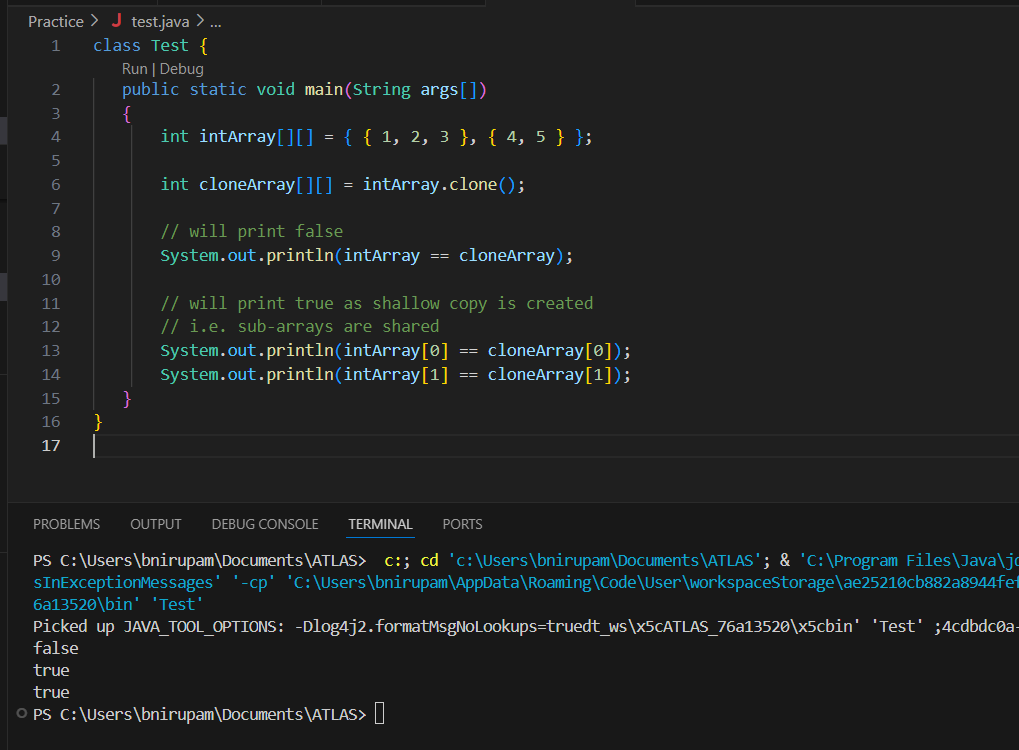
}

**Output**

false

true

true



**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);

   }

}

public class My\_Calculation2 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);

   }

}

Inheritance provides - reusability

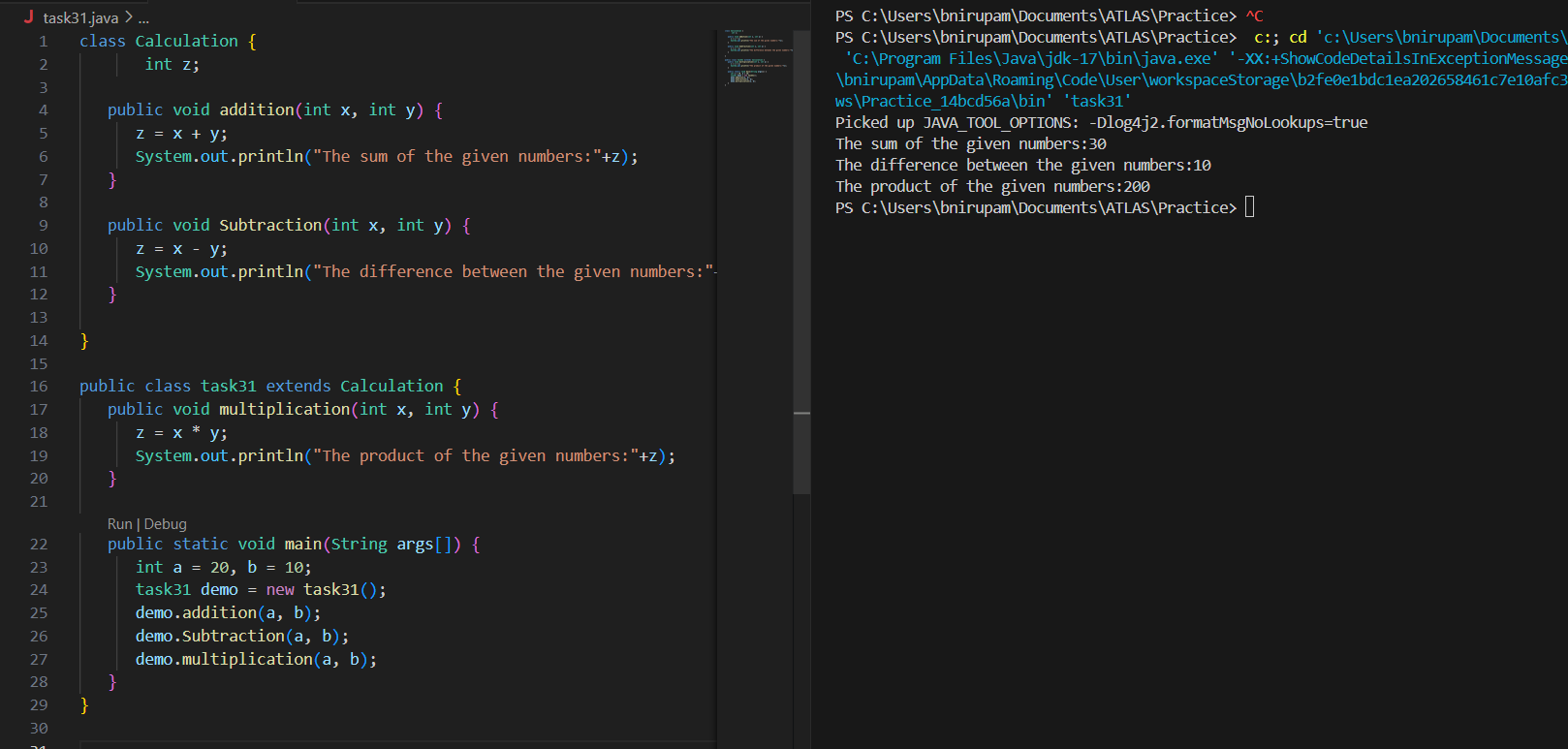
It avoids - duplication

Multi level inheritance

Clac < ========= My\_calculation < ======= calculation

Class calc extends My\_calculation{

}



—--**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 {

—--

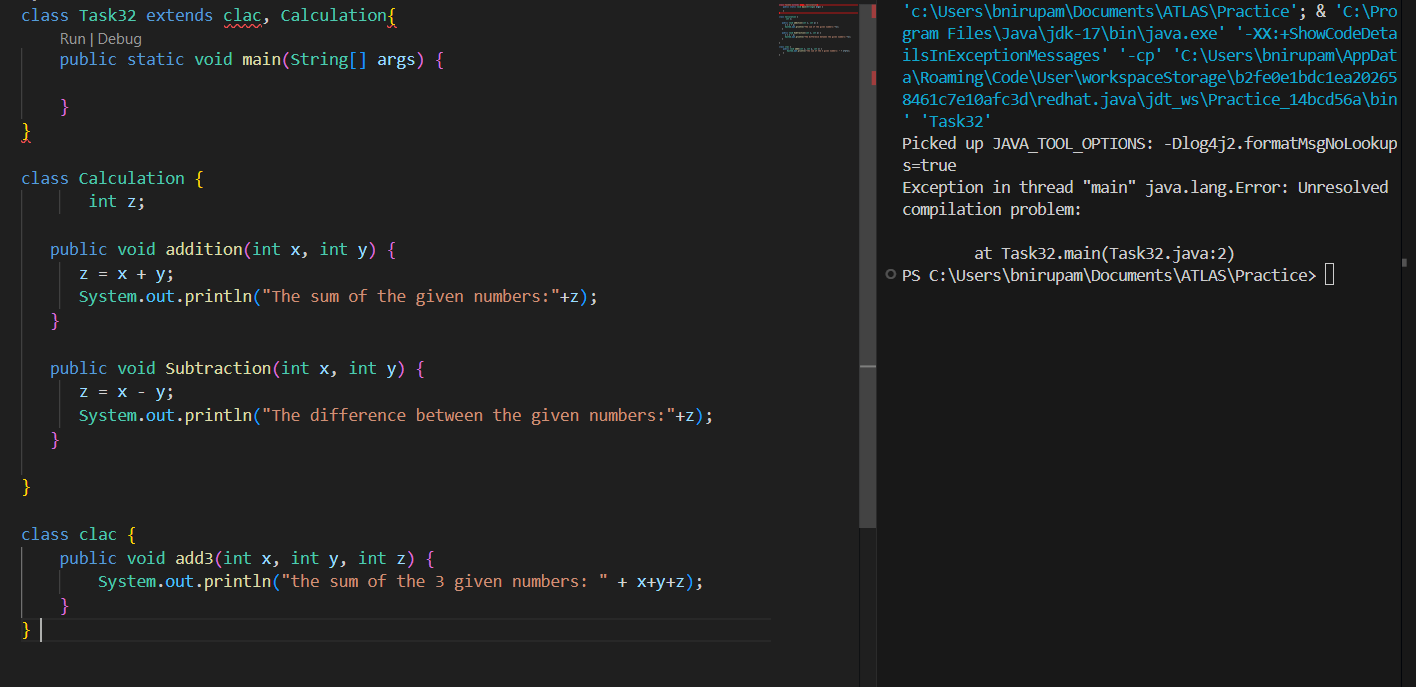
—--

}

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

// —---------------------------------- ???????????????????????

}



**Task 033**

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);

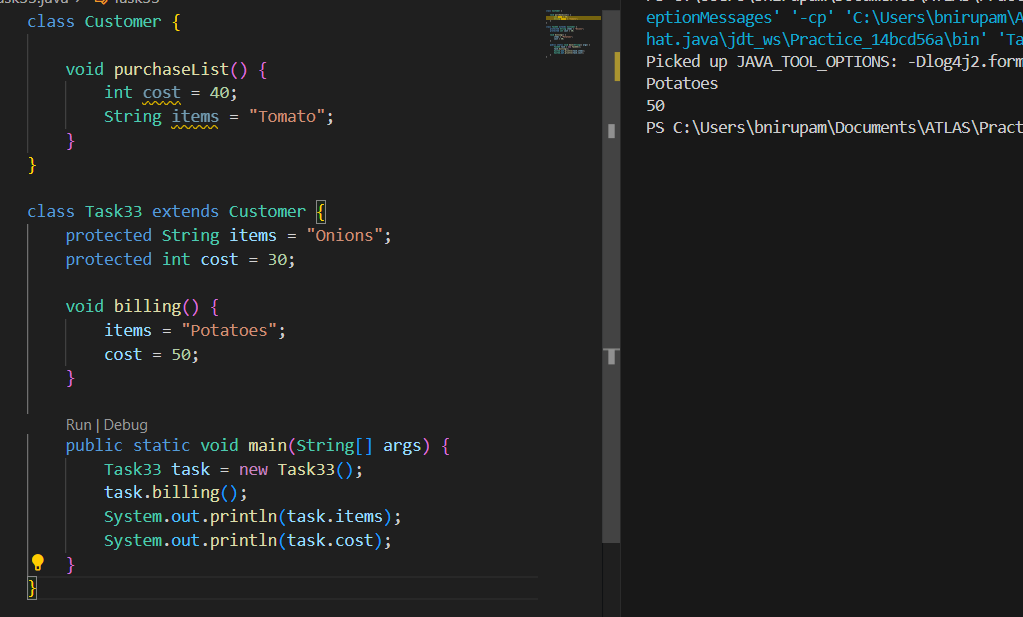
sout”(%%%%%%%%%%%%%%”);

Sout(super.items);

sout(suer.cost);

}

}



**Task 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);

}

Sequence of Parameters

**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)

}

class Calculator {

    int add(int a, int b) {

        return a + b;

    }

    int add(int a, int b, int c) {

        return a + b + c;

    }

    double add(double a, double b) {

        return a + b;

    }

    char add (char a, char b) {

        return (char)(a + b);

    }

}

public class task34to36 {

    public static void main(String[] args) {

        Calculator calc = new Calculator();

        System.out.println(calc.add(2, 3));

        System.out.println(calc.add(2, 3, 4));

        System.out.println(calc.add(2.5, 3.5));

    }

}

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; //===============>  ??????  
The pwd field in the Employee class is declared as private. This means it can only be accessed from within the Employee class itself. Even a subclass like Hr cannot directly see or modify a private member of its parent.

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

The Salary field in the Employee class is protected. protected members are accessible within the class itself, by any class in the same package, and most importantly, by subclasses (like Hr), regardless of their package.

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

The empid field is public. public members are the most permissive; they can be accessed from anywhere.

psvm(){

}

} class Employee {

    private int password = 1234;

    protected int salary = 4567;

    public int empId = 5864;

    private String name;

    public Employee(String name) {

        this.name = name;

    }

    public int getPassword() {

        return password;

    }

    public String getName() {

        return name;

    }

}

public class Task037 extends Employee {

    public Task037() {

        super("Default");

    }

    public static void main(String[] args) {

        Employee e = new Employee("Nirupama");

        System.out.println(e.getPassword());

        System.out.println(e.salary);

        System.out.println(e.empId);

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

    }

}

**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;

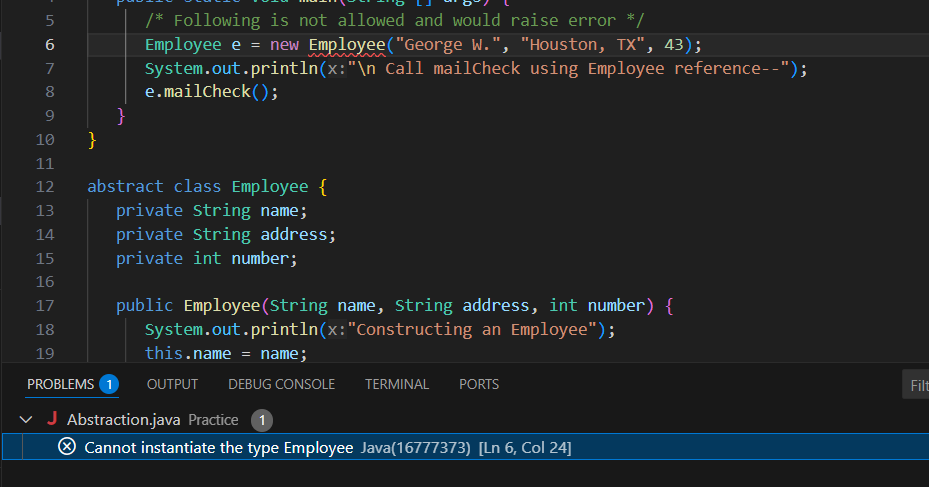
   }

   public int getNumber() {

      return number;

   }

}



**Task 039**

Rewrite the above code to give the output without errors..

public class Abstraction {

    public static void main(String[] args) {

        Employee e = new SalariedEmployee("George W.", "Houston, TX", 43, 72000.0);

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

        e.mailCheck();

        System.out.println("Monthly Pay: " + e.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 abstract double computePay();

    public void mailCheck() {

        System.out.println("Mailing a check to " + name + " at " + 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 SalariedEmployee extends Employee {

    private double annualSalary;

    public SalariedEmployee(String name, String address, int number, double annualSalary) {

        super(name, address, number);

        this.annualSalary = annualSalary;

    }

    @Override

    public double computePay() {

        return annualSalary / 12;

    }

}

**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.");

    }

}

// Main class to demonstrate abstraction

public class Main {

    public static void main(String[] args) {

        Gadgets remote = new TVRemote();

        remote.turnOn();

        remote.turnOff();

    }

}

abstract class Gadgets {

    abstract void turnOn();

    abstract void turnOff();

}

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.");

    }

}

class FanRemote extends Gadgets {

    @Override

    void turnOn() {

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

    }

    @Override

    void turnOff() {

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

    }

}

class CoolerRemote extends Gadgets {

    @Override

    void turnOn() {

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

    }

    @Override

    void turnOff() {

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

    }

}

public class Task040 {

    public static void main(String[] args) {

        Gadgets remote;

        remote = new TVRemote();

        remote.turnOn();

        remote.turnOff();

        remote = new ACRemote();

        remote.turnOn();

        remote.turnOff();

        remote = new FanRemote();

        remote.turnOn();

        remote.turnOff();

        remote = new CoolerRemote();

        remote.turnOn();

        remote.turnOff();

    }

}