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;

// First file: Element.java

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()) {

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);

}

}

// Second file: Main.java

public class Main {

public static void main(String[] args) {

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

System.out.println("Name: " + e.label +

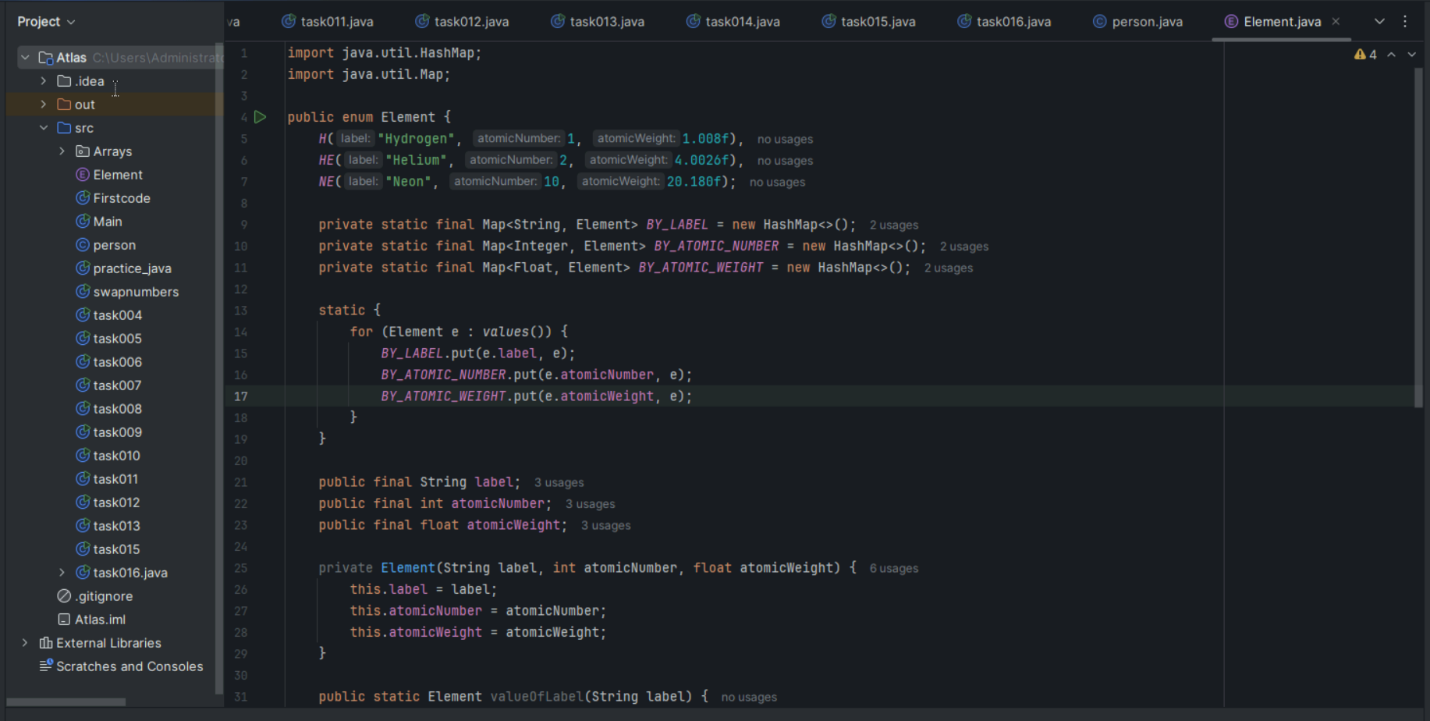
", Atomic Number: " + e.atomicNumber +

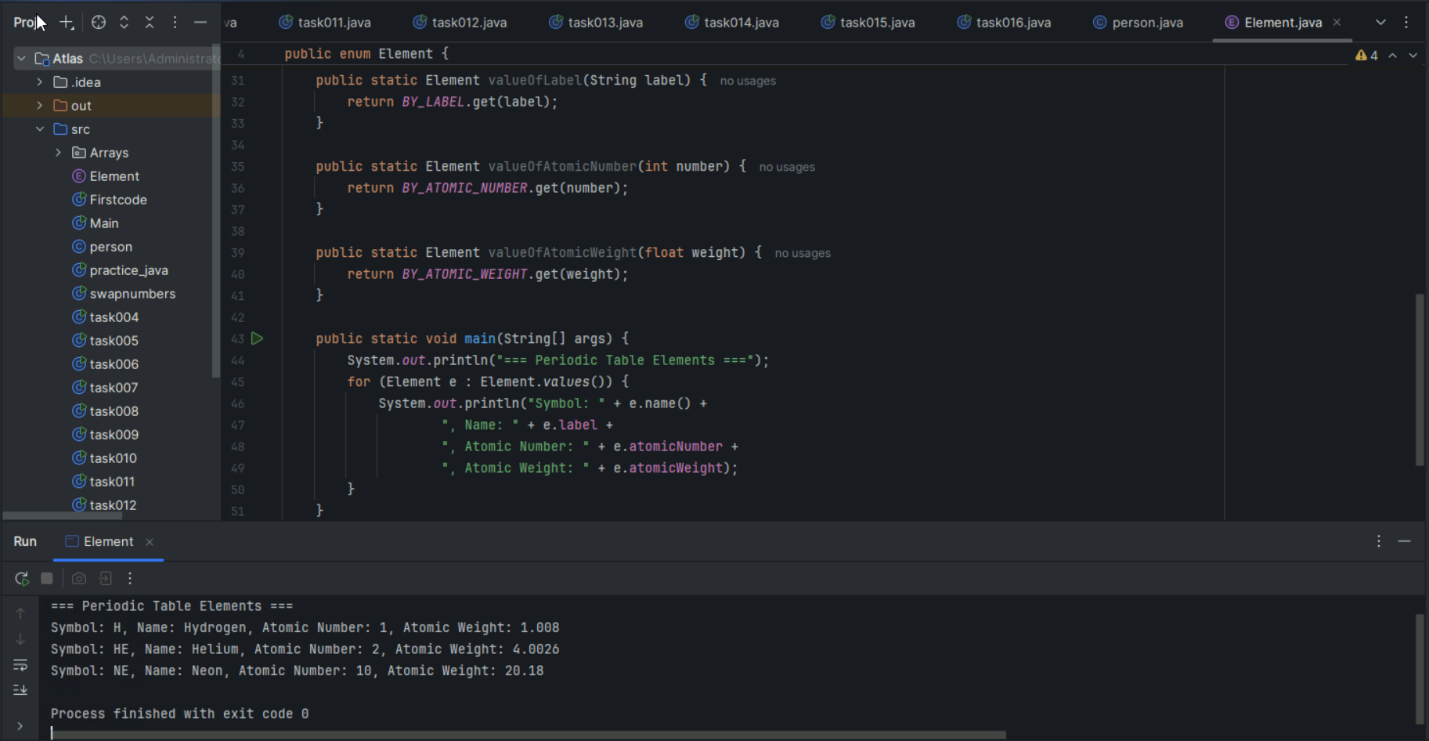
", Atomic Weight: " + e.atomicWeight);

}

}

}





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);

  }

}

—----------------------------------what is the reason for the error —---------------explain

the error in Task017.java:

Original problematic code:

public class Task017 {

public static void main(String[] args) {

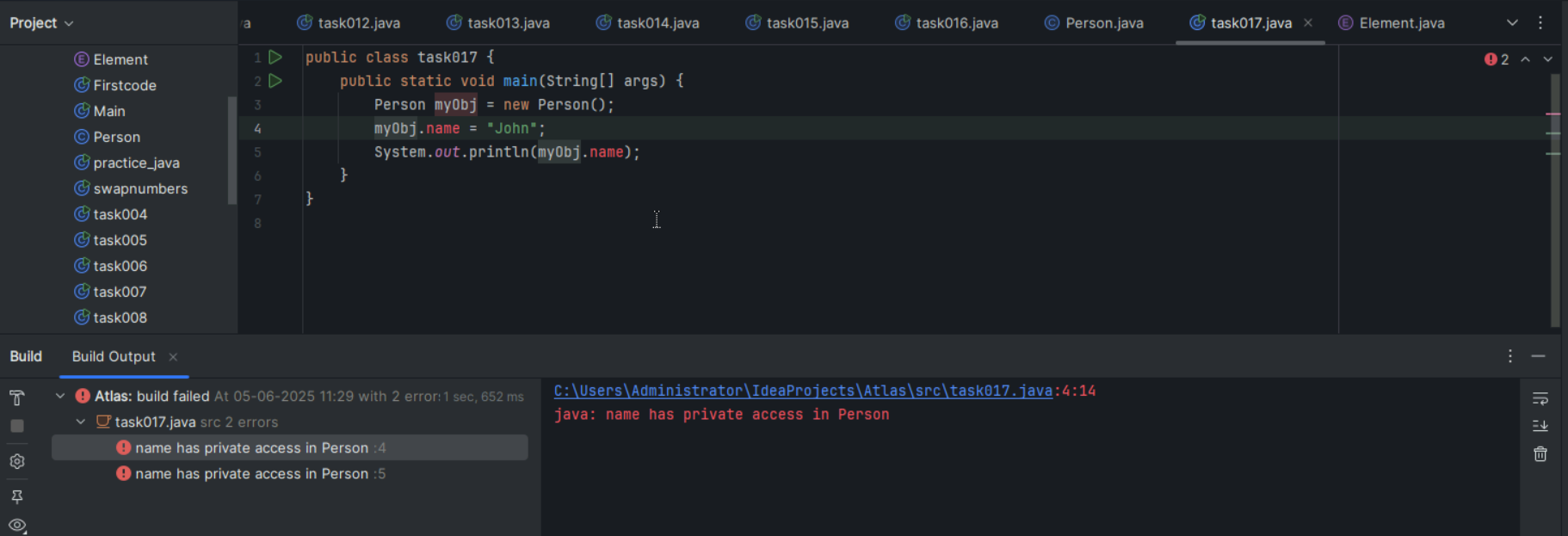
Person myObj = new Person();

myObj.name = "John"; // This line causes the error

System.out.println(myObj.name); // This line also causes the error

}

}



The error occurs because:

The variable 'name' in Person class is declared as private

Private members cannot be accessed directly from outside the class

We must use getter and setter methods to access private members

Correct code:

public class Task017 {

public static void main(String[] args) {

Person myObj = new Person();

myObj.setName("John"); // Use setter method

System.out.println(myObj.getName()); // Use getter method

}

}

This demonstrates:

Encapsulation - hiding data from direct access

Private variables can only be accessed through public methods

Getter and setter methods provide controlled access to private variables

The error message would be something like:

name has private access in Person

Task 018

Now create one more program named Task018.java

public class Main {

  public static void main(String[] args) {

    Person myObj = new Person();

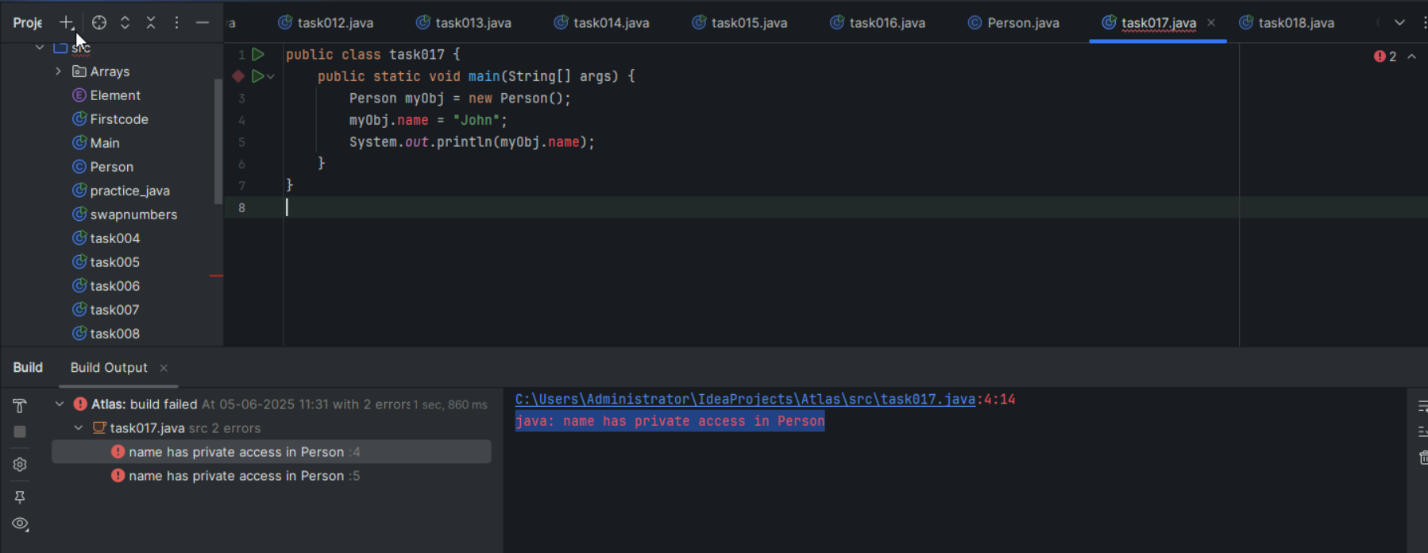
    myObj.setName("John");

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

  }

}

Now —--------------think what is the output of the above code—--------------





If a variable is private, always access it through getter and setter.

We cannot use myObj.name directly from another class.

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”);

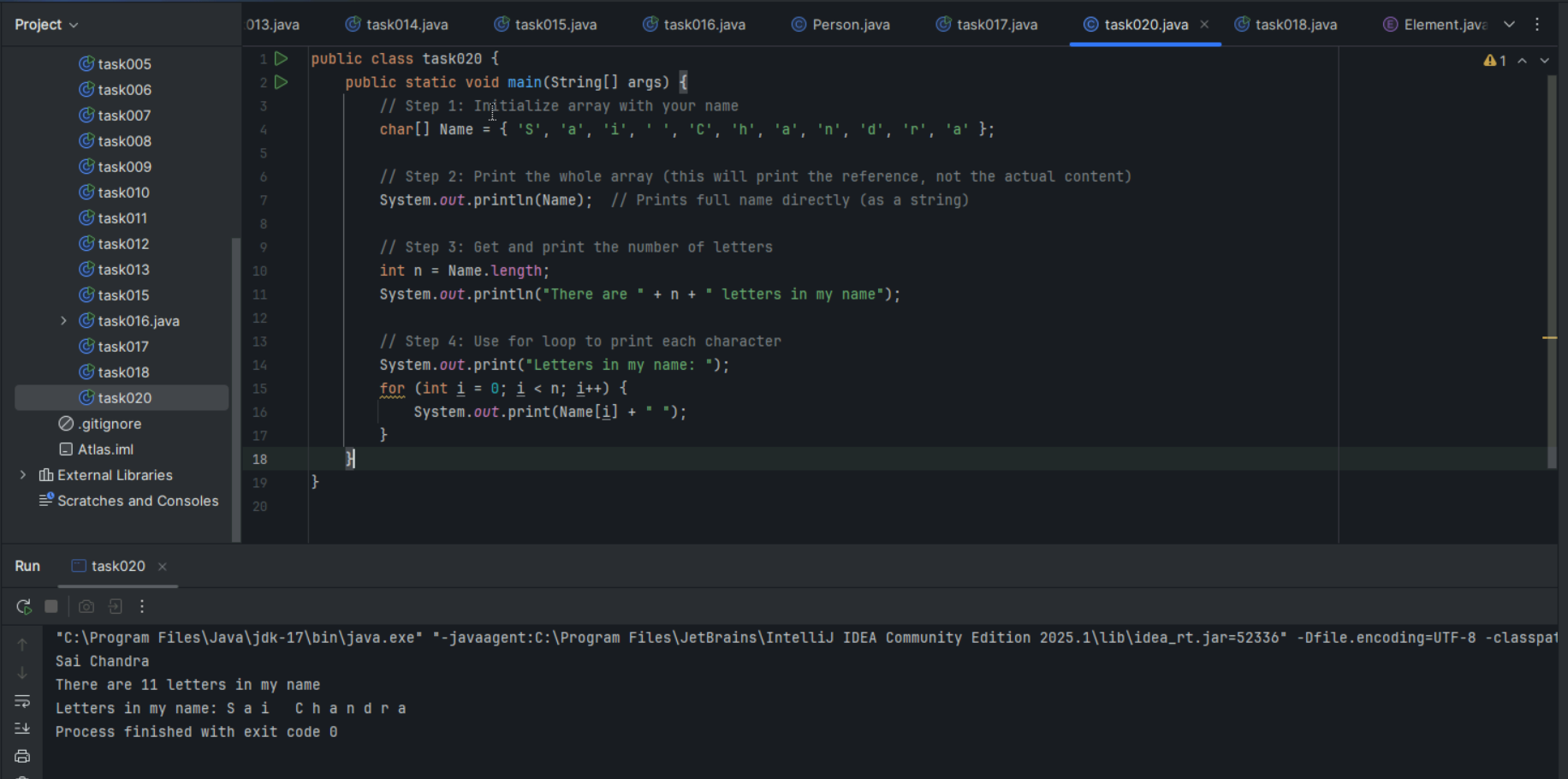
Use for loop to display each letter..

HInt: use ghe below code snippet…

// traversing array

        for (int i = 0; i < n; i++)

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



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();

        intArray[1]= 4;

        // 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] + " ");

        }

        System.out.println("Original Array");

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

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

        }

    }

 }

+++++++++++++++++++++++++++++++++++++++++=============+++++++++++++++++

**Task 21 to Task 30   – home tasks — plz refer Doc 17 Arrays in java..**

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

=========================================+++++++++++++++++++++++++++++++

**OOPS (Object Oriented Programming)** ++++++++++++++++++++++++++++++++++++++++++==============================

Inheritance, Polymorphism, Encapsulation, Abstraction, Interfaces, Exception Handling,

4 pillers of OOPS

Inheritance, Polymorphism, Encapsulation, Abstraction

Inheritance : getting parental characteristics

Types of inheritance

Single inheritance

Multi level inheritance

hYBRID IN

Hyrarical inhe

Multiple Inheritance — not supported in java

Due to diamond problem

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);

   }

}

Inheritance provides - reusability

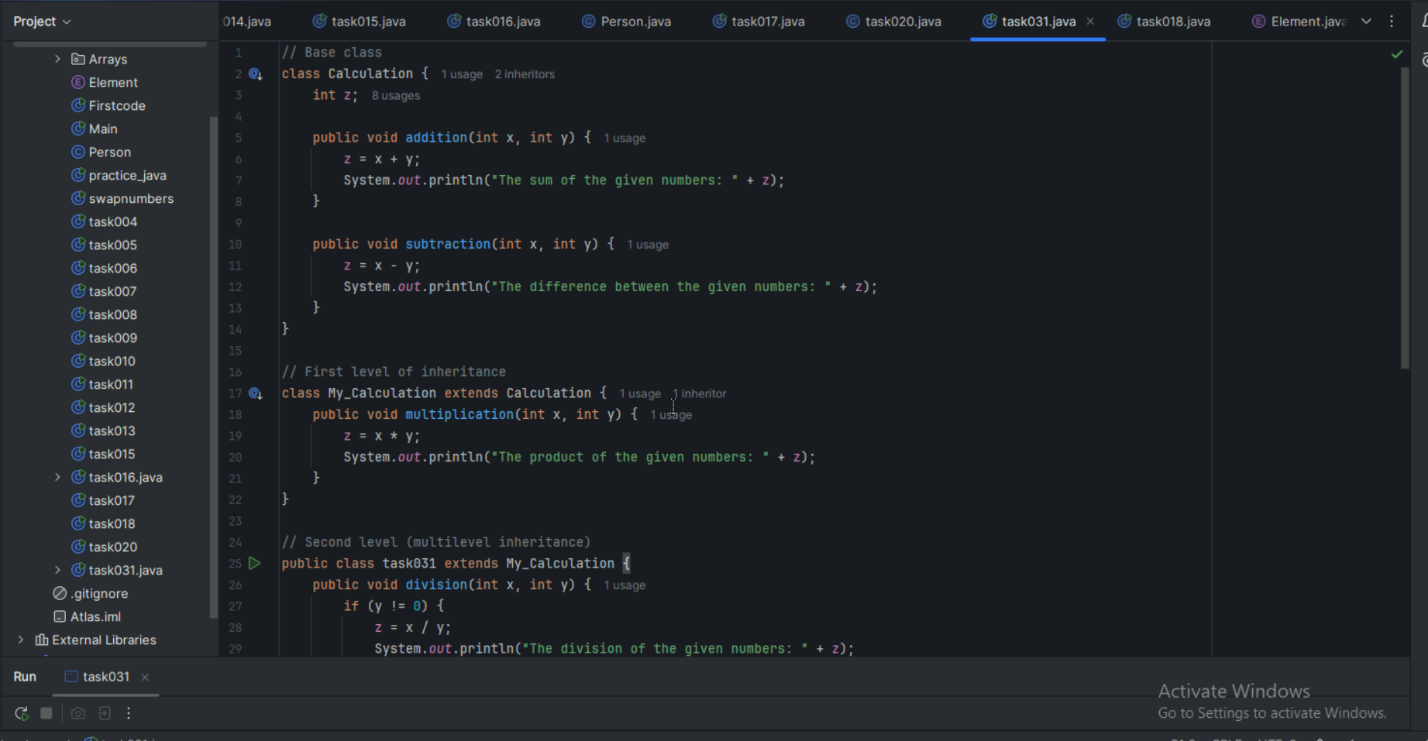
It avoids - duplication

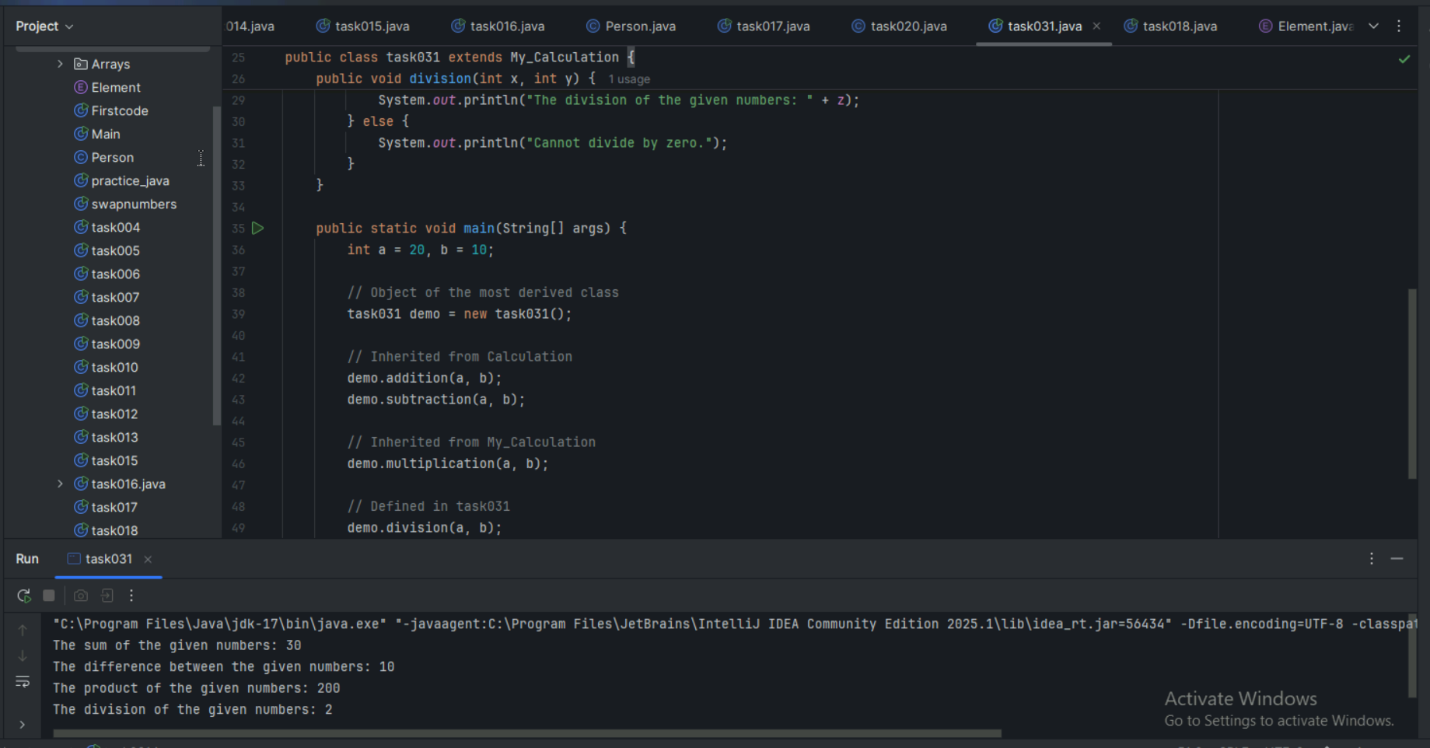
Multi level inheritance

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);

}

}

—----------------------------------------------------------------------------------------------------------------------------

Polymorphism –  Method overloading

-------------------------------------------------------------------------------------------------------------------------------

Two or more methods having the same name but differ -

No of parameters

Type of parameters

Sequence of parameters

No of parameters:

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

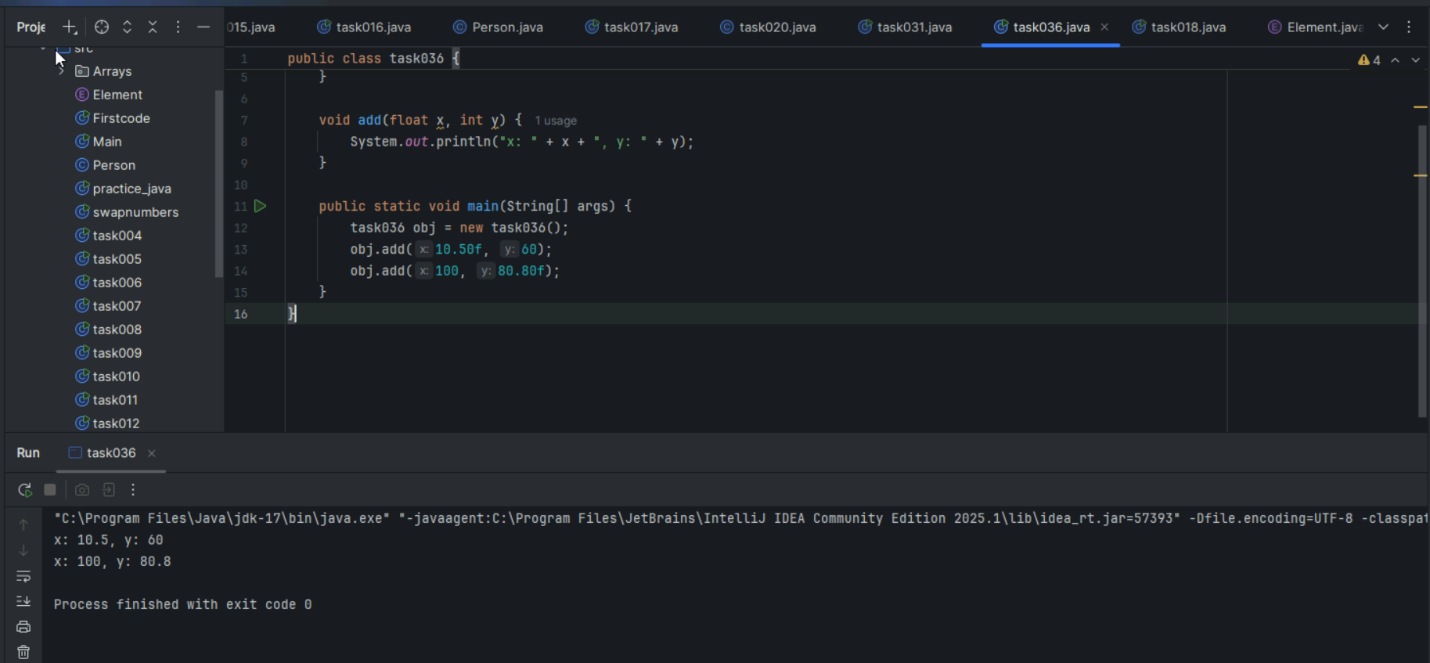
}

psvm(){

add(10.50f, 60);

add(100, 80.80f)

}



========================================================================

Encapsulation

========================================================================

Data hiding – secured data

Access modifiers 👍

private

Protected

Public

Task 037:

Class Employee{

Private int pwd;

Protected int Salary;

Public int empid:

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

}

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

}

}

Class Hr extends Employee {

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

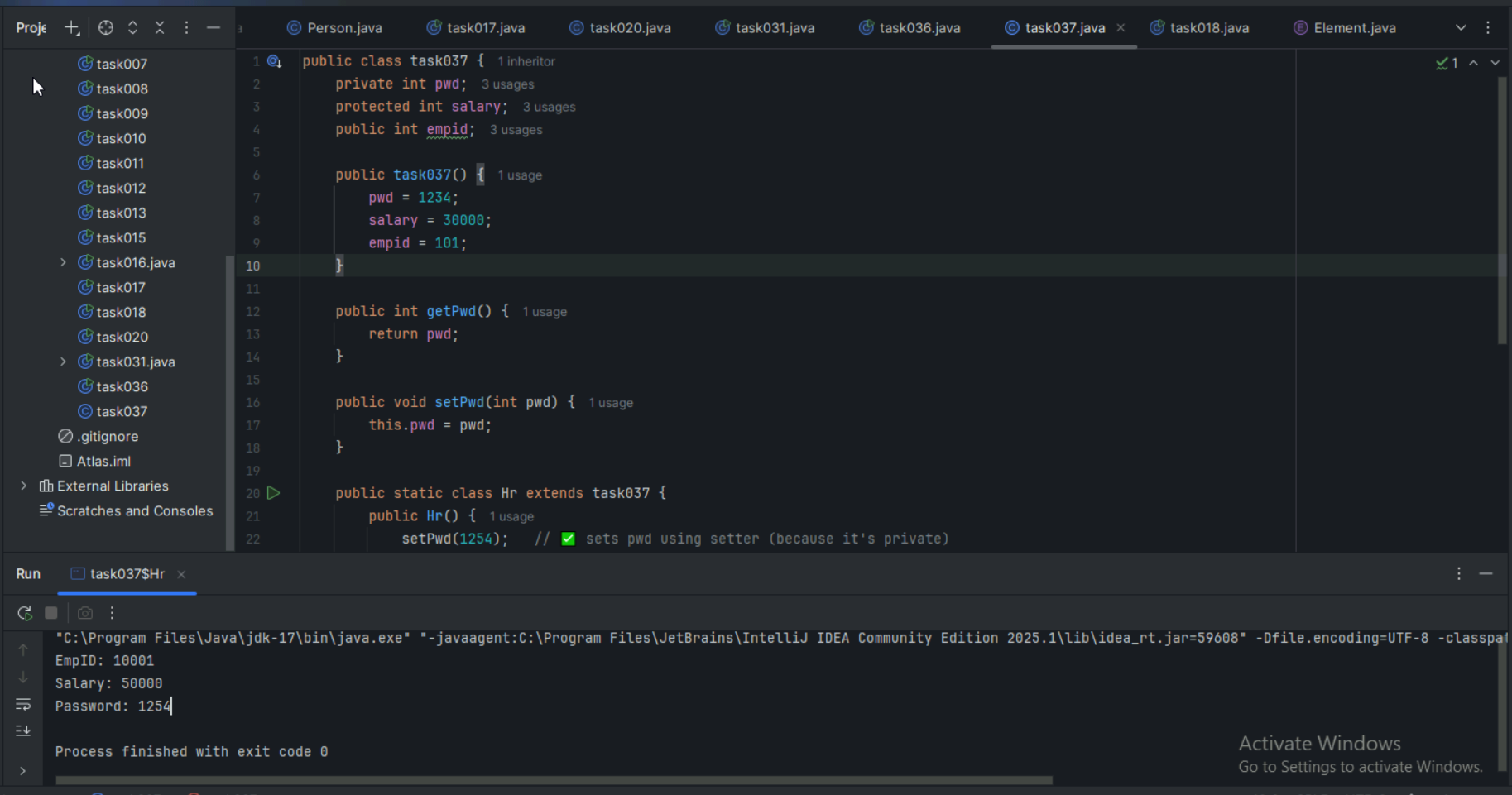
super.Salary = 50000; //==================>  ?

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

psvm(){

}

}



========================================================================

Abstraction ========================================================================

Java Abstract Classes

A Java class which contains the abstract keyword in its declaration is known as abstract class.

Java abstract classes may or may not contain abstract methods, i.e., methods without body ( public void get(); )

But, if a class has at least one abstract method, then the class must be declared abstract.

If a class is declared abstract, it cannot be instantiated.

To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.

If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

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;

   }

}

