PRACTICE PROBLEMS

PASUPULETI VINAY KUMAR

Writing a program in Java to implement implicit and explicit type casting

CODE:

**package** program1;

**public** **class** Practice\_1 {

**public** **static** **void** main(String[] args) {

System.***out***.println("Implicit Type Casting");

**char** a='A';

System.***out***.println("Value of a: "+a);

**int** b=a;

System.***out***.println("Value of b: "+b);

**float** c=a;

System.***out***.println("Value of c: "+c);

**long** d=a;

System.***out***.println("Value of d: "+d);

**double** e=a;

System.***out***.println("Value of e: "+e);

System.***out***.println("\n");

System.***out***.println("Explicit Type Casting");

**double** x=.998;

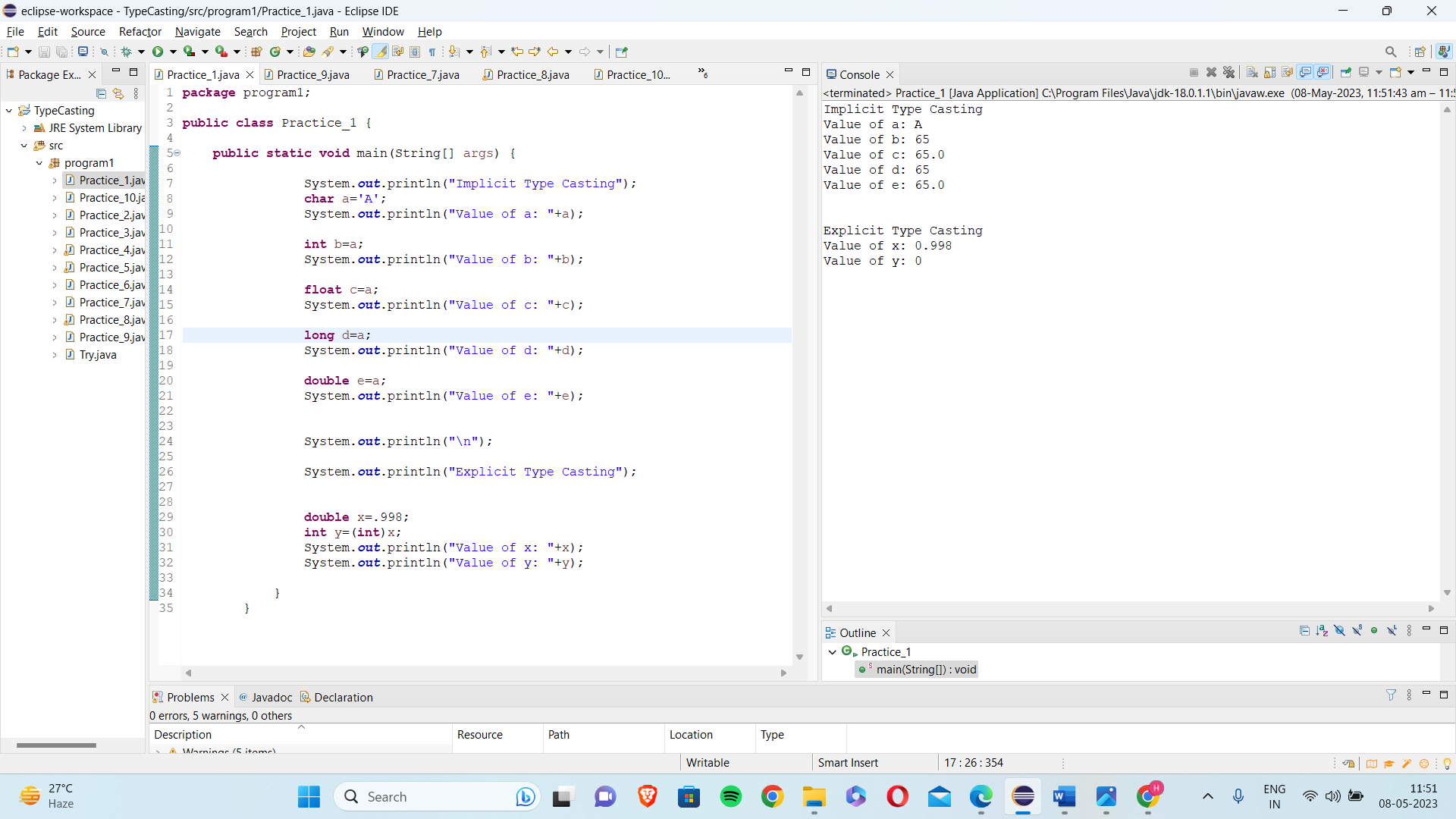
**int** y=(**int**)x;

System.***out***.println("Value of x: "+x);

System.***out***.println("Value of y: "+y);

}

}



1. Writing a program in Java to implement access modifiers

Solution:

**package** program1;

**class** Modifiers

{

**public** String x="Public Access Modifier";

**protected** String y="Protected Access Modifier";

String z="Default Access Modifier";

**private** String name;

//To access private properties,we use getters & setters

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name =name;

}

}

**public** **class** Practice\_2

{

**public** **static** **void** main(String[] args)

{

Modifiers obj=**new** Modifiers();

System.***out***.println(obj.x);

System.***out***.println(obj.y);

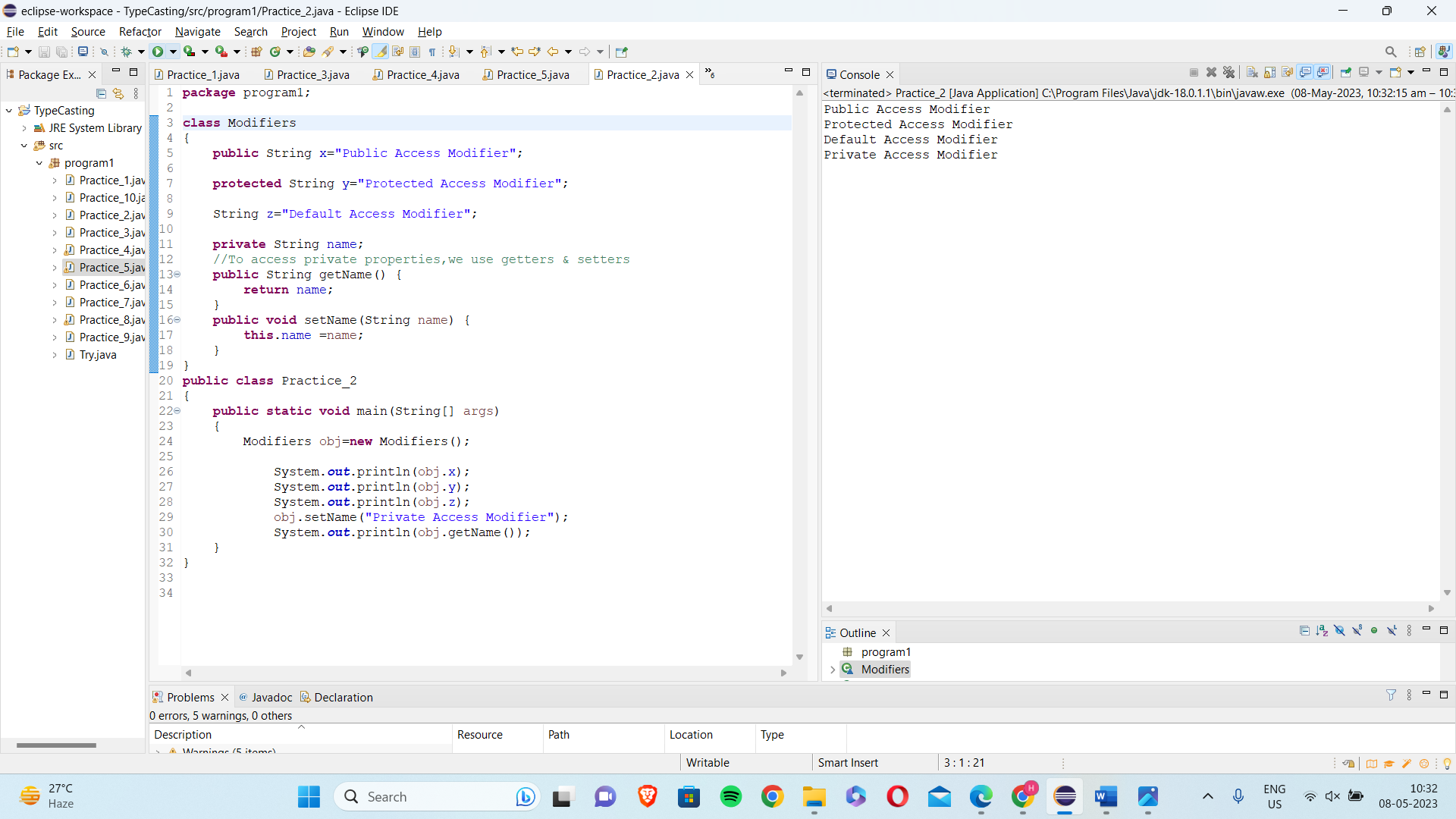
System.***out***.println(obj.z);

obj.setName("Private Access Modifier");

System.***out***.println(obj.getName());

}

}



1. Writing a program in Java to verify implementations of methods and ways of calling a method

Solution :

**package** program1;

**class** Practice

{

**void** display()

{

System.***out***.println("Calling a Method");

}

**static** **void** show()

{

System.***out***.println("calling a Static Method");

}

}

**class** Method

{

**public** **int** add(**int** a ,**int** b)

{

**int** c=a+b;

**return** c;

}

}

**public** **class** Practice\_3

{

**public** **static** **void** main(String[] args)

{

Practice p=**new** Practice();

p.display();

Practice.*show*();

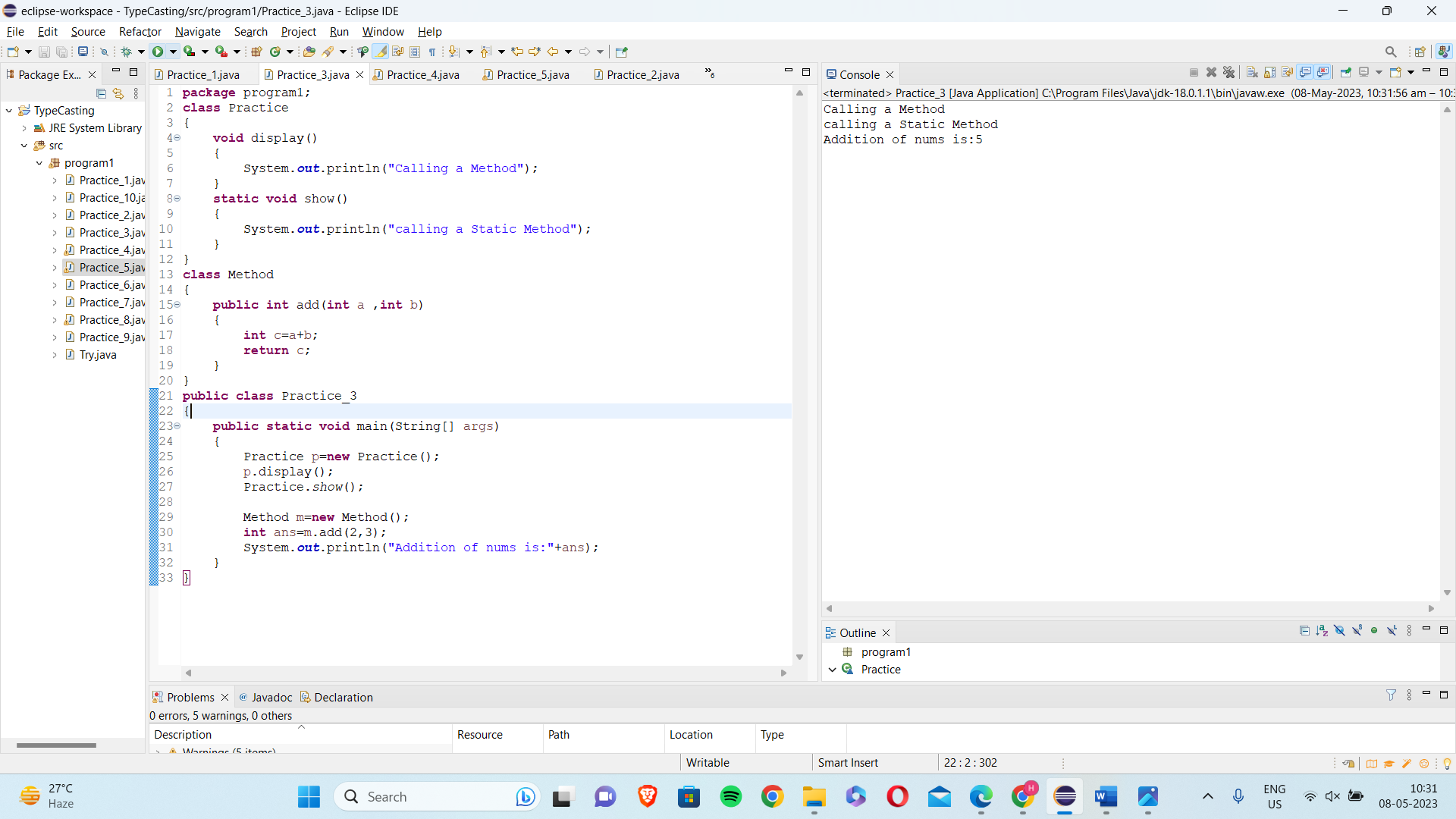
Method m=**new** Method();

**int** ans=m.add(2,3);

System.***out***.println("Addition of nums is:"+ans);

}

}



1. Writing a program in Java to verify the implementations of constructor types

Solution :

**package** program1;

**class** Constructor

{

**int** age;

String name;

Constructor()

{

System.***out***.println("Constructor is working");

}

Constructor(**int** a,String b)

{

System.***out***.println("Age is: "+a+" and"+" Name is: "+b);

}

}

**public** **class** Practice\_4

{

**public** **static** **void** main(String[] args)

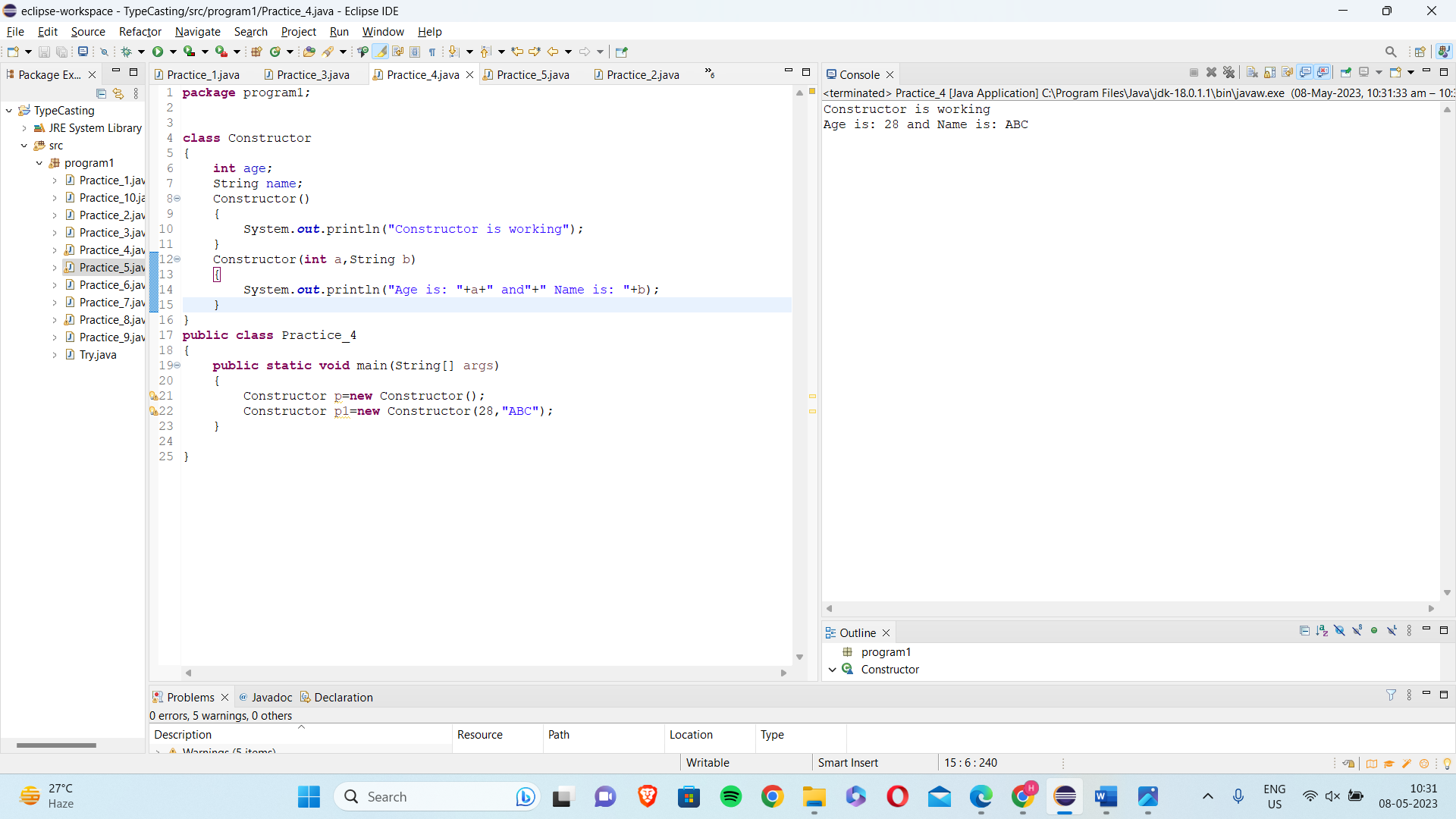
{

Constructor p=**new** Constructor();

Constructor p1=**new** Constructor(28,"ABC");

}

}



5)   
Writing a program in Java to verify implementations of collections

Solution :

**package** program1;

**import** java.util.\*;

**public** **class** Practice\_5 {

**public** **static** **void** main(String[] args) {

System.***out***.println("ArrayList");

ArrayList<String> city=**new** ArrayList<String>();

city.add("Bangalore");//

city.add("Hyderabad");

System.***out***.println(city);

System.***out***.println("\n");

System.***out***.println("Vector");

Vector<Integer> vec = **new** Vector();

vec.addElement(15);

vec.addElement(80);

System.***out***.println(vec);

System.***out***.println("\n");

System.***out***.println("LinkedList");

LinkedList<String> names=**new** LinkedList<String>();

names.add("Alex");

names.add("John");

Iterator<String> itr=names.iterator();

**while**(itr.hasNext()){

System.***out***.println(itr.next());

System.***out***.println("\n");

System.***out***.println("HashSet");

HashSet<Integer> set=**new** HashSet<Integer>();

set.add(101);

set.add(103);

set.add(102);

set.add(104);

System.***out***.println(set);

System.***out***.println("\n");

System.***out***.println("LinkedHashSet");

LinkedHashSet<Integer> set2=**new** LinkedHashSet<Integer>();

set2.add(11);

set2.add(13);

set2.add(12);

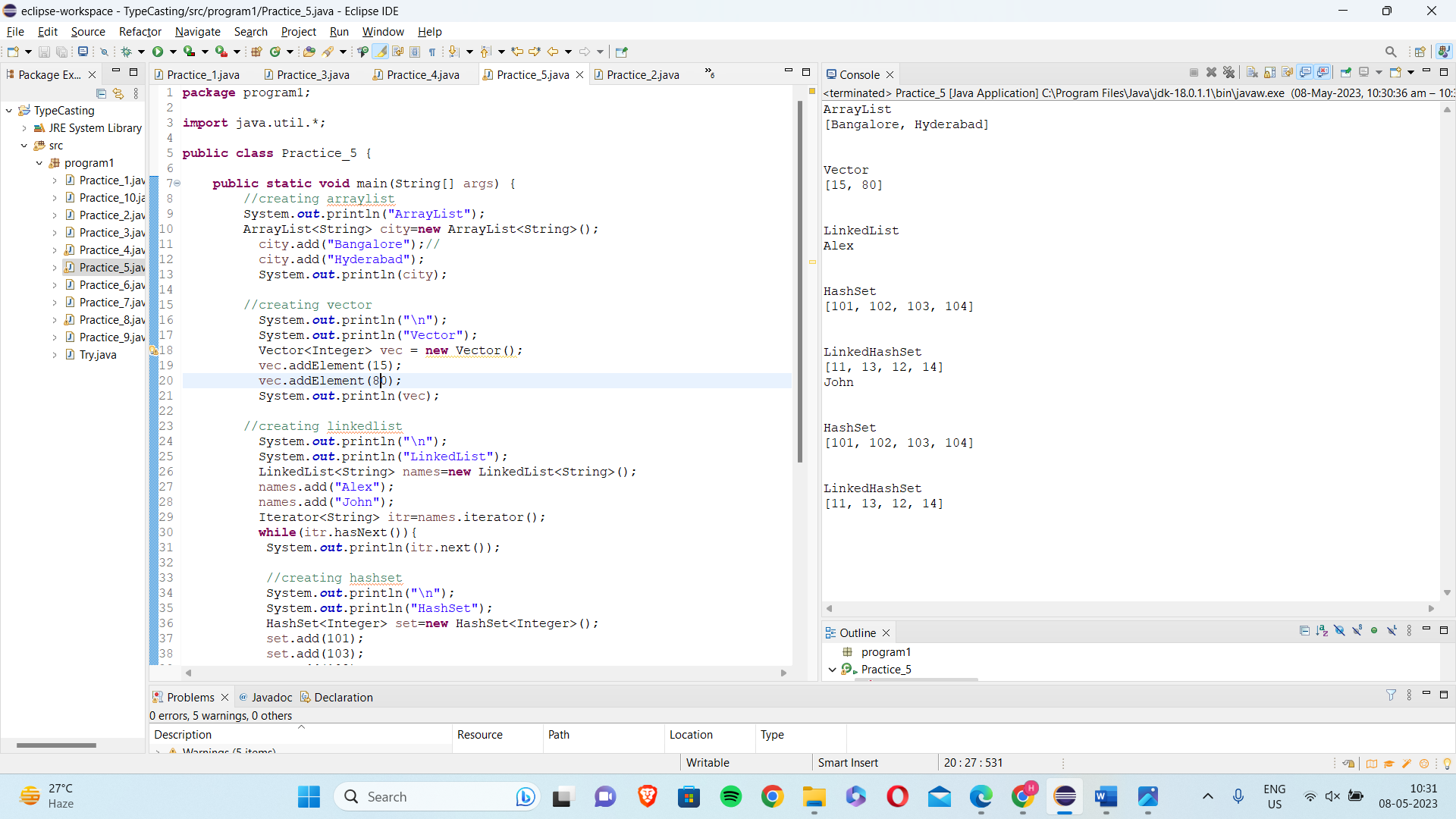
set2.add(14);

System.***out***.println(set2);

}

}

}



6)Writing a program in Java to verify implementations of maps

Solutions:

**package** program1;

**import** java.util.HashMap;

**import** java.util.Hashtable;

**import** java.util.Map;

**import** java.util.TreeMap;

**public** **class** Practice\_6 {

**public** **static** **void** main(String[] args) {

System.***out***.println("HashMap");

HashMap<Integer, String> hm = **new** HashMap<>();

hm.put(1, "Tim");

hm.put(2, "Mary");

hm.put(3, "Catie");

System.***out***.println("The elements of HashMap are ");

**for** (Map.Entry<Integer, String> m : hm.entrySet()) {

System.***out***.println(m.getKey() + " " + m.getValue());

}

System.***out***.println("\n");

System.***out***.println("HashTable");

Hashtable<Integer, String> ht = **new** Hashtable<>();

ht.put(4, "Ales");

ht.put(5, "Rosy");

ht.put(6, "Jack");

ht.put(7, "John");

System.***out***.println("The elements of HashTable are ");

**for** (Map.Entry<Integer, String> n : ht.entrySet()) {

System.***out***.println(n.getKey() + " " + n.getValue());

}

System.***out***.println("\n");

System.***out***.println("TreeMap");

TreeMap<Integer, String> map = **new** TreeMap<>();

map.put(8, "Annie");

map.put(9, "Carlotte");

map.put(10,"Catie");

System.***out***.println("The elements of TreeMap are ");

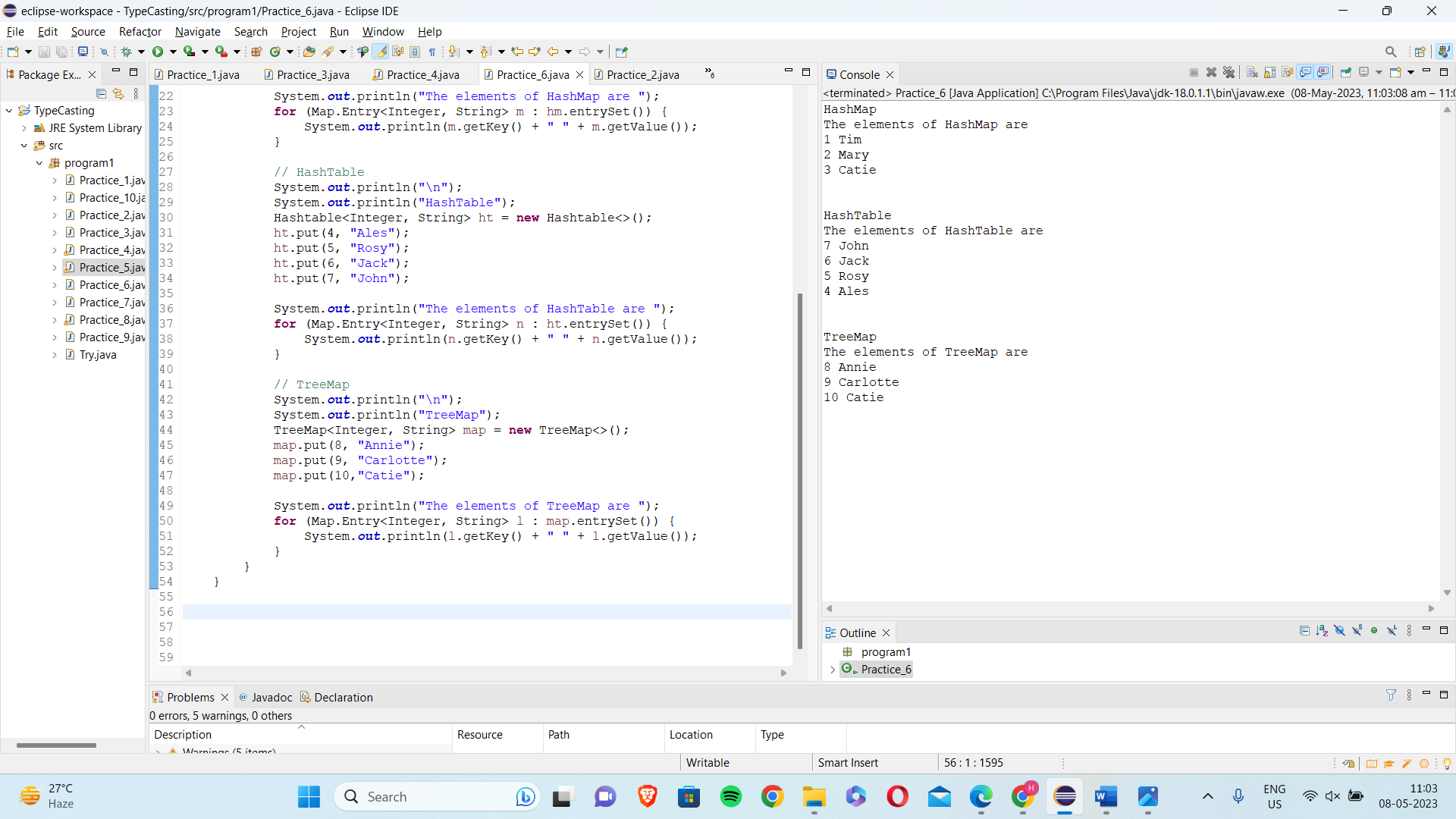
**for** (Map.Entry<Integer, String> l : map.entrySet()) {

System.***out***.println(l.getKey() + " " + l.getValue());

}

}

}



7) Writing a program in Java to verify the implementation of inner classes

Solution:

**package** program1;

**class** Pro1

{

**void** show()

{

System.***out***.println("Hey,I am Outer class");

}

**static** **class** Pro2 //inner class may have static keyword

{

**void** display()

{

System.***out***.println("Hey,I am Inner class");

}

}

}

**public** **class** Practice\_7

{

**public** **static** **void** main(String[] args)

{

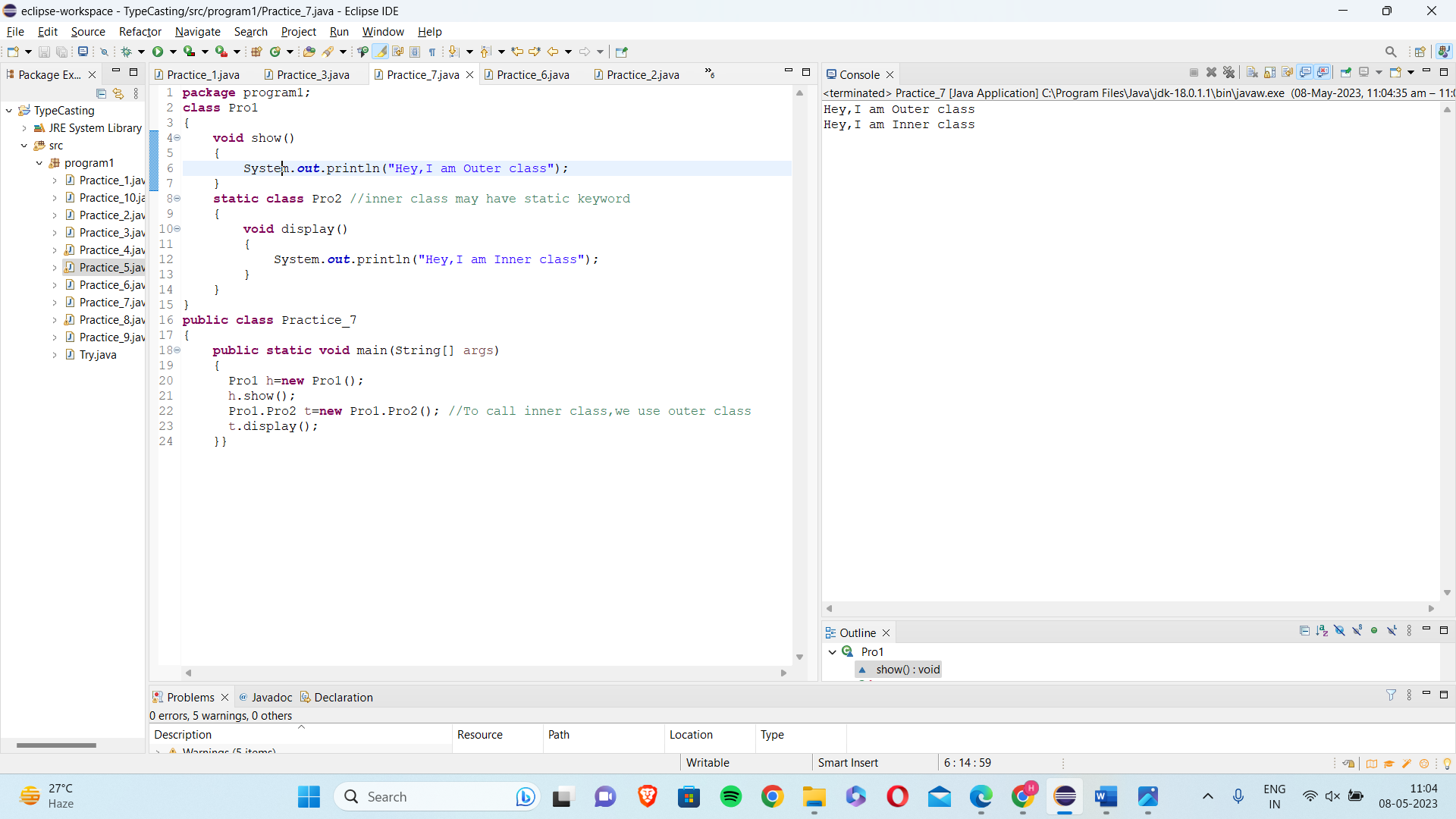
Pro1 h=**new** Pro1();

h.show();

Pro1.Pro2 t=**new** Pro1.Pro2(); //To call inner class,we use outer class

t.display();

}}



8) Write a program to create strings and display the conversion of string to StringBuffer and StringBuilder.

Solution:

**package** program1;

**import** java.io.StringReader;

**public** **class** Practice\_8

{

**public** **static** **void** main(String[] args)

{

String s=**new** String("String Implementation-1");

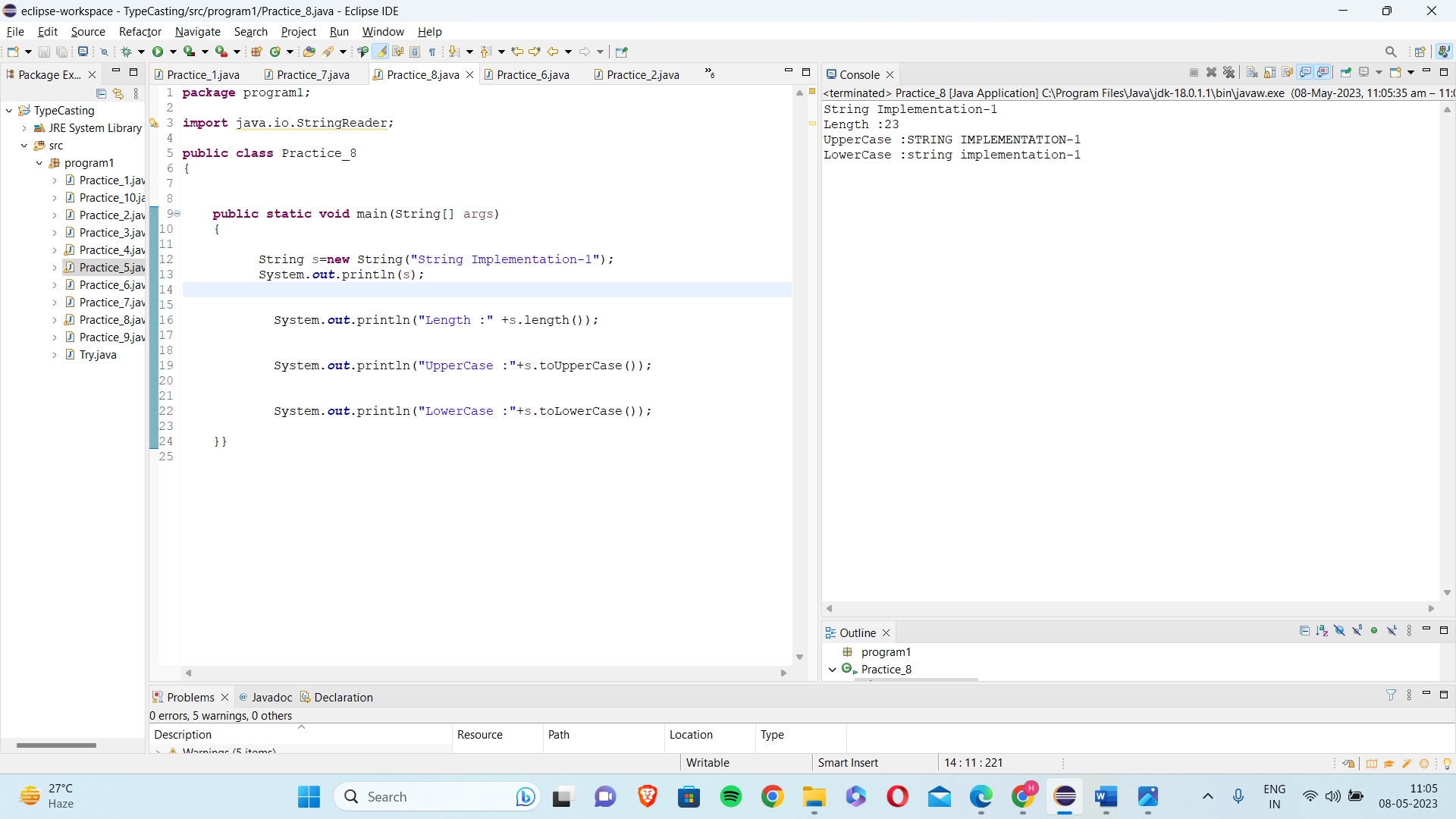
System.***out***.println(s);

System.***out***.println("Length :" +s.length());

System.***out***.println("UpperCase :"+s.toUpperCase());

System.***out***.println("LowerCase :"+s.toLowerCase());

}}



9) Writing a program in Java to verify implementation of arrays

Solution:

**package** program1;

**public** **class** Practice\_9 {

**public** **static** **void** main(String[] args) {

//single-dimensional array

**int** a[]= {10,20,30,40,50};

**for**(**int** i=0;i<5;i++) {

System.***out***.println("Elements of array a: "+a[i]);

}

//multidimensional array

**int**[][] b = {

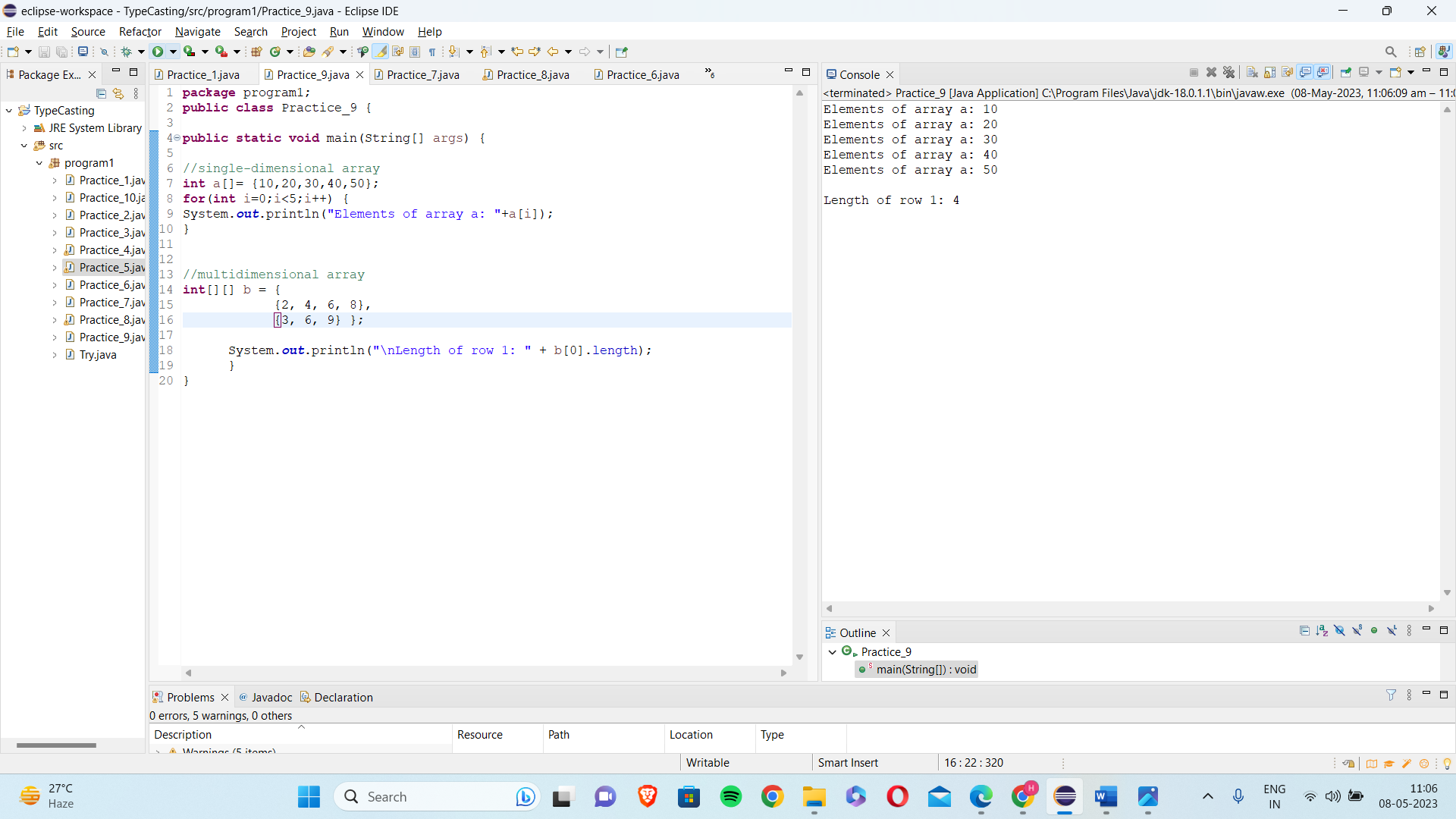
{2, 4, 6, 8},

{3, 6, 9} };

System.***out***.println("\nLength of row 1: " + b[0].length);

}

}



10) Writing a program in Java to verify implementations of regular expressions

Solution:

**package** program1;

**import** java.util.regex.\*;

**public** **class** Practice\_10 {

**public** **static** **void** main(String[] args) {

String pattern = "[a-z]+";

String check = "Regular Expressions";

Pattern p = Pattern.*compile*(pattern);

Matcher c = p.matcher(check);

**while** (c.find())

System.***out***.println( check.substring( c.start(), c.end() ) );

}

}

OUTPUT:

A screenshot of a computer

Description automatically generated