7. Demonstrate types of Inheritance?

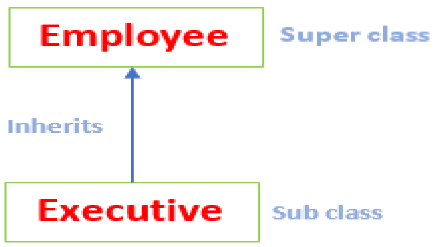
A. Inheritance is the most powerful feature of object oriented programming. It allows us to inherit the properties of one class into another class.

There are 4 types of Inheritances. They are:

- Single Inheritance
- Multi-level Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

i. Single Inheritance:

In single inheritance, a sub-class is derived from only one super class. It inherits the properties and behaviour of a single-parent class. Sometimes it is also known as simple inheritance.



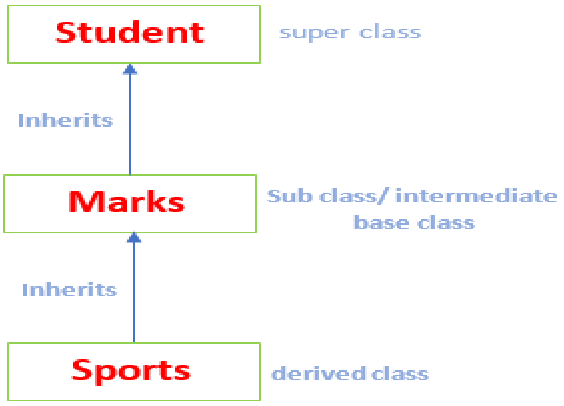
Single Inheritance

```
class Employee
{
float salary=34534*12;
}
public class Executive extends Employee
{
float bonus=3000*6;
public static void main(String args[])
{
Executive obj=new Executive();
System.out.println("Total salary credited: "+obj.salary);
System.out.println("Bonus of six months: "+obj.bonus);
}
}
```

```
Output: Total salary credited: 414408.0
        Bonus of six months: 18000.0
```

ii. Multi-level Inheritance:

In multi-level inheritance a class is derived from a class which is also derived from another class or in a simple words, a class which is having more than one parent class is called as multi-level inheritance.



Multi-level Inheritance

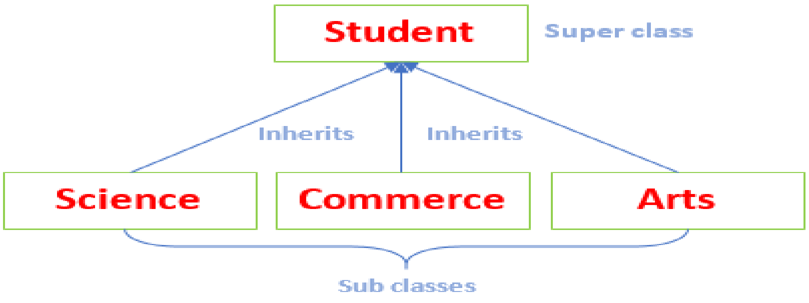
```
//super class
    class Student
    {
    int reg_no;
    void getNo(int no)
        {
    reg_no=no;
    }
     void putNo()
    {
        System.out.println("registration number= "+reg_no);
        }
    }
    //intermediate sub class
    class Marks extends Student
    {
    float marks;
    void getMarks(float m)
    {
    marks=m;
    }
    void putMarks()
{
  System.out.println("marks= "+marks);
}
}
//derived class
class Sports extends Marks
{
float score;
void getScore(float scr)
{
score=scr;
}
void putScore()
{
System.out.println("score= "+score);
}
}
public class MultilevelInheritanceExample
{
public static void main(String args[])
{
Sports ob=new Sports();
ob.getNo(0987);
ob.putNo();
ob.getMarks(78);
ob.putMarks();
ob.getScore(68.7);
ob.putScore();
}
    }
```

Output:

```
registration number= 0987
marks= 78.0
score= 68.7
```

iii. Hierarchical Inheritance:

If a number of classes derived from a single base class, it is known as hierarchical inheritance.



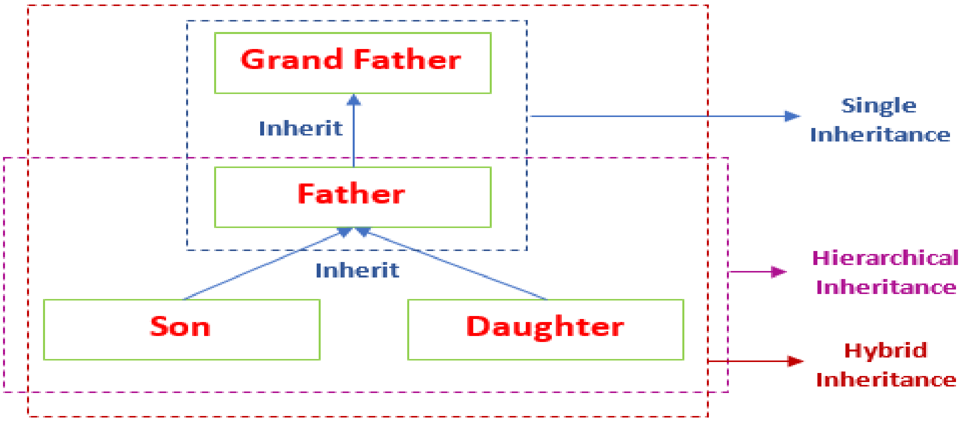
Hierarchical Inheritance

```
//parent class
class Student
{
public void methodStudent()
{
System.out.println("The method of the class Student invoked.");
}
}
class Science extends Student
{
public void methodScience()
{
System.out.println("The method of the class Science invoked.");
}
}
class Commerce extends Student
{
public void methodCommerce()
{
System.out.println("The method of the class Commerce invoked.");
}
}
class Arts extends Student
{
public void methodArts()
{
System.out.println("The method of the class Arts invoked.");
}
}
public class HierarchicalInheritanceExample
{
public static void main(String args[])
{
Science sci = new Science();
Commerce comm = new Commerce();
Arts art = new Arts();
//all the sub classes can access the method of super class
sci.methodStudent();
comm.methodStudent();
art.methodStudent();
}
}
```

Output:

```
The method of the class Student invoked.
The method of the class Student invoked.
The method of the class Student invoked.
```

iv. Hybrid Inheritance:
It is a combination of two or more types of inheritance.



Hybrid Inheritance

```
//parent class
class GrandFather
{
public void show()
{
System.out.println("I am grandfather.");
}
}
//inherits GrandFather properties
class Father extends GrandFather
{
public void show()
{
System.out.println("I am father.");
}
}
//inherits Father properties
class Son extends Father
{
public void show()
{
System.out.println("I am son.");
}
}
//inherits Father properties
public class Daughter extends Father
{
public void show()
{
System.out.println("I am a daughter.");
}
public static void main(String args[])
{
Daughter obj = new Daughter();
obj.show();
}
}
```

Output:

```
I am daughter.
```

6.Constructor:

> creates an instance of a class
> similar to a java method,except :
    > name is same as the class name
    > it will not have a return type
> Whenever we write the keyword new, to create an instance of a class. A default constructor will be invoked and an object of the class is returned.

Types of constructors:
  > default constructor
    > the role of default contructor is to initialize the object and return it to the calling code
    > default constructor is always without an argument

  > No argument constructor

  > parameterized constructor

```
package constructorDemo;
public class ConstructorDemo {

	// no argument constructor - syntax

	public ConstructorDemo()
	{
		System.out.println(" this is no argument 
constructor");
	}

	// a constructor with argument

	public ConstructorDemo( int a)
	{
		System.out.println(" this is argument 
constructor");
		System.out.println(" this value of argument a 
: " + a);
	}

	public ConstructorDemo( int a, int b)
	{
		System.out.println(" this is multiple 
argument constructor");
```

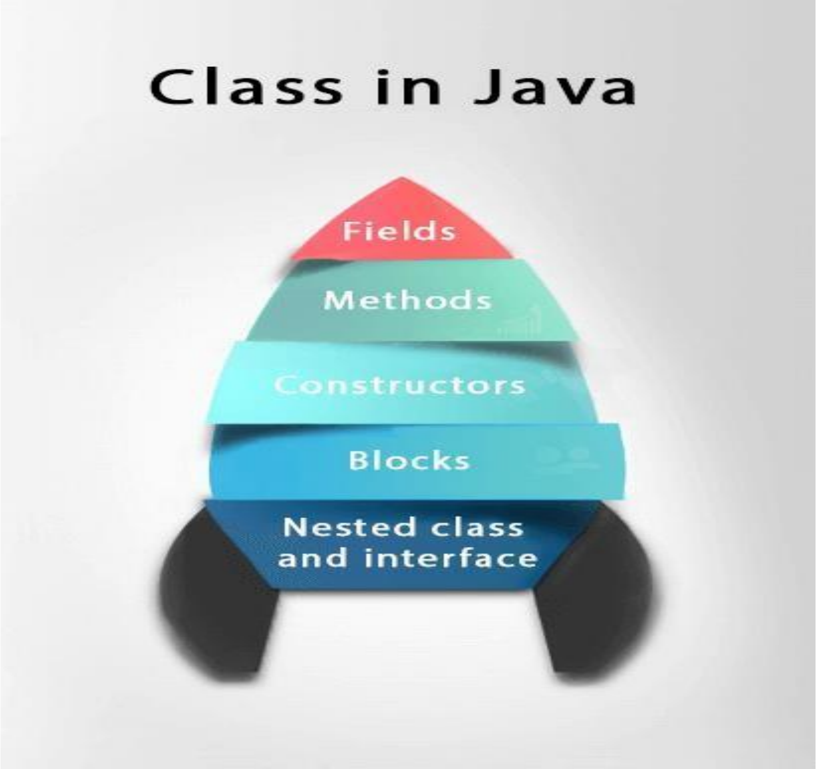
```
        System.out.println(" this value of argument a
: " + a);
        System.out.println(" this value of argument b
: " + b);
    }

    public static void main(String[] args) {

        // to execute a constructor just create an
object

        ConstructorDemo obj    = new
ConstructorDemo();

        ConstructorDemo obj2  = new
ConstructorDemo(23); // value of a is 23

        ConstructorDemo obj3  = new
ConstructorDemo(68,34);



    }

}
```

Classes:

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- **Fields**
- **Methods**
- **Constructors**
- **Blocks**
- **Nested class and interface**



Syntax to declare a class:

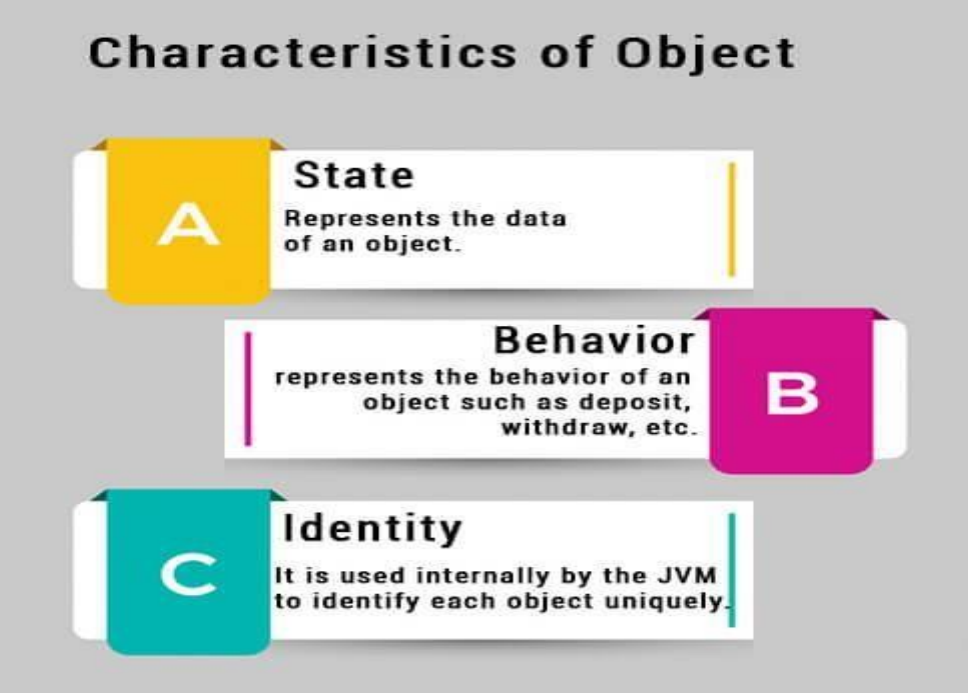
```
class < class_name > {
   field;
   method;
}
```

Objects:

An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc.

(Or)

- An object is *a real-world entity*.



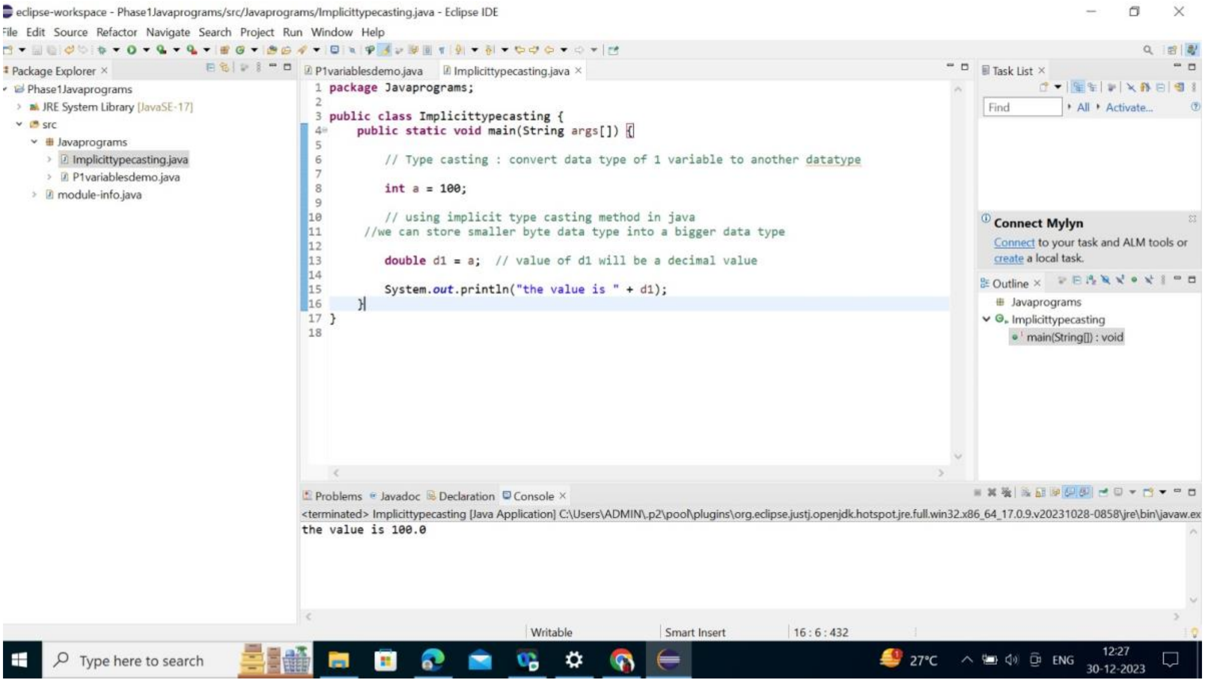
Characteristics of Object

A **State**
Represents the data of an object.

B **Behavior**
represents the behavior of an object such as deposit, withdraw, etc.

C **Identity**
It is used internally by the JVM to identify each object uniquely.

1.Implicit type casting:

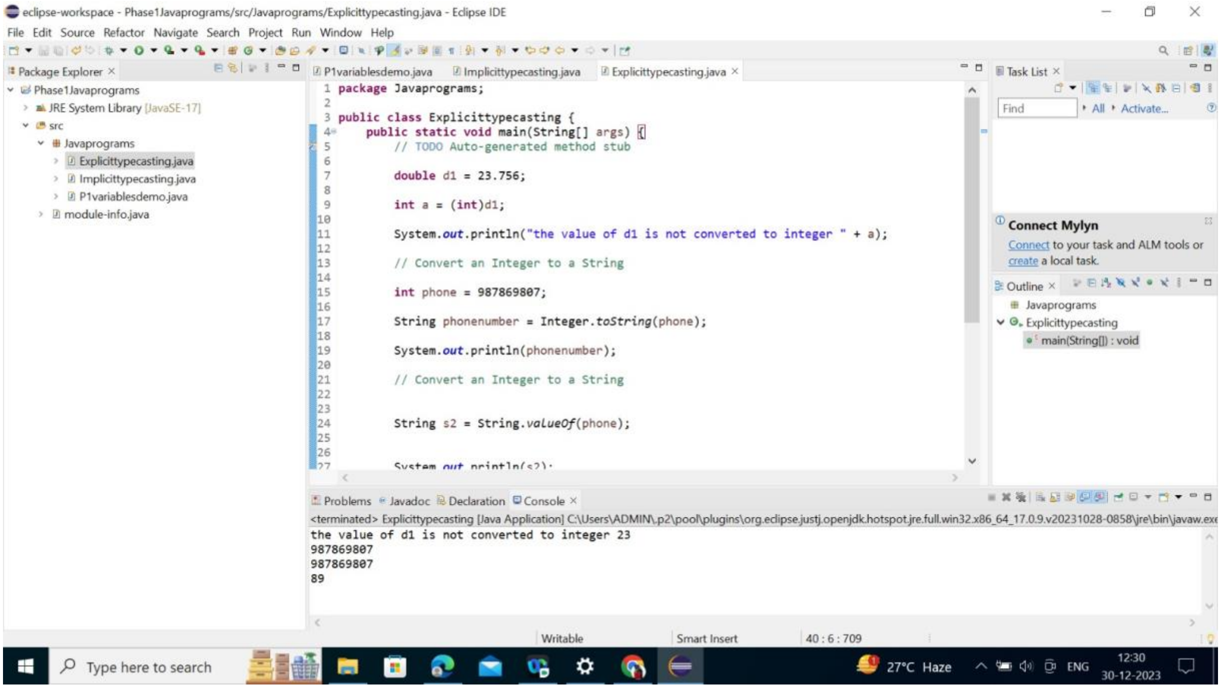


eclipse-workspace - Phase1Javaprograms/src/Javaprograms/Implicittypecasting.java - Eclipse IDE

```java
package Javaprograms;

public class Implicittypecasting {
    public static void main(String args[]) {

        // Type casting : convert data type of 1 variable to another datatype

        int a = 100;

        // using implicit type casting method in java
      //we can store smaller byte data type into a bigger data type

        double d1 = a;  // value of d1 will be a decimal value

        System.out.println("the value is " + d1);
    }
}
```

```
<terminated> Implicittypecasting [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.ex
the value is 100.0
```

Explicit type casting:

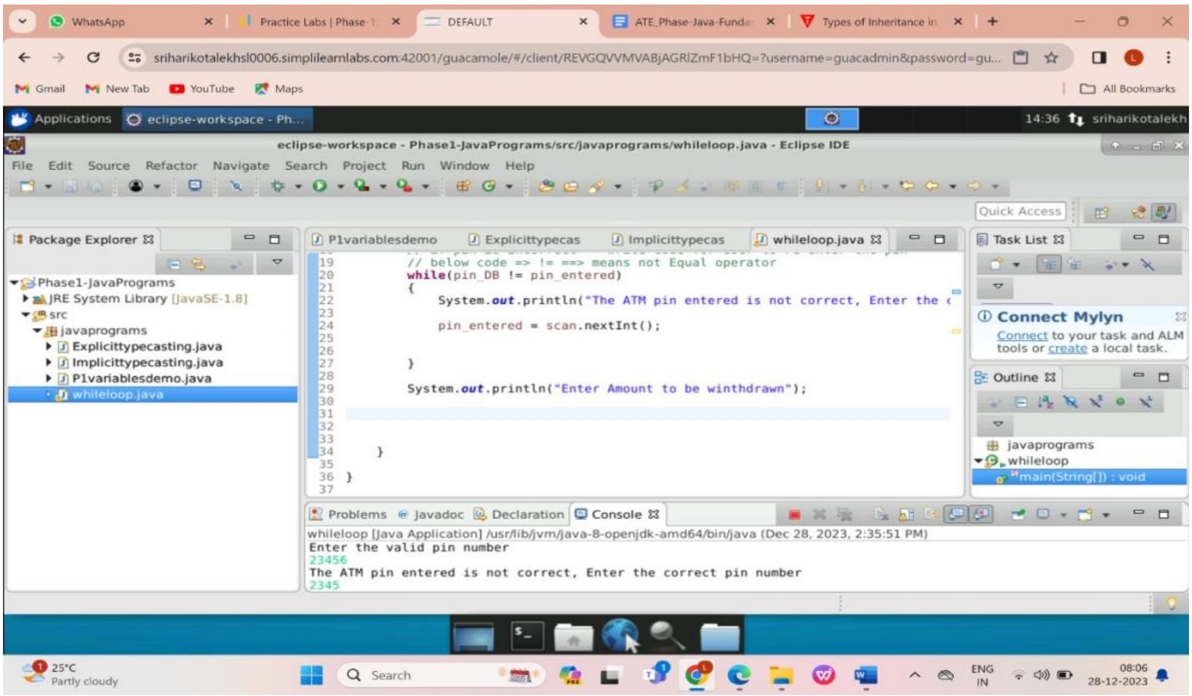


```
package Javaprograms;

public class Explicittypecasting {
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        double d1 = 23.756;

        int a = (int)d1;

        System.out.println("the value of d1 is not converted to integer " + a);

        // Convert an Integer to a String

        int phone = 987869807;

        String phonenumber = Integer.toString(phone);

        System.out.println(phonenumber);

        // Convert an Integer to a String


        String s2 = String.valueOf(phone);


        System.out.println(s2);
```

<terminated> Explicittypecasting [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe

```
the value of d1 is not converted to integer 23
987869807
987869807
89
```

3. While loop:

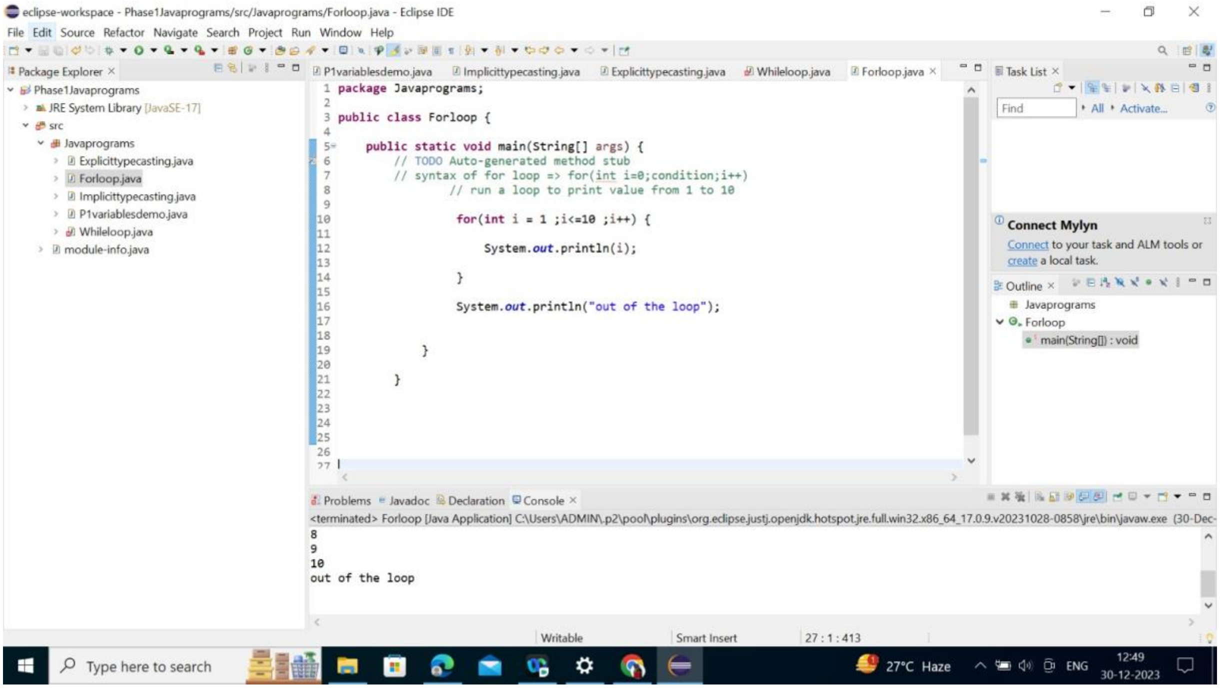


```
        // below code => != ==> means not Equal operator
        while(pin_DB != pin_entered)
        {
            System.out.println("The ATM pin entered is not correct, Enter the c

            pin_entered = scan.nextInt();


        }

        System.out.println("Enter Amount to be winthdrawn");




    }

}
```

whileloop [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Dec 28, 2023, 2:35:51 PM)

```
Enter the valid pin number
23456
The ATM pin entered is not correct, Enter the correct pin number
2345
```

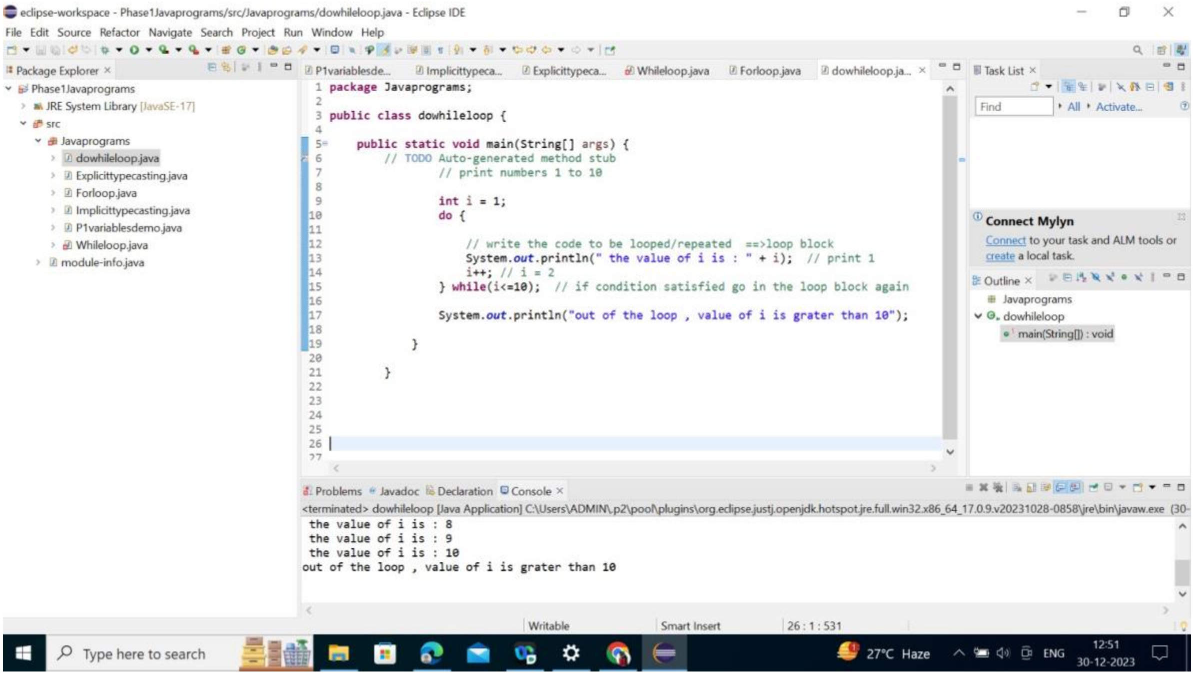
4. for loop:



```
package Javaprograms;

public class Forloop {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		// syntax of for loop => for(int i=0;condition;i++)
				// run a loop to print value from 1 to 10
		
				for(int i = 1 ;i<=10 ;i++) {
					
					System.out.println(i);
					
				}
				
				System.out.println("out of the loop");
				
				
			}
			
		}
```

```
<terminated> Forloop [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (30-Dec-
8
9
10
out of the loop
```

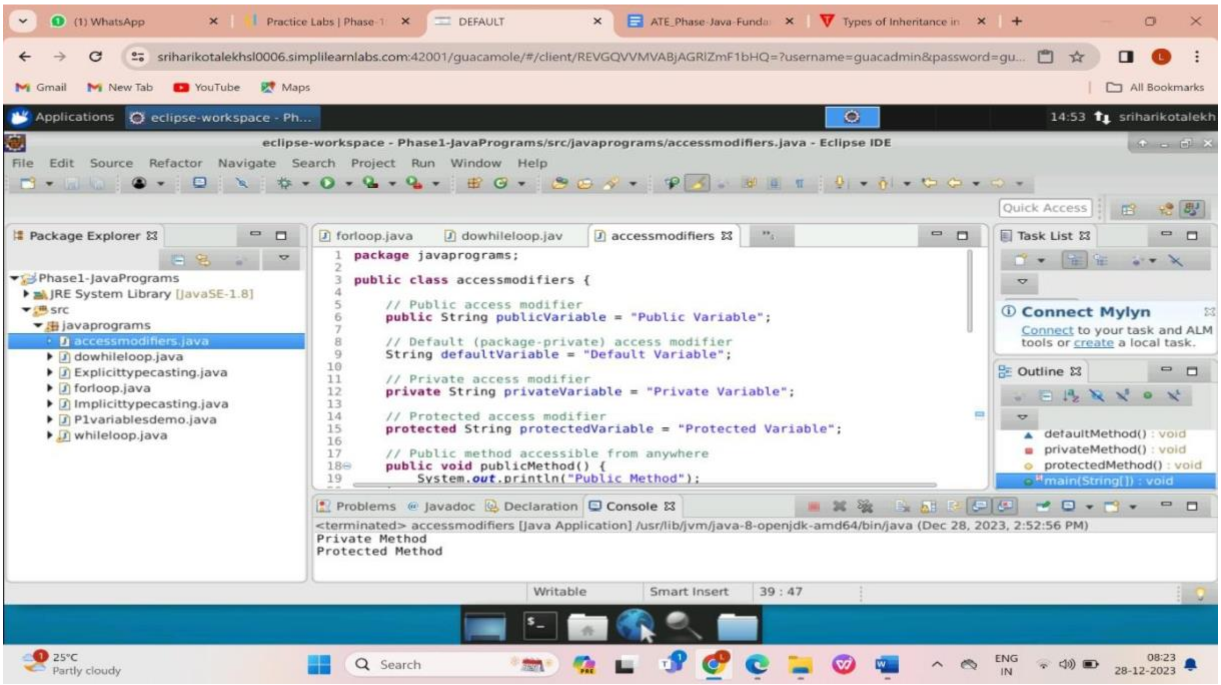
5. do-while loop:



```
package Javaprograms;

public class dowhileloop {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
				// print numbers 1 to 10
		
				int i = 1;
				do {
					
					// write the code to be looped/repeated  ==>loop block
					System.out.println(" the value of i is : " + i);  // print 1
					i++; // i = 2
				} while(i<=10);  // if condition satisfied go in the loop block again
				
				System.out.println("out of the loop , value of i is grater than 10");
				
			}
			
		}
```

```
<terminated> dowhileloop [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (30-
 the value of i is : 8
 the value of i is : 9
 the value of i is : 10
out of the loop , value of i is grater than 10
```

2.Access Modifiers:

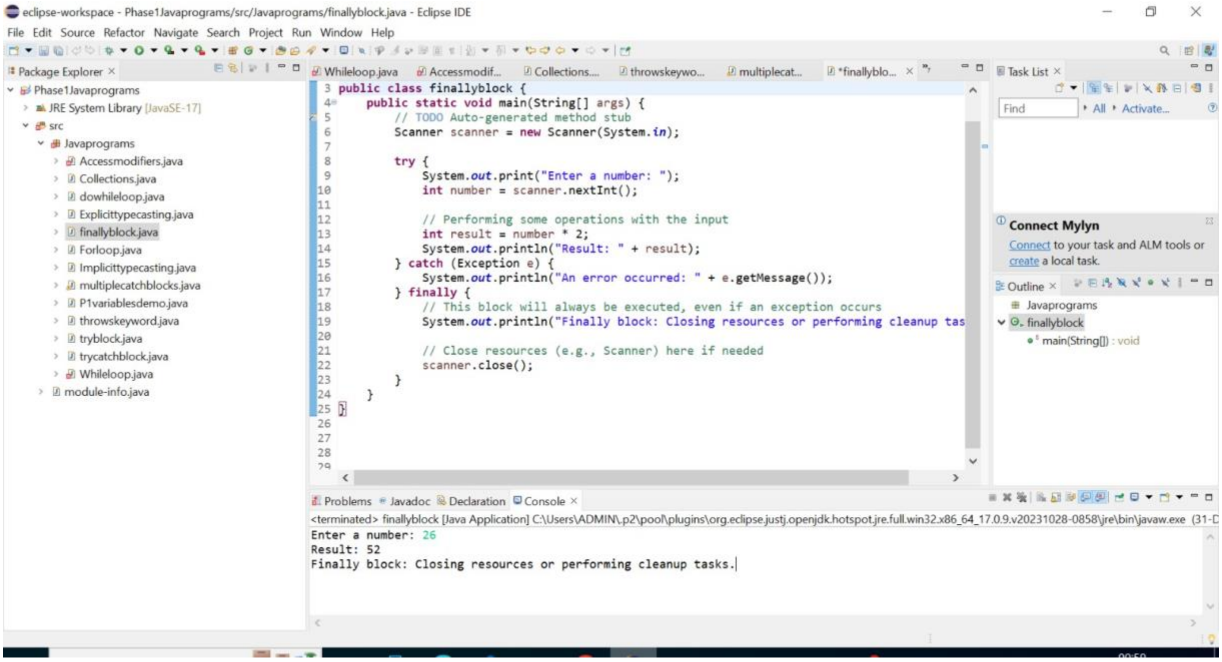


eclipse-workspace - Phase1-JavaPrograms/src/javaprograms/accessmodifiers.java - Eclipse IDE

```java
package javaprograms;

public class accessmodifiers {

    // Public access modifier
    public String publicVariable = "Public Variable";

    // Default (package-private) access modifier
    String defaultVariable = "Default Variable";

    // Private access modifier
    private String privateVariable = "Private Variable";

    // Protected access modifier
    protected String protectedVariable = "Protected Variable";

    // Public method accessible from anywhere
    public void publicMethod() {
        System.out.println("Public Method");
```

```
<terminated> accessmodifiers [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Dec 28, 2023, 2:52:56 PM)
Private Method
Protected Method
```

12.finally block:

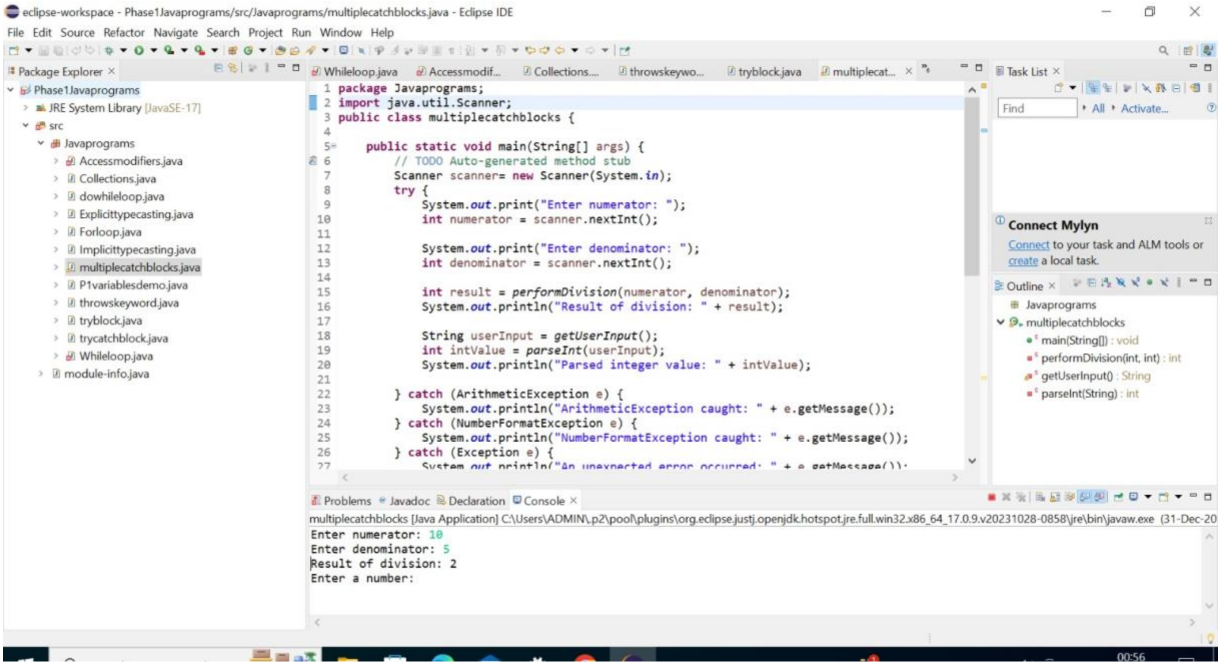


eclipse-workspace - Phase1Javaprograms/src/Javaprograms/finallyblock.java - Eclipse IDE

```java
public class finallyblock {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter a number: ");
            int number = scanner.nextInt();

            // Performing some operations with the input
            int result = number * 2;
            System.out.println("Result: " + result);
        } catch (Exception e) {
            System.out.println("An error occurred: " + e.getMessage());
        } finally {
            // This block will always be executed, even if an exception occurs
            System.out.println("Finally block: Closing resources or performing cleanup tas

            // Close resources (e.g., Scanner) here if needed
            scanner.close();
        }
    }
}
```

```
<terminated> finallyblock [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (31-D
Enter a number: 26
Result: 52
Finally block: Closing resources or performing cleanup tasks.
```

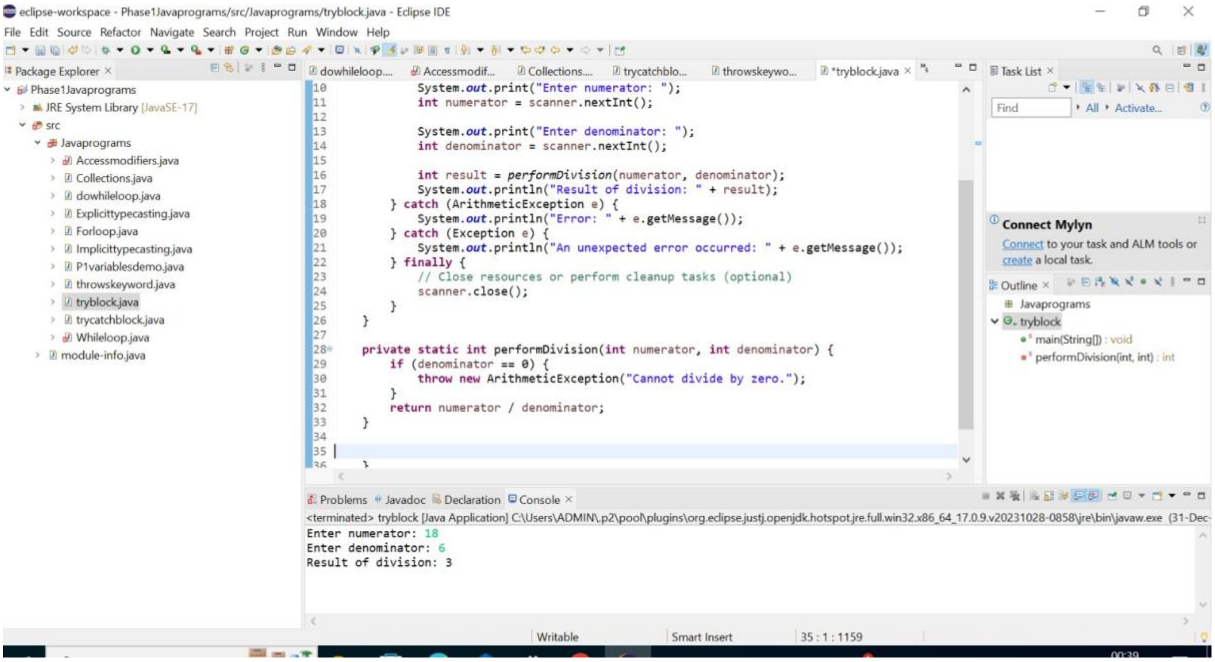
11.Multiple catch blocks:



```java
package Javaprograms;
import java.util.Scanner;
public class multiplecatchblocks {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner= new Scanner(System.in);
        try {
            System.out.print("Enter numerator: ");
            int numerator = scanner.nextInt();

            System.out.print("Enter denominator: ");
            int denominator = scanner.nextInt();

            int result = performDivision(numerator, denominator);
            System.out.println("Result of division: " + result);

            String userInput = getUserInput();
            int intValue = parseInt(userInput);
            System.out.println("Parsed integer value: " + intValue);

        } catch (ArithmeticException e) {
            System.out.println("ArithmeticException caught: " + e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("NumberFormatException caught: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
```

```
multiplecatchblocks [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (31-Dec-20
Enter numerator: 10
Enter denominator: 5
Result of division: 2
Enter a number:
```

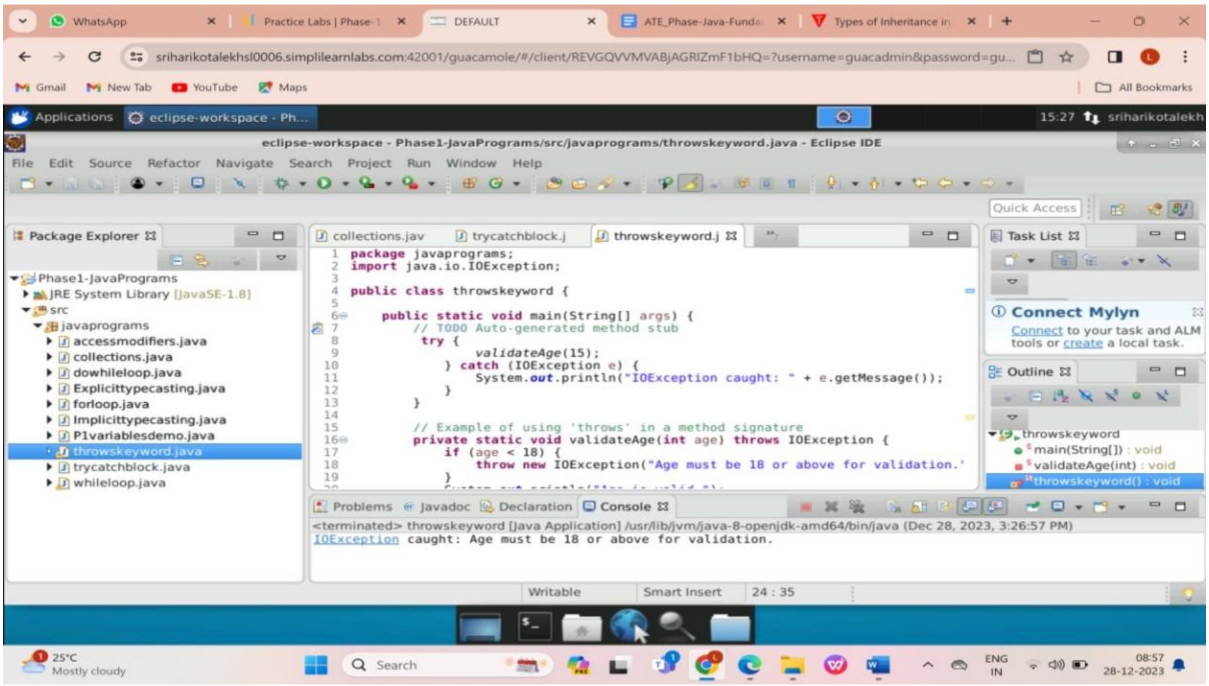
10.Try block:



```java
            System.out.print("Enter numerator: ");
            int numerator = scanner.nextInt();

            System.out.print("Enter denominator: ");
            int denominator = scanner.nextInt();

            int result = performDivision(numerator, denominator);
            System.out.println("Result of division: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            // Close resources or perform cleanup tasks (optional)
            scanner.close();
        }
    }

    private static int performDivision(int numerator, int denominator) {
        if (denominator == 0) {
            throw new ArithmeticException("Cannot divide by zero.");
        }
        return numerator / denominator;
    }


}
```

```
<terminated> tryblock [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (31-Dec-
Enter numerator: 18
Enter denominator: 6
Result of division: 3
```

9.Throws keyword:

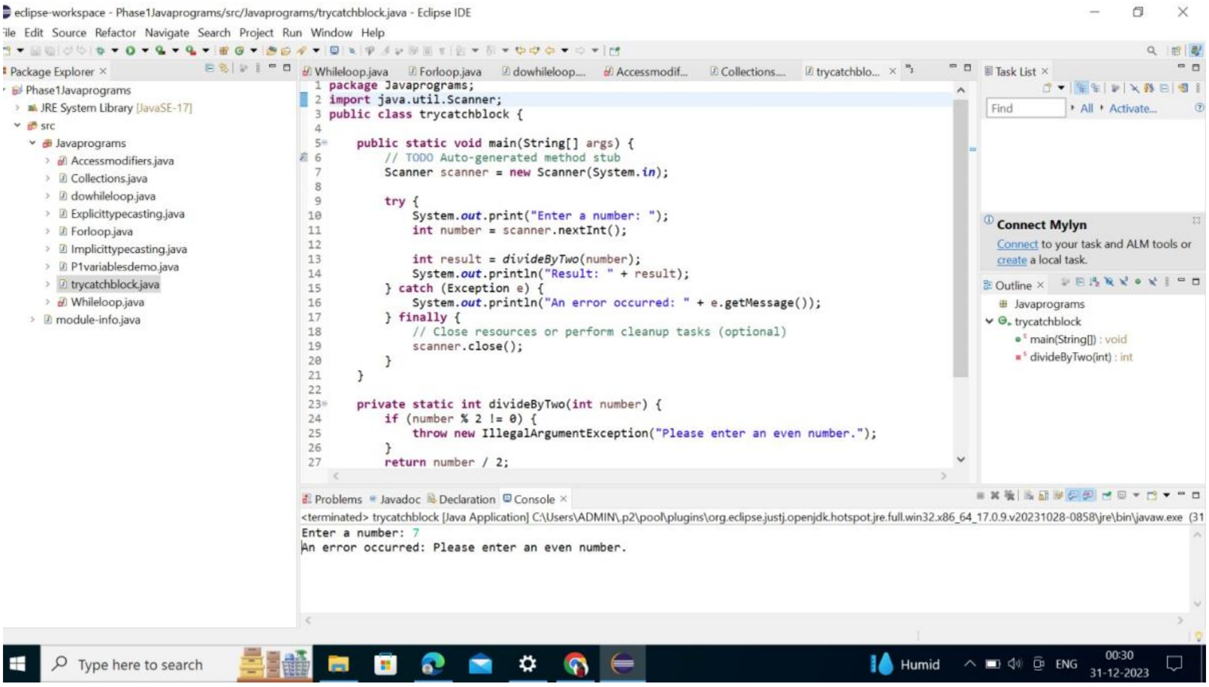


eclipse-workspace - Phase1-JavaPrograms/src/javaprograms/throwskeyword.java - Eclipse IDE

```java
package javaprograms;
import java.io.IOException;

public class throwskeyword {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
         try {
                validateAge(15);
            } catch (IOException e) {
                System.out.println("IOException caught: " + e.getMessage());
            }
        }

        // Example of using 'throws' in a method signature
        private static void validateAge(int age) throws IOException {
            if (age < 18) {
                throw new IOException("Age must be 18 or above for validation.'
            }
```

```
<terminated> throwskeyword [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Dec 28, 2023, 3:26:57 PM)
IOException caught: Age must be 18 or above for validation.
```

8.Try catch:

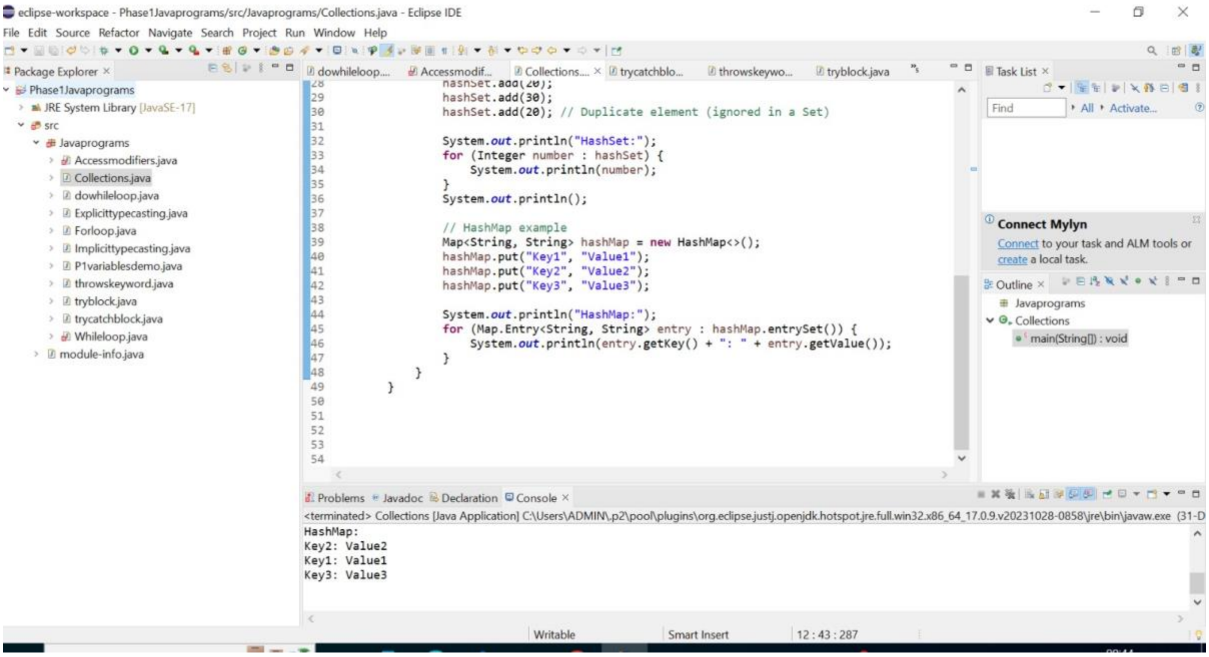


eclipse-workspace - Phase1Javaprograms/src/Javaprograms/trycatchblock.java - Eclipse IDE

```java
package Javaprograms;
import java.util.Scanner;
public class trycatchblock {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter a number: ");
            int number = scanner.nextInt();

            int result = divideByTwo(number);
            System.out.println("Result: " + result);
        } catch (Exception e) {
            System.out.println("An error occurred: " + e.getMessage());
        } finally {
            // Close resources or perform cleanup tasks (optional)
            scanner.close();
        }
    }

    private static int divideByTwo(int number) {
        if (number % 2 != 0) {
            throw new IllegalArgumentException("Please enter an even number.");
        }
        return number / 2;
```

```
<terminated> trycatchblock [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (31
Enter a number: 7
An error occurred: Please enter an even number.
```

7.Collections:



```
                hashSet.add(20);
                hashSet.add(30);
                hashSet.add(20); // Duplicate element (ignored in a Set)

                System.out.println("HashSet:");
                for (Integer number : hashSet) {
                    System.out.println(number);
                }
                System.out.println();

                // HashMap example
                Map<String, String> hashMap = new HashMap<>();
                hashMap.put("Key1", "Value1");
                hashMap.put("Key2", "Value2");
                hashMap.put("Key3", "Value3");

                System.out.println("HashMap:");
                for (Map.Entry<String, String> entry : hashMap.entrySet()) {
                    System.out.println(entry.getKey() + ": " + entry.getValue());
                }
            }
        }
```

```
<terminated> Collections [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (31-D
HashMap:
Key2: Value2
Key1: Value1
Key3: Value3
```