7.Demonstrate types of Inheritance?

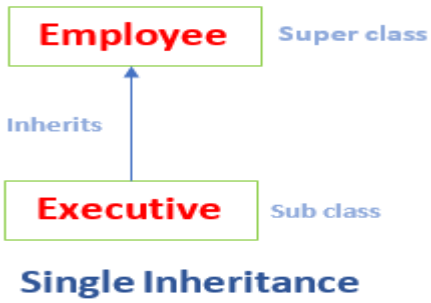
A. Inheritance is the most powerful feature of object oriented programming. It allows us to inherit the properties of one class into another class.

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.



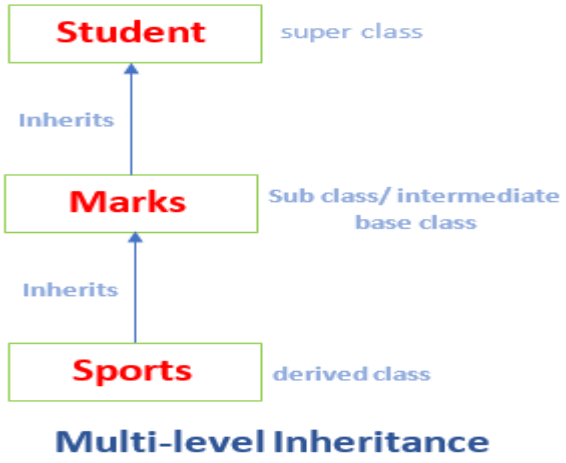
Single Inheritance

```
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()
```

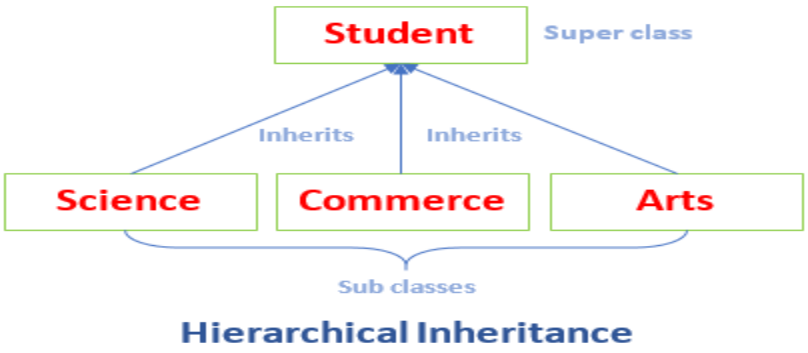
```
{
   System.out.println("marks= "+marks);
}
}
//derived class
class Sports extends Marks
{
float score;
void getScore(float scr)
{
score=scr;
}
void putScore()
{
System.out.println("score= "+score);
}
}
public class MultilevelInheritanceExample
{
public static void main(String args[])
{
Sports ob=new 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.");
```

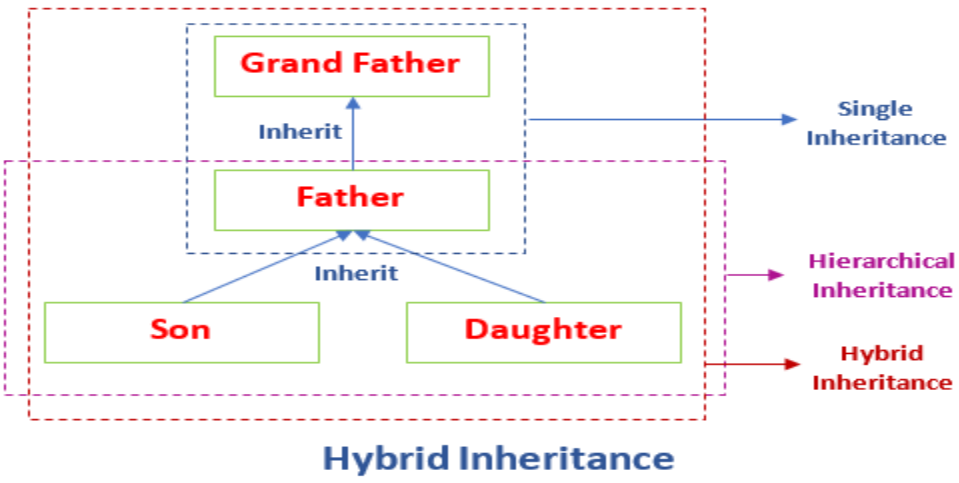
```
}
}
public class HierarchicalInheritanceExample
{
public static void main(String args[])
{
Science sci = new Science();
Commerce comm = new Commerce();
Arts art = new Arts();
//all the sub classes can access the method of super class
sci.methodStudent();
comm.methodStudent();
art.methodStudent();
}
}
```

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");
```

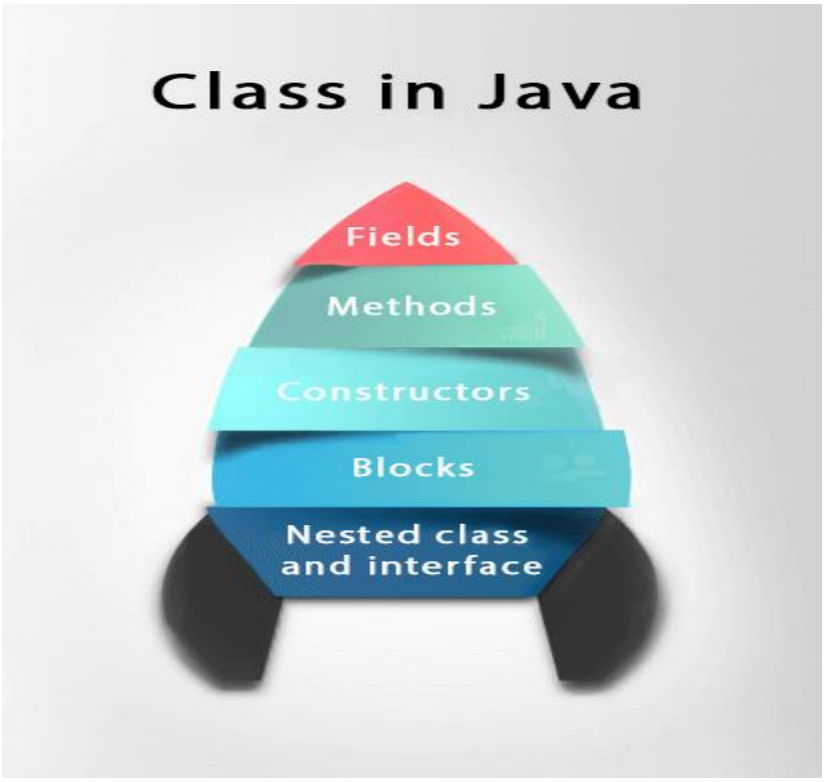
```
        System.out.println(" this value of argument a
: " + a);
        System.out.println(" this value of argument b
: " + b);
    }

    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:

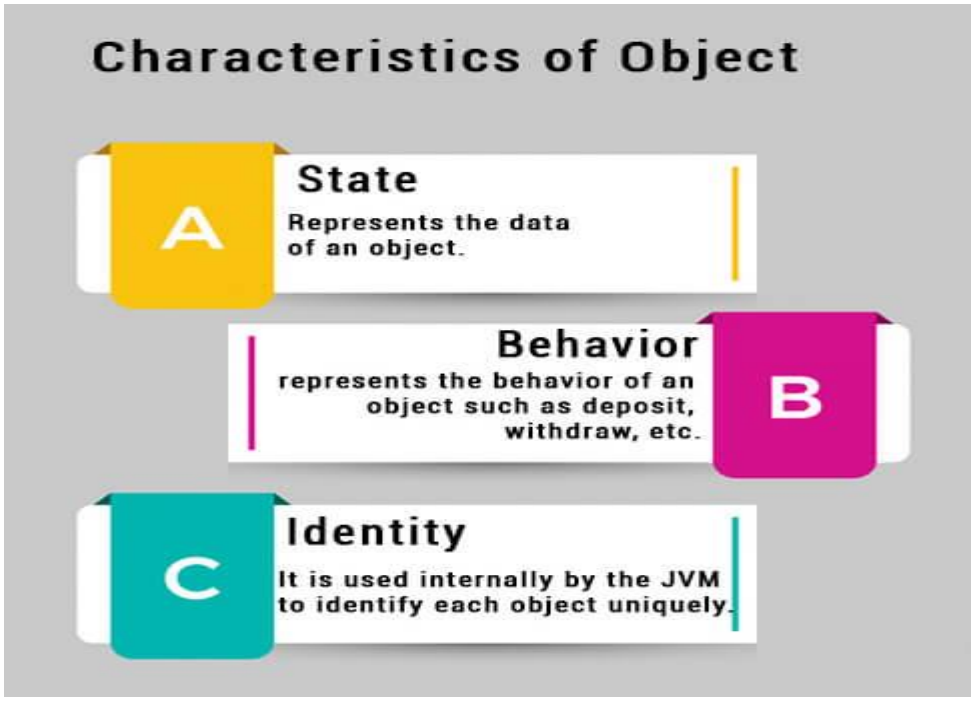
```
class < class_name > {
    field;
    method;
}
```

Objects:

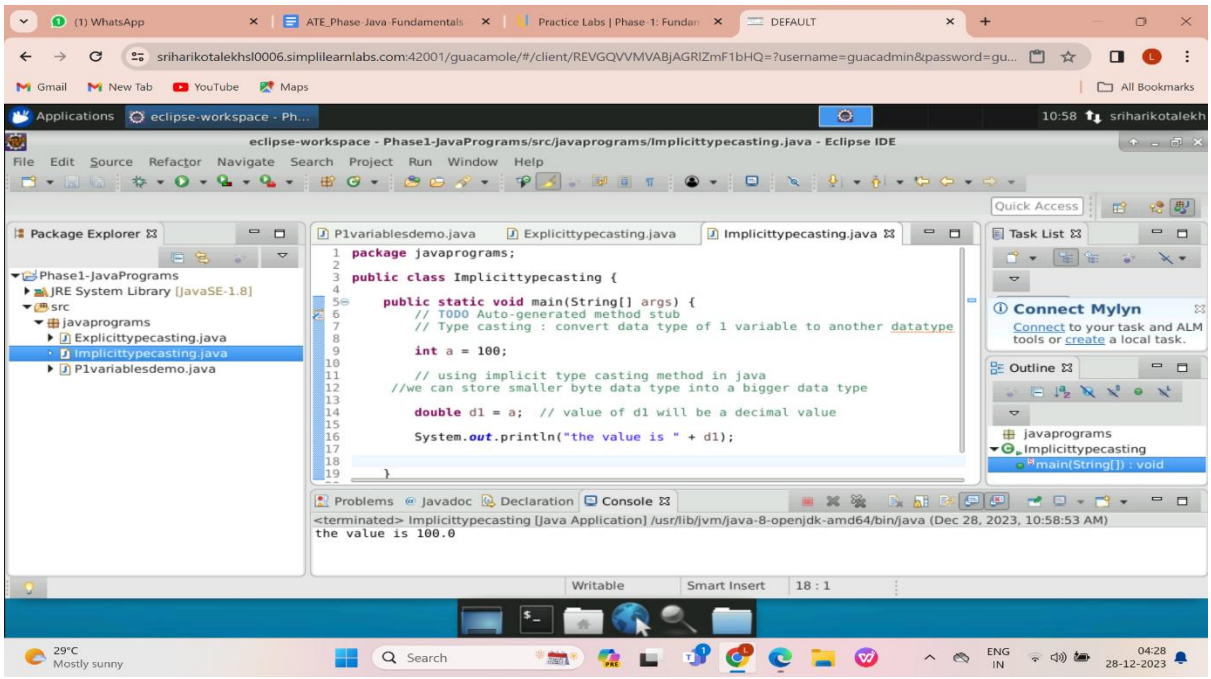
An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc.

(Or)

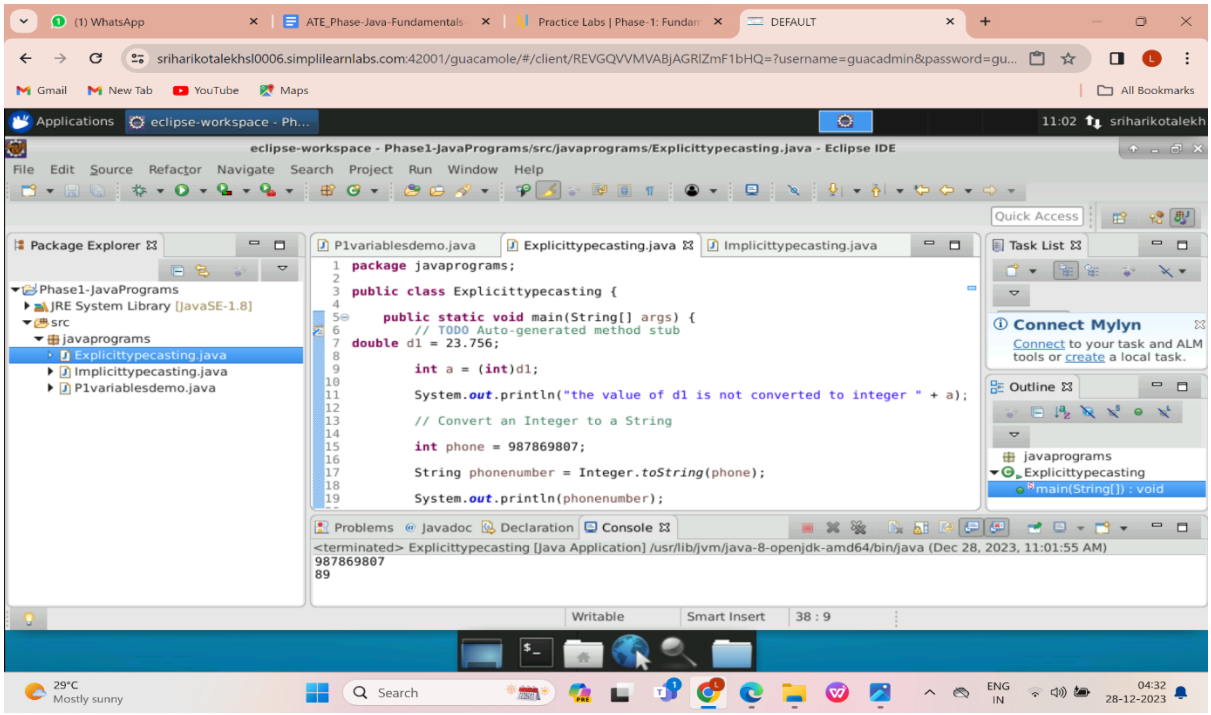
- An object is *a real-world entity*.



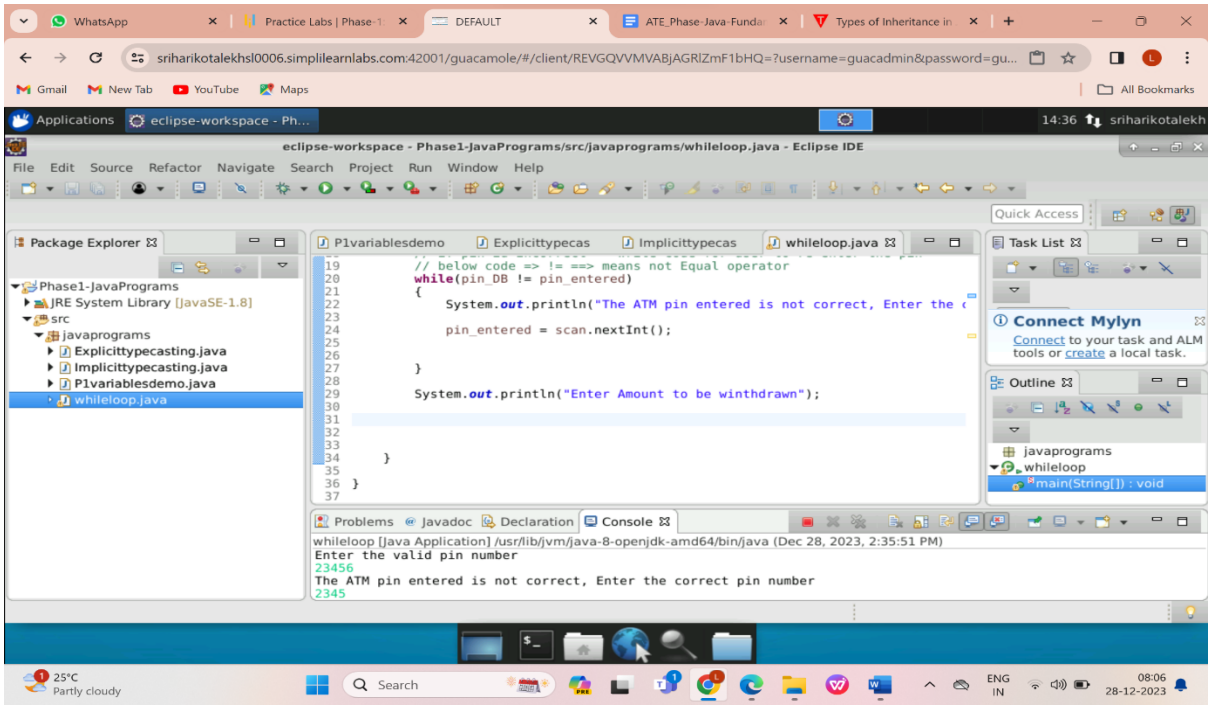
1.Implicit type casting:



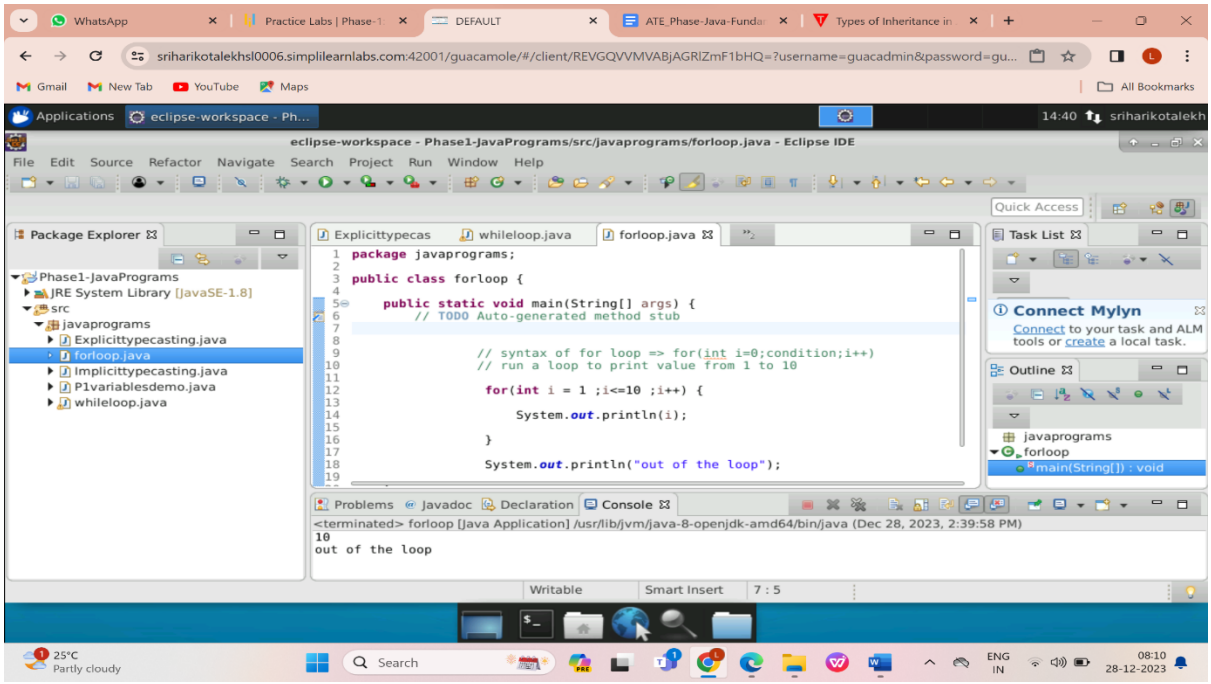
Explicit type casting:


eclipse-workspace - Phase1-JavaPrograms/src/javaprograms/Explicittypecasting.java - Eclipse IDE
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);
<terminated> Explicittypecasting [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Dec 28, 2023, 11:01:55 AM)
987869807
89

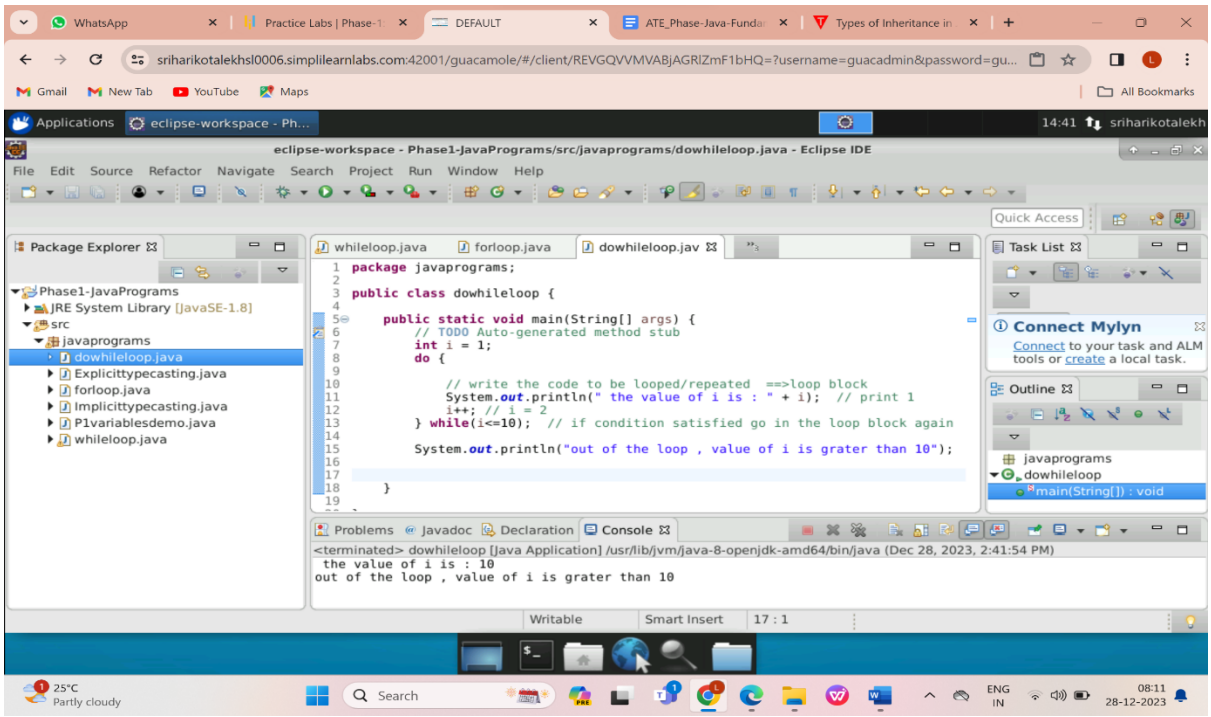
3.While loop:


eclipse-workspace - Phase1-JavaPrograms/src/javaprograms/whileloop.java - Eclipse IDE
// below code => != ==> means not Equal operator
while(pin_DB != pin_entered)
{
System.out.println("The ATM pin entered is not correct, Enter the
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

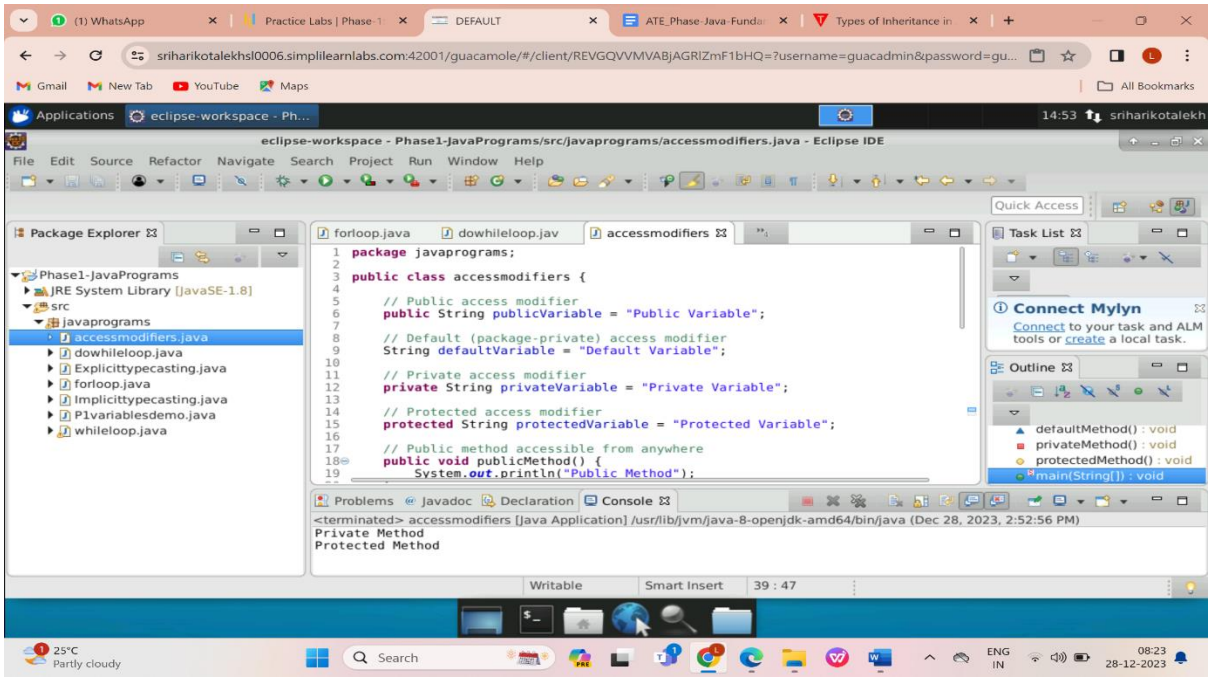
4.for loop:



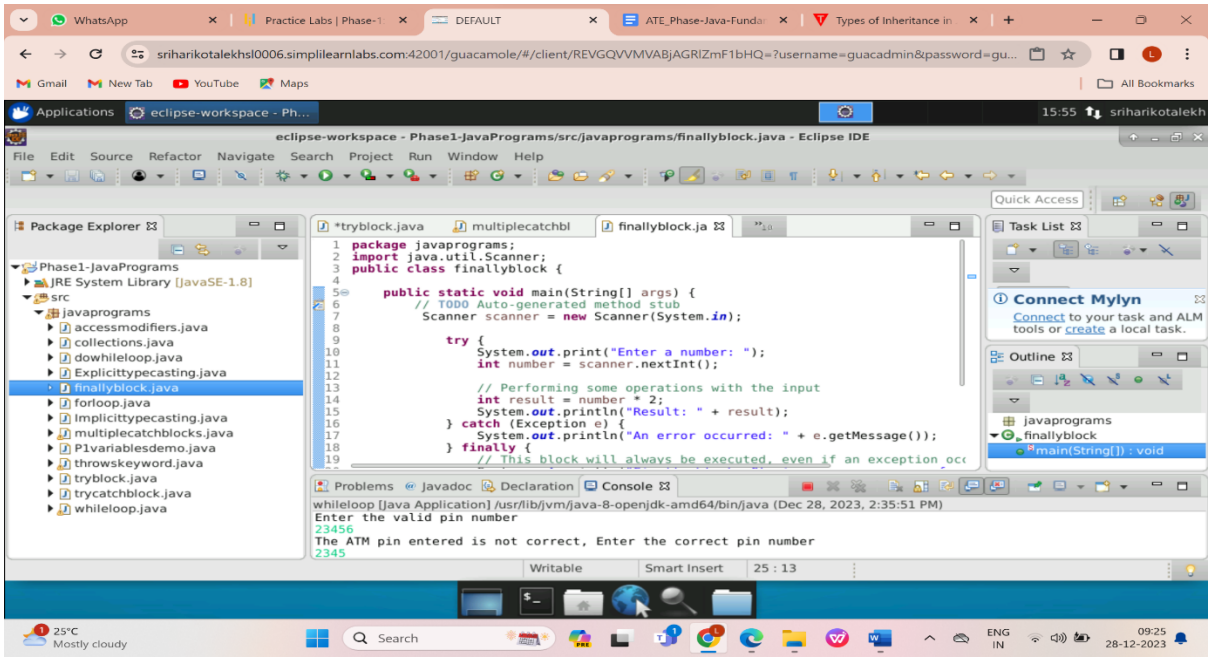
5.do-while loop:



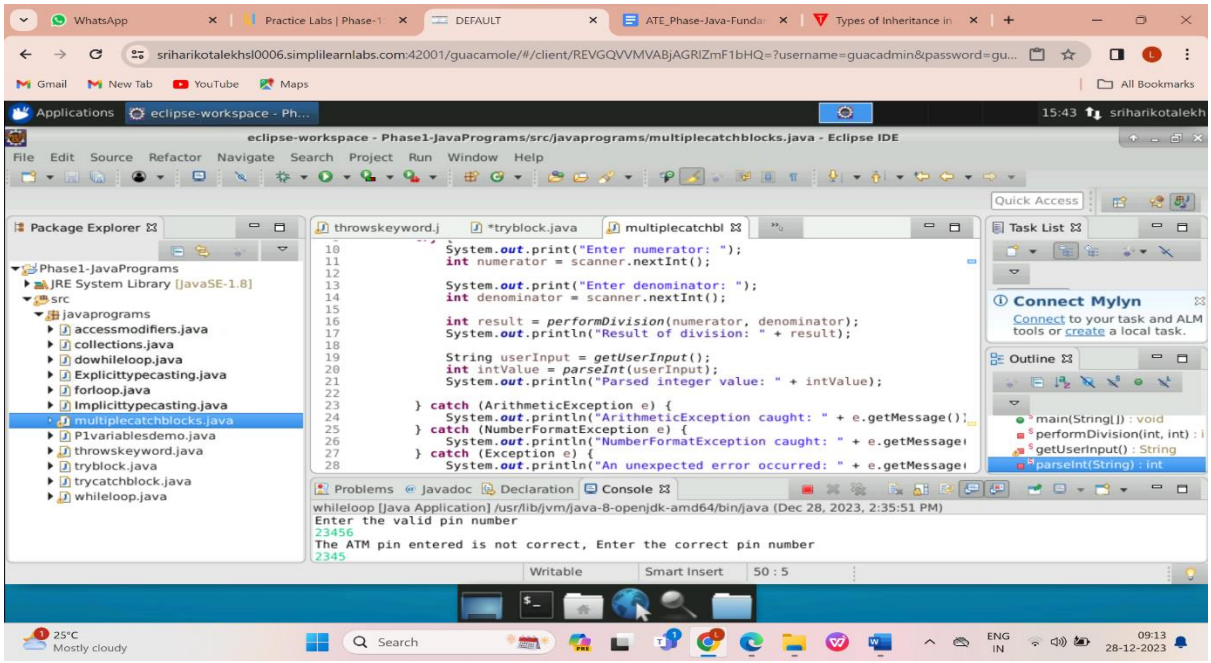
2.Access Modifiers:



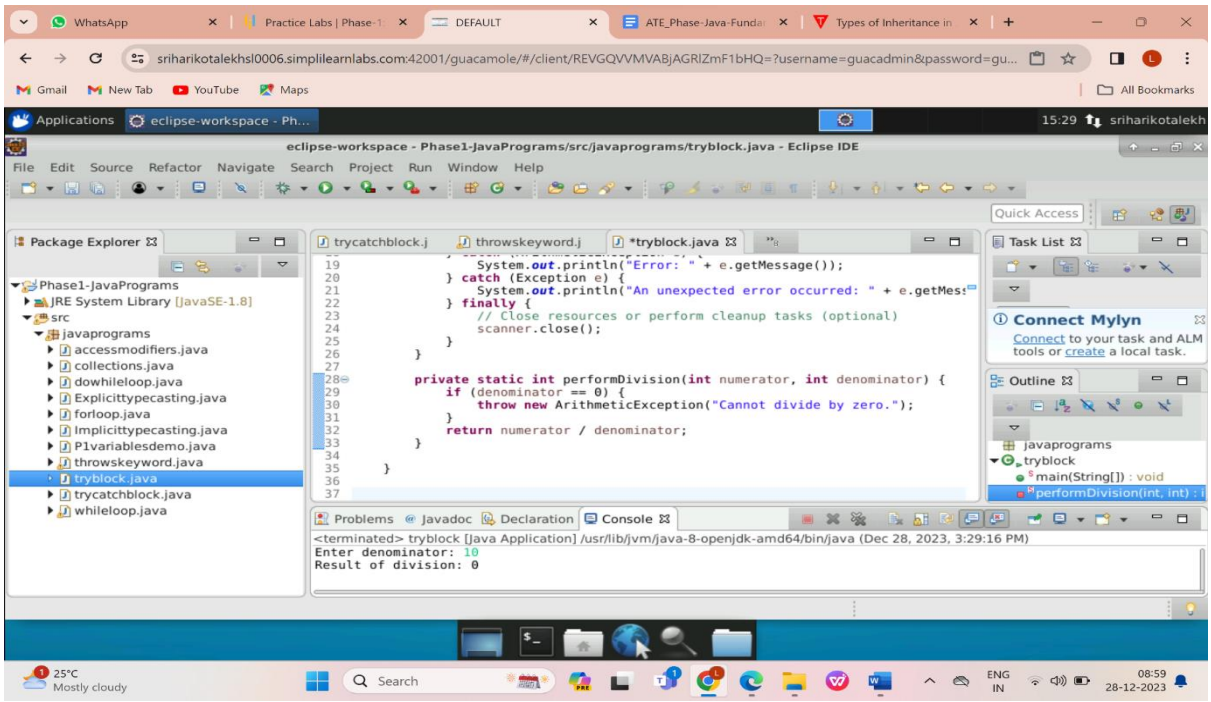
12.finally block:



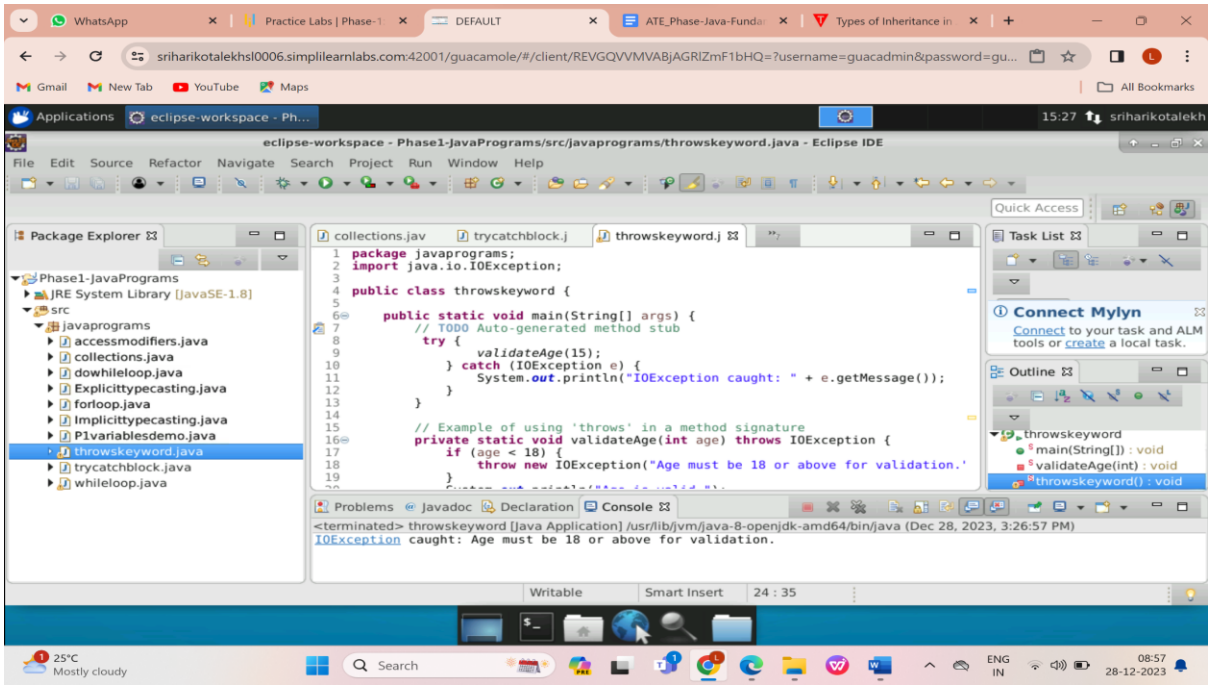
11.Multiple catch blocks:



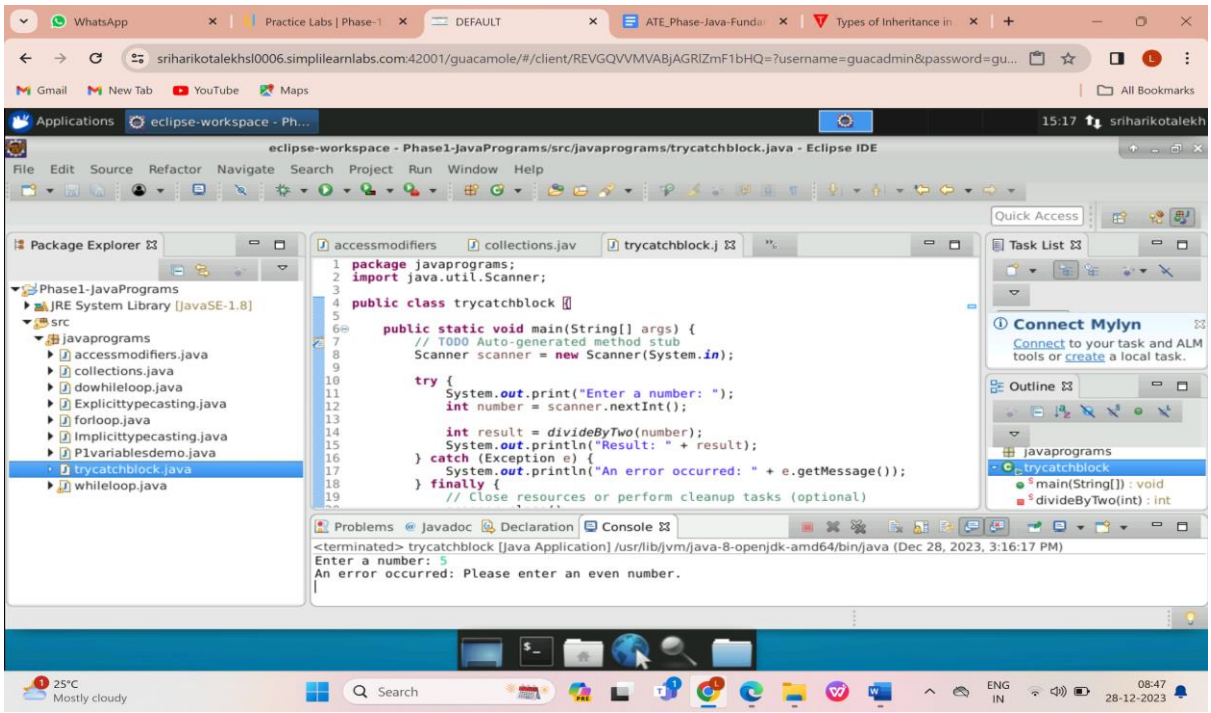
10.Try block:



9.Throws keyword:


eclipse-workspace - Phase1-JavaPrograms/src/javaprograms/throwskeyword.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Phase1-JavaPrograms
JRE System Library [JavaSE-1.8]
src
javaprograms
accessmodifiers.java
collections.java
dowhileloop.java
Explicittypecasting.java
forloop.java
Implicittypecasting.java
P1variablesdemo.java
throwskeyword.java
trycatchblock.java
whileloop.java
collections.jav
trycatchblock.j
throwskeyword.j
1 package javaprograms;
2 import java.io.IOException;
3
4 public class throwskeyword {
5
6 public static void main(String[] args) {
7 // TODO Auto-generated method stub
8 try {
9 validateAge(15);
10 } catch (IOException e) {
11 System.out.println("IOException caught: " + e.getMessage());
12 }
13 }
14
15 // Example of using 'throws' in a method signature
16 private static void validateAge(int age) throws IOException {
17 if (age < 18) {
18 throw new IOException("Age must be 18 or above for validation.'
19 }
Task List
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
throwskeyword
main(String[]) : void
validateAge(int) : void
throwskeyword() : void
Problems Javadoc Declaration Console
<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.
Writable Smart Insert 24 : 35

8.Try catch:


eclipse-workspace - Phase1-JavaPrograms/src/javaprograms/trycatchblock.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Phase1-JavaPrograms
JRE System Library [JavaSE-1.8]
src
javaprograms
accessmodifiers.java
collections.java
dowhileloop.java
Explicittypecasting.java
forloop.java
Implicittypecasting.java
P1variablesdemo.java
trycatchblock.java
whileloop.java
accessmodifiers
collections.jav
trycatchblock.j
1 package javaprograms;
2 import java.util.Scanner;
3
4 public class trycatchblock {
5
6 public static void main(String[] args) {
7 // TODO Auto-generated method stub
8 Scanner scanner = new Scanner(System.in);
9
10 try {
11 System.out.print("Enter a number: ");
12 int number = scanner.nextInt();
13
14 int result = divideByTwo(number);
15 System.out.println("Result: " + result);
16 } catch (Exception e) {
17 System.out.println("An error occurred: " + e.getMessage());
18 } finally {
19 // Close resources or perform cleanup tasks (optional)
Task List
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
javaprograms
trycatchblock
main(String[]) : void
divideByTwo(int) : int
Problems Javadoc Declaration Console
<terminated> trycatchblock [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Dec 28, 2023, 3:16:17 PM)
Enter a number: 5
An error occurred: Please enter an even number.

7.Collections: