# OOPS (Object-Oriented Programming System):

* Object means real word entity. Here we can design a program using a classes and objects.

Types:

1. Class
2. Object
3. Polymorphism
4. Abstraction
5. Encapsulation
6. Inheritance

# Class:

* There are two types of classes in programming predefined and user defined.
* Predefined means is already defined in program we simply use those classes and there functionalities.
* Class is a like a blueprint to create an object.
* Class is a group of similar objects.
* Class does not allow memory when it’s created.
* Class is declared once.

Syntax:

Class demo {

// variables

// methods

}

# Object:

* Object is an instance of a class.
* Object is created **many times** as per requirement.
* Object allow memory when it is created.
* Object is created using new keyword.

Example:

Class demo {

// variables

//method

}

demo d=new demo();

Class name object ref keyword

# Polymorphism:

* The word polymorphism means having many forms.
* In java polymorphism is divided into 2 types….

1. Compile time polymorphism.
2. Run time polymorphism.

Compile time polymorphism:

* Why we call it as compile time polymorphism because at the compile time it self it will decide which method has to call.
* When they are multiple functions with same name in same class but different parameters this functions is called as overloaded functions.
* Functions are overloaded by number of parameters or changing the type of parameters.

Syntax:

public void show (int i){

System.out.println(“first method”);

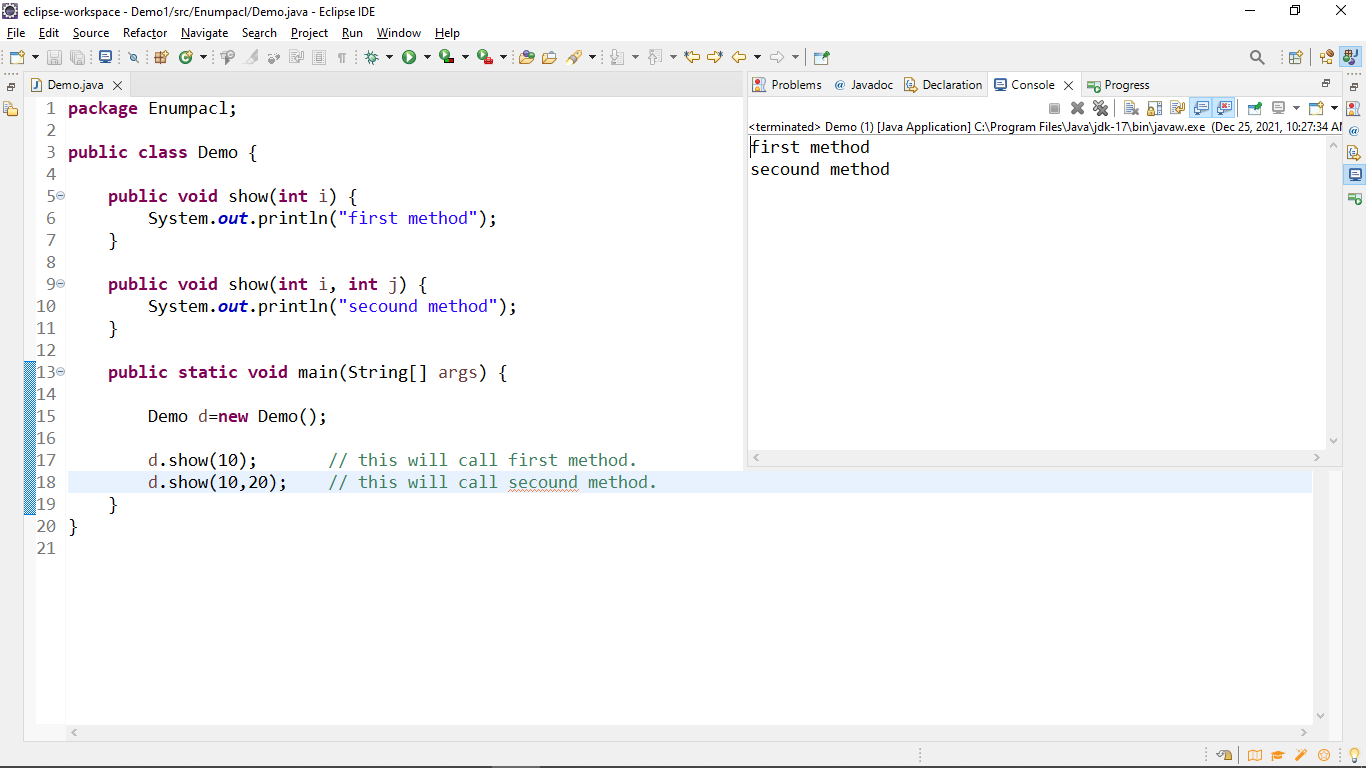
}

public void show (int I, int j){

System.out.println(“second method”);

}

Example:



Explanation:

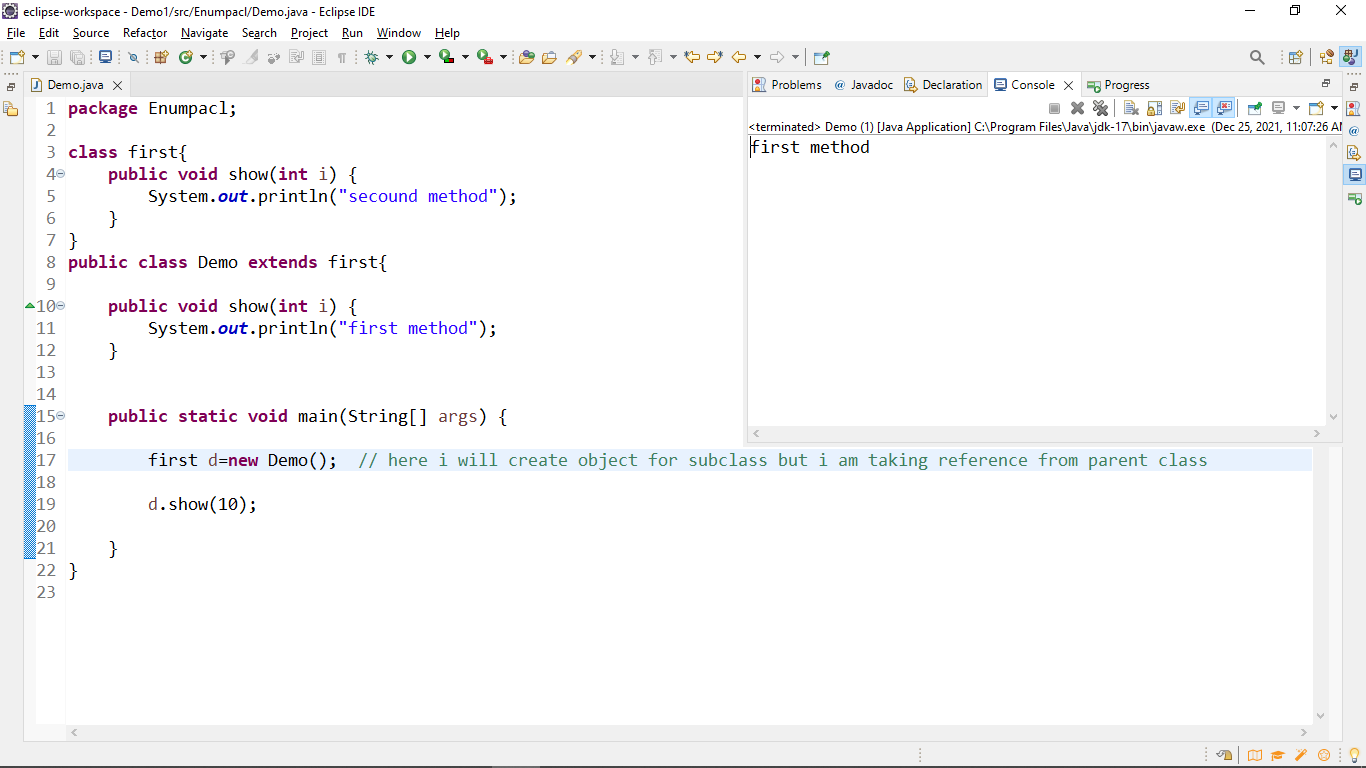
* In above example there is a Demo class have two methods with same name but different parameters.
* When we call a show method with one arguments it call first show method when we call a show method with two arguments it will call second method.

# Run time polymorphism:

* Why we call it as run time polymorphism because it will decide which method have to call in run time because object is created in run time.
* Here we take object reference from super class and we create a object for sub class.

Object reference from super class

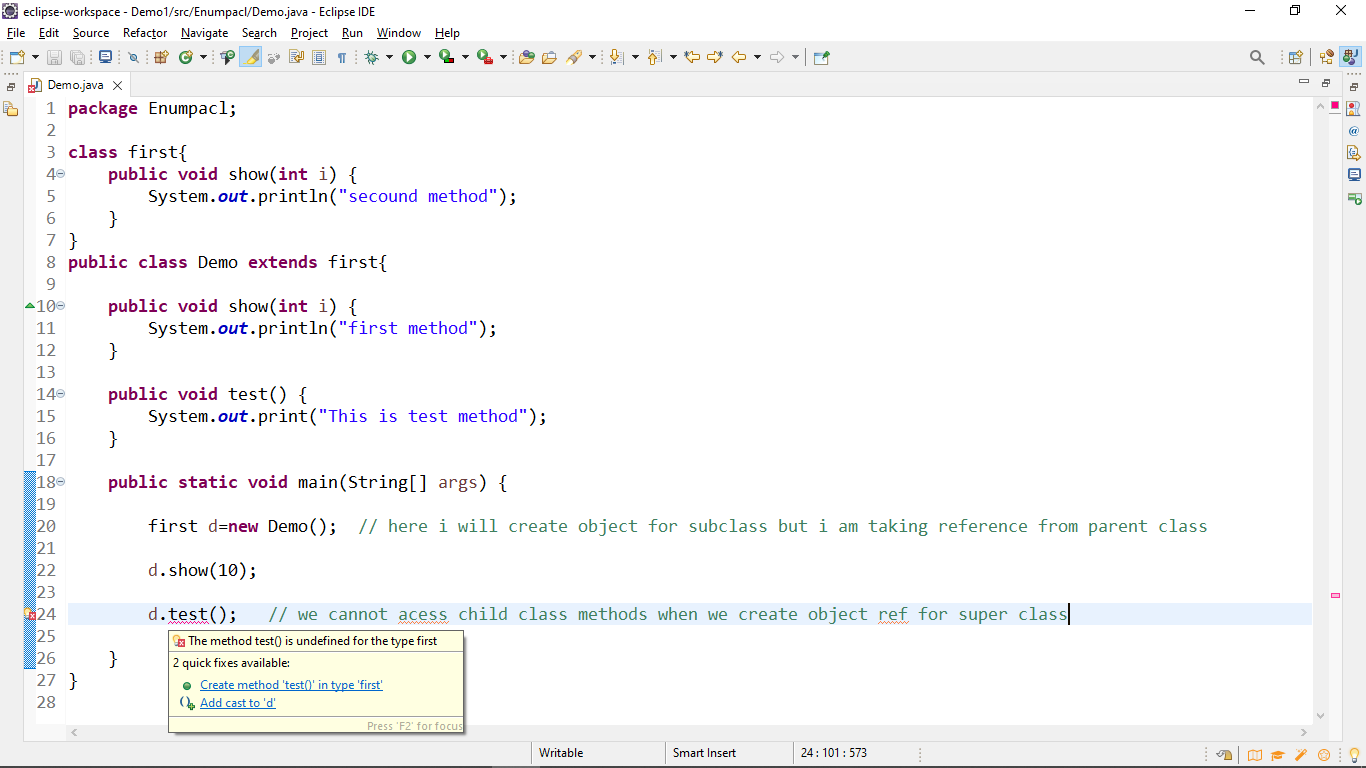
Example:



Explanation:

* In above example they is two classes and second class extends the first class they is a common method child class method can override the super class method.
* When we take a object reference from super class and we create a object for child class we access only methods in super class then child class methods can override the super class methods that’s why we call it also as dynamic –method dispatch.

Example:



Abstraction:

* Abstraction means hiding implementation from user and show only functionality to the user.

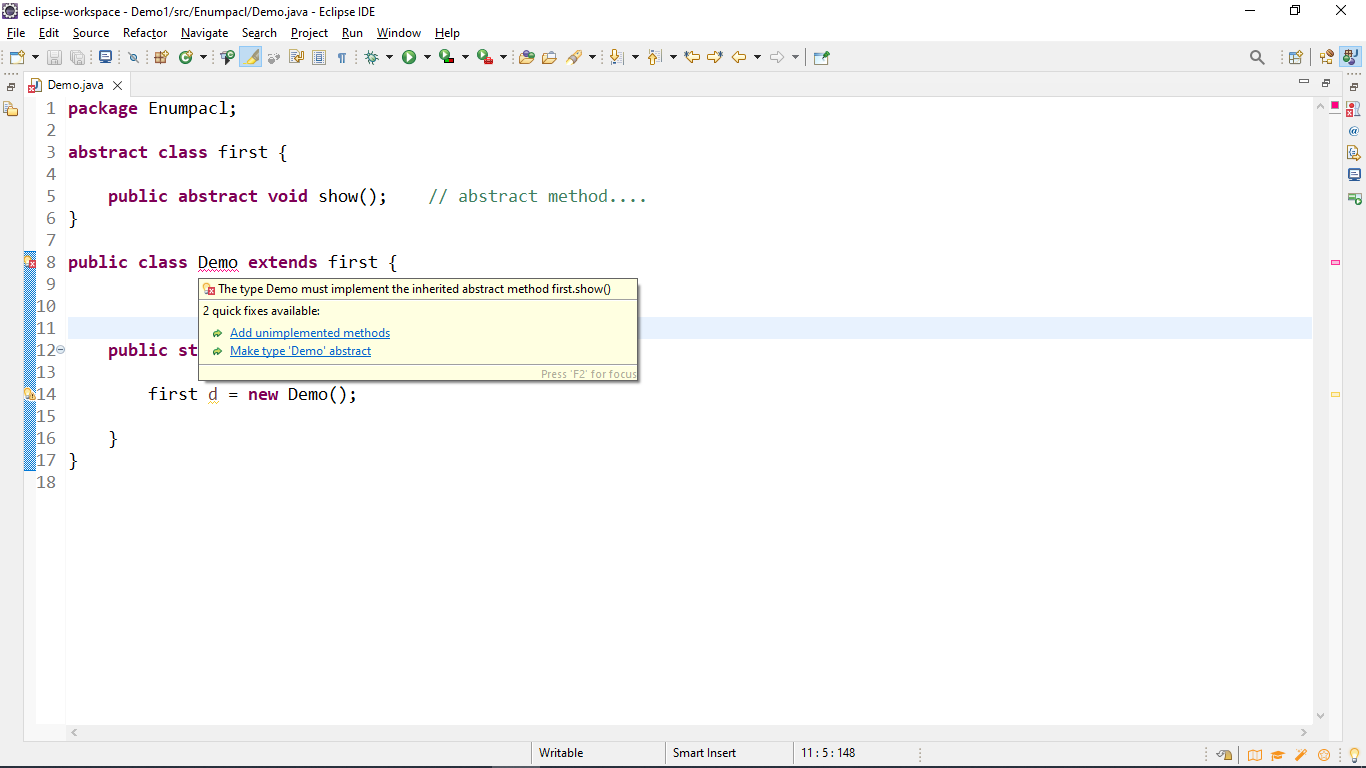
Ex:

We are using Gmail in Gmail backend they is logic for send mails and received mails but we are seeing only functionality ( send , receive mails) not the implementation.

Rules for creating abstract classes and methods:

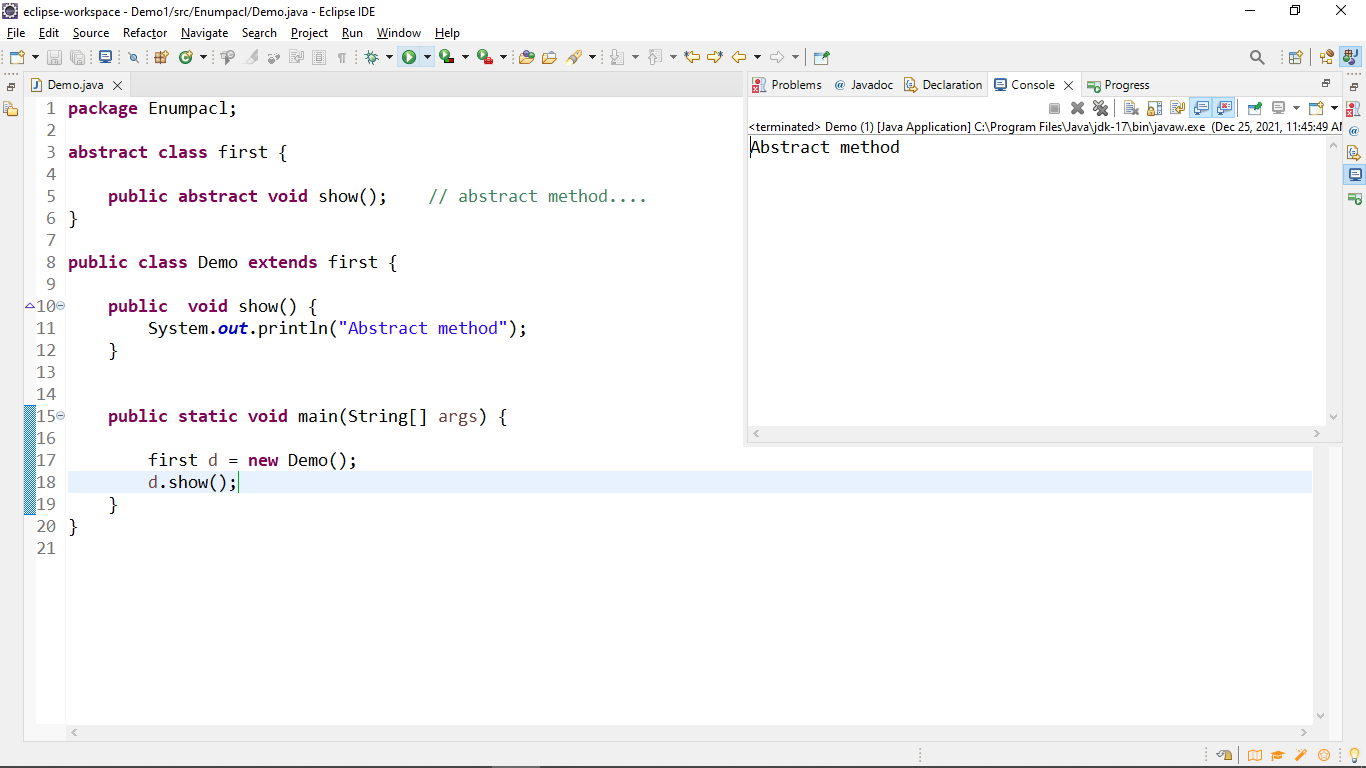
1. Abstract class or abstract methods can declared only with abstract keyword.
2. Abstract classes have static methods and constrictors.
3. We cannot instated abstract class but we take it as reference.
4. Abstract methods don’t have body it just declared and implemented in subclass.

Example:



* In above picture it showing compile time error because if we extends any abstract class they is any abstract method we have to define in subclass otherwise it show compile time error.

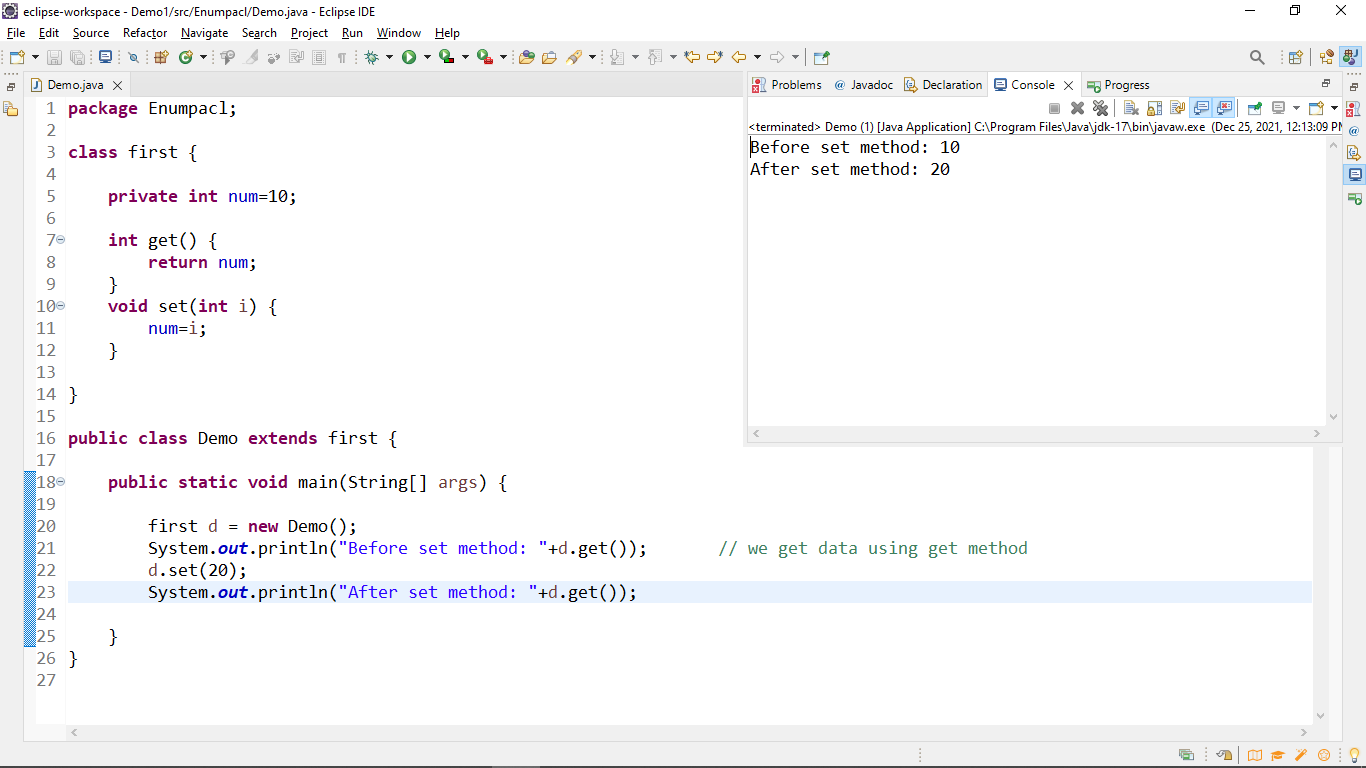
# **Correct way to create abstract classes:**



Encapsulation:

* In encapsulation wrapping a data and methods in a single unit.
* Here data is hidden for other classes and we get and set data using methods it’s also known as data hiding.
* We use private access modifier to hide variables.

Example:



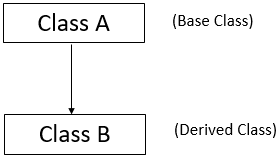
Inheritance:

* **Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object.
* We use extends keyword to archive inheritance.

Types of inheritance in java:

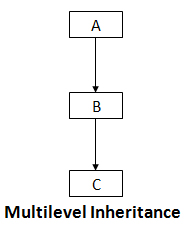
1. single inheritance:

When a class inherits another class, it is known as a single inheritance.



1. Multilevel inheritance:

When there is a chain of inheritance, it is known as multilevel inheritance.



1. Hierarchical inheritance:

When two or more classes inherits a single class, it is known as hierarchical inheritance.

