* 1. Interface
  2. Abstract class
  3. Inheritance
  4. Class and object

**package** Question1;

**interface** Vehicles {

**public** String run(); //in interface method, doesn't required abstract keyword

**public** **int** wheels();

**public** **int** carryPeople();

**public** **float** weight();

/\*void commonParts(){

System.out.println("Common"); //cannot be used like this

}\*/

**default** **public** **void** fun(){

}

}

**package** Question1;

**public** **class** Bus1 **implements** Vehicles{

@Override

**public** String run() {

**return** "Land";

}

@Override

**public** **int** wheels() {

**return** 10;

}

@Override

**public** **int** carryPeople() {

**return** 100;

}

@Override

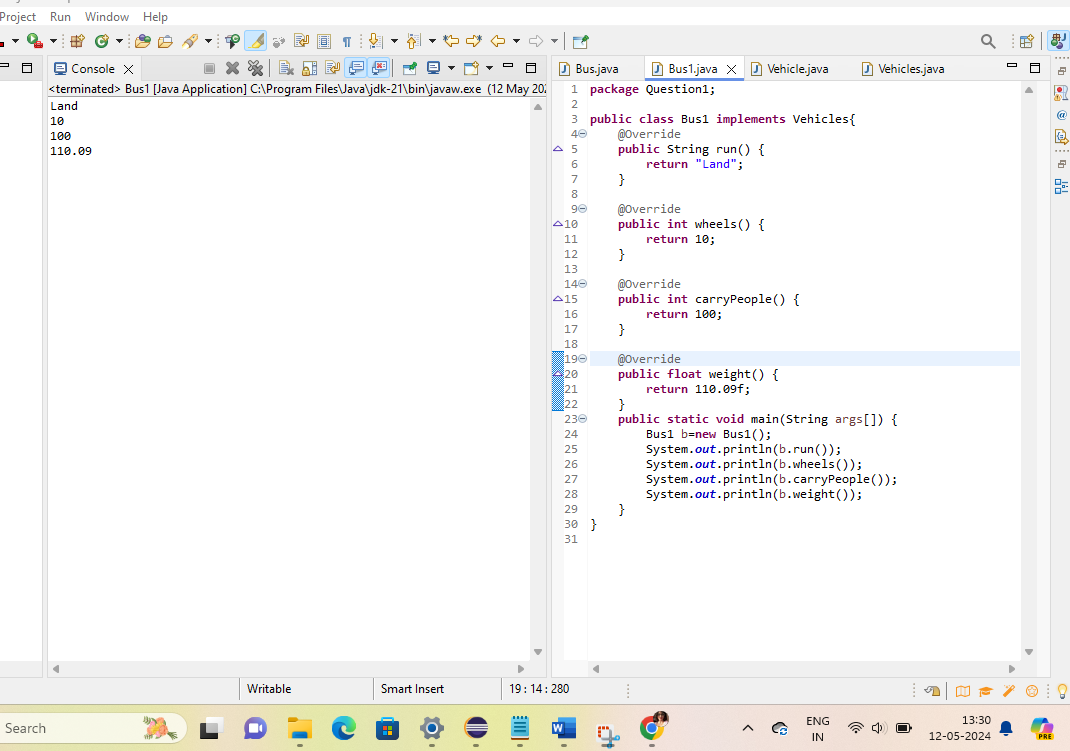
**public** **float** weight() {

**return** 110.09f;

}

}

Output :



Abstract and Inheritence

:

**package** Question1;

**public** **class** Bus **extends** Vehicle { //Bus is a concrete class, because it is a child of vehicle abstract class.

**int** wheels(){

**return** 6;

}

@Override

String run() {

**return** "Land";

}

@Override

**int** carryPeople() {

**return** 100;

}

@Override

**float** weight() {

**return** 100.09f;

}

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

Bus b=**new** Bus();

System.***out***.println(b.wheels());

System.***out***.println(b.run());

System.***out***.println(b.carryPeople());

System.***out***.println(b.weight());

}

}

**package** Question1;

**abstract** **class** Vehicle {

**abstract** String run();

**abstract** **int** wheels();

**abstract** **int** carryPeople();

**abstract** **float** weight(); //abstract functions

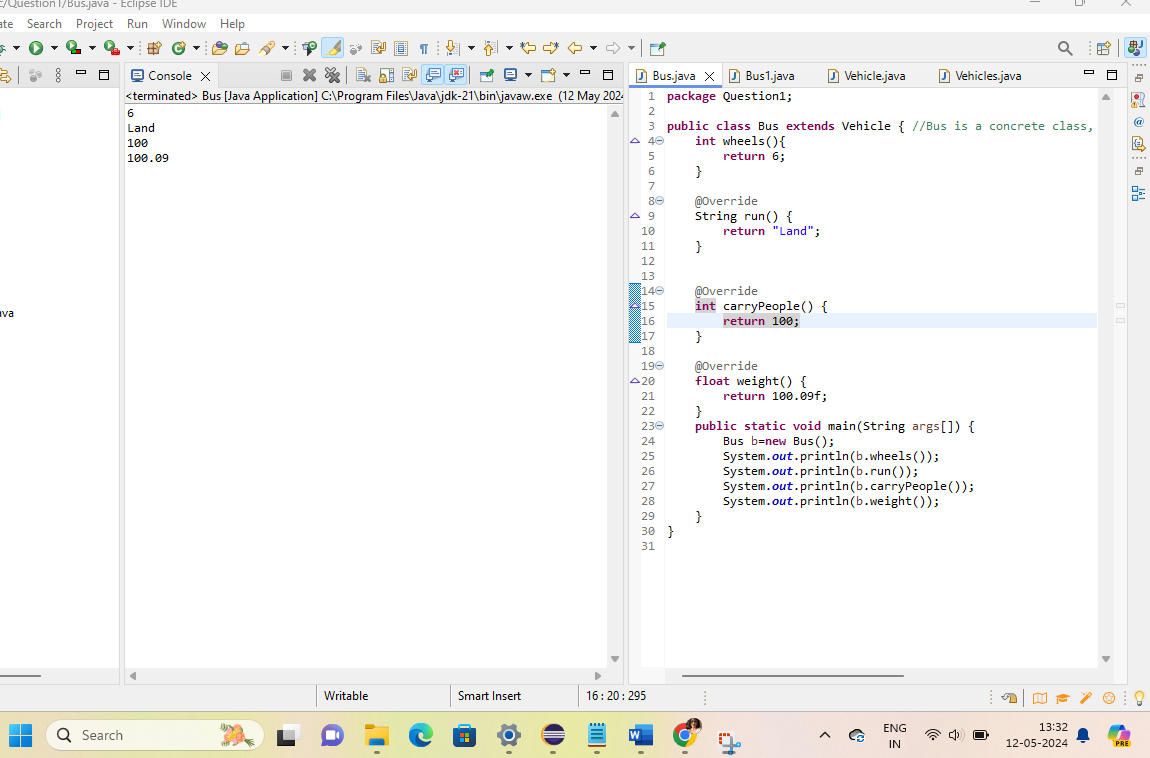
**void** commonParts(){ //non-abstract functions

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

}

}

//abstract class is implemented in Bus and Taxi class file



Constructor:

**package** Question1;

**public** **class** Student {

String name;

**int** roll;

**static** String *schoolName*;

Student(){

System.***out***.println("Inside The Default Constructor");

}

Student(String name, **int** roll, String schoolName){

**this**.name=name;

**this**.roll=roll;

**this**.*schoolName*=schoolName;

}

Student(String schoolName){

**this**.*schoolName*=schoolName;

}

**public** **void** play(){

System.***out***.println("Student love to play");

}

**public** **static** **void** study(){

System.***out***.println("Students love to study");

}

@Override

**public** String toString() {

**return** "Student{" +

"name='" + name + '\'' +

", roll=" + roll +

", schoolName="+ *schoolName* +

'}';

}

}

**package** Question1;

**public** **class** ConstructorTesting {

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

Student stu1= **new** Student("Pooja",100,"DRS"); //refer student.java code for parameterized constructor

System.***out***.println(stu1.name);

System.***out***.println(stu1.roll);

System.***out***.println(stu1.*schoolName*);

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

Student stu2= **new** Student("DRS"); //refer student.java code for parameterized constructor

System.***out***.println(stu2.name);

System.***out***.println(stu2.roll);

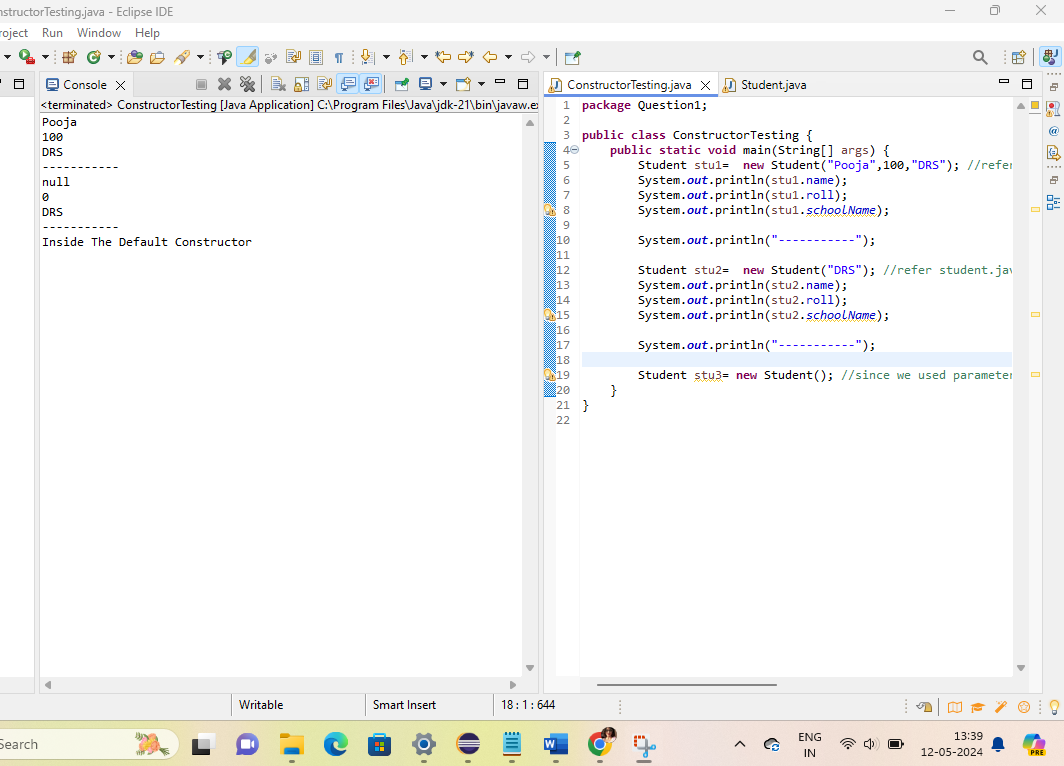
System.***out***.println(stu2.*schoolName*);

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

Student stu3= **new** Student(); //since we used parameterized constructor we should mention default constructor in the parent code.

}

}



Method overloading :

**package** Question1;

**public** **class** methodOverload {

//Java program to demonstrate working of method

//overloading in Java

// Overloaded sum(). This sum takes two int parameters

**public** **int** sum(**int** x, **int** y) { **return** (x + y); }

// Overloaded sum(). This sum takes three int parameters

**public** **int** sum(**int** x, **int** y, **int** z)

{

**return** (x + y + z);

}

// Overloaded sum(). This sum takes two double

// parameters

**public** **double** sum(**double** x, **double** y)

{

**return** (x + y);

}

// Driver code

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

{

methodOverload s = **new** methodOverload();

System.***out***.println(s.sum(10, 20));

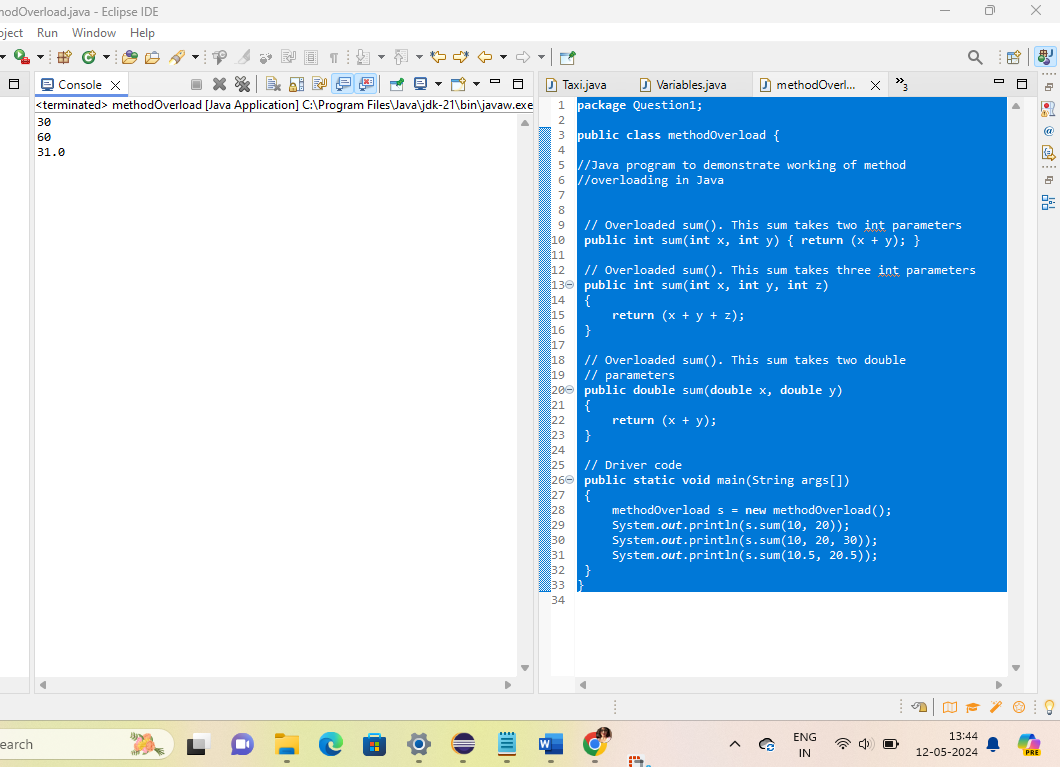
System.***out***.println(s.sum(10, 20, 30));

System.***out***.println(s.sum(10.5, 20.5));

}

}

Output :



Method overriding :

**package** Question1;

//Java program to demonstrate

//method overriding in java

//Base Class

**class** Parent {

**void** show() { System.***out***.println("Parent's show()"); }

}

//Inherited class

**class** Child **extends** Parent {

// This method overrides show() of Parent

@Override **void** show()

{

System.***out***.println("Child's show()");

}

}

//Driver class

**class** Main {

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

{

// If a Parent type reference refers

// to a Parent object, then Parent's

// show is called

Parent obj1 = **new** Parent();

obj1.show();

// If a Parent type reference refers

// to a Child object Child's show()

// is called. This is called RUN TIME

// POLYMORPHISM.

Parent obj2 = **new** Child();

obj2.show();

}

}

Output :

