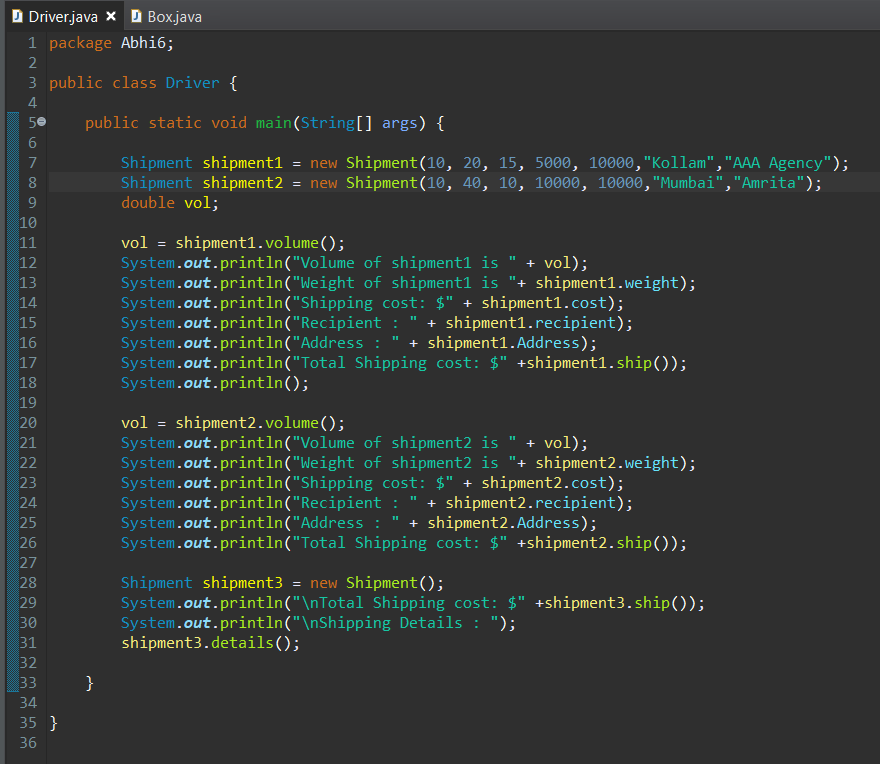
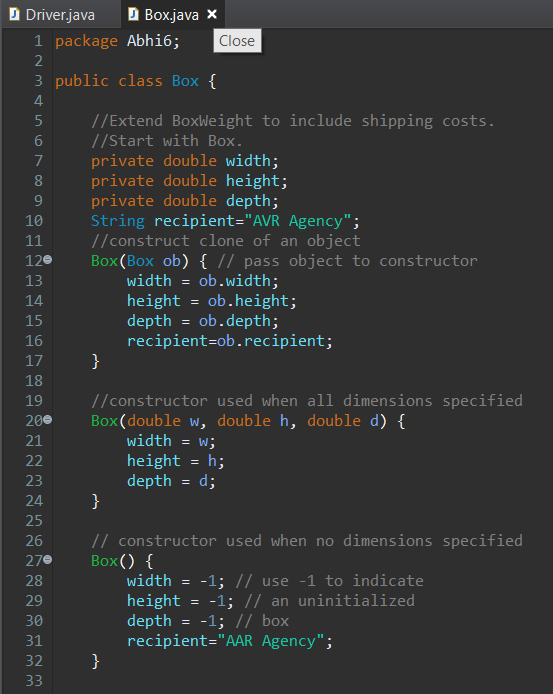
**19CSE204**

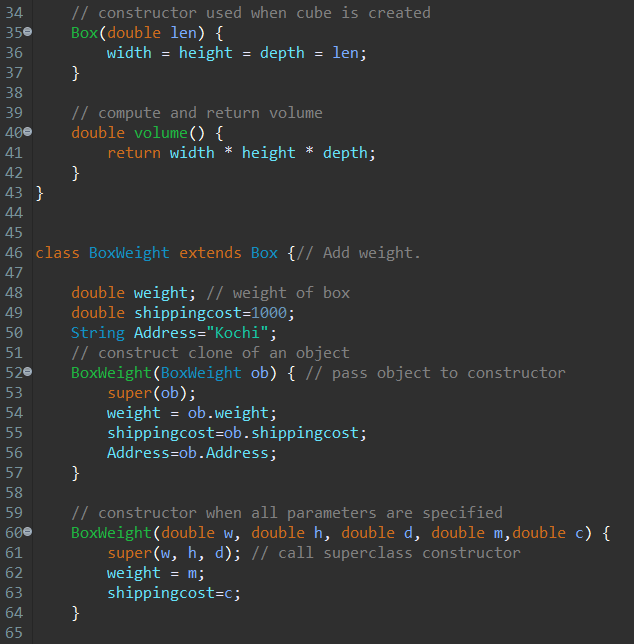
**OBJECT ORIENTED PARADIGM**

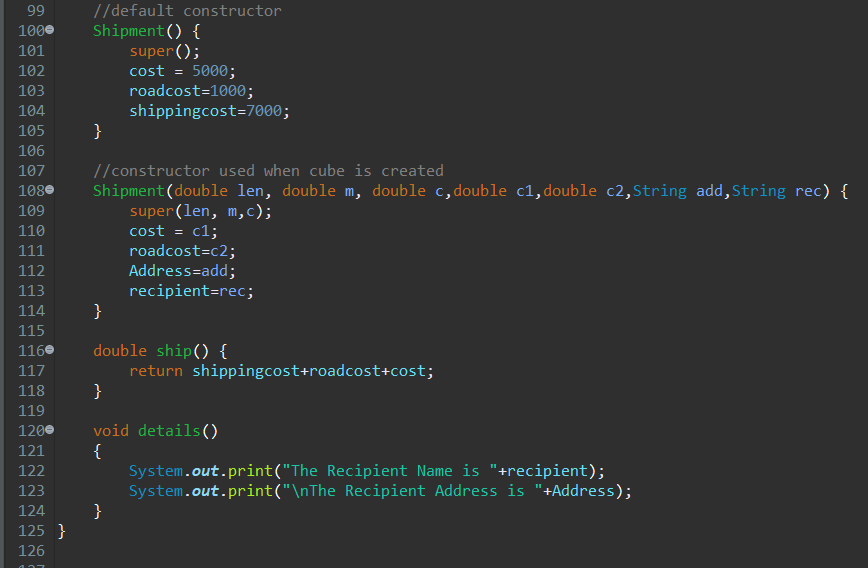
**S. Abhishek**

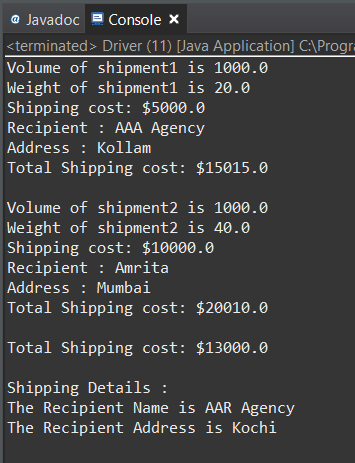
**AM. EN. U4CSE19147**











**Box (Box ob) {**

**// pass object to constructor**

**width = ob.width;**

**height = ob.height;**

**depth = ob.depth;**

**recipient=ob.recipient;**

**}**

This creates the copy of the object and pass it to the constructor which can be later used to pass to the Derived Class.

**Box(double w, double h, double d) {**

**width = w;**

**height = h;**

**depth = d;**

**}**

This is the Parameterized constructor and it is called when three arguments are passed to the constructor when creating the object to invoke the instance of the class.

**// constructor used when no dimensions specified**

**Box() {**

**width = -1; // use -1 to indicate**

**height = -1; // an uninitialized**

**depth = -1; // box**

**recipient="AAR Agency";**

**}**

This is the Default constructor and it is called when no arguments are passed to the constructor when creating the object to invoke the instance of the class.

**// constructor used when cube is created**

**Box(double len) {**

**width = height = depth = len;**

**}**

This is the Parameterized constructor and it is called when only one argument is passed to the constructor when creating the object to invoke the instance of the class.

**double volume() {**

**return width \* height \* depth;**

This method is used to calculate the volume of the Box .

**}**

**class BoxWeight extends Box**

This is used to create the derived class from the parent class and here the parent class is the “Box” and the derived class is the “Boxweight”.

It obtains all the members and member function of the class Box to the class Boxweight.

**BoxWeight (BoxWeight ob) { // pass object to constructor**

This passes the object to the constructor.

**super(ob);**

It extracts the objects from the Parent class and here the “Super” keyword is used to extract the members of the Super class without duplicating the code.

**weight = ob.weight;**

Member of the Boxweight Class

**shippingcost=ob.shippingcost;**

Member of the Boxweight Class

**Address=ob.Address;**

Member of the Boxweight Class

**}**

This class contains its own members and methods in addition to the members and methods extracted from the parent class.

**BoxWeight(double w, double h, double d, double m,double c) {**

**super(w, h, d);**

**// call superclass constructor**

**weight = m;**

**shippingcost=c;**

**}**

This is the Parameterized constructor and it is called when five arguments are passed to the constructor when creating the object to invoke the instance of the class.

It also extracts the objects from the super class to avoid Code Duplication.

**// default constructor**

**BoxWeight()**

**{**

**super();**

**weight = -1;**

**}**

This is the Default constructor and it is called when no arguments are passed to the constructor when creating the object to invoke the instance of the class.

It also extracts the objects from the super class to avoid Code Duplication.

**//constructor used when cube is created**

**BoxWeight(double len, double m,double c)**

**{**

This passes the object to the constructor.

**super(len);**

**weight = m;**

**shippingcost=c;**

**}**

This is the Parameterized constructor and it is called when three arguments are passed to the constructor when creating the object to invoke the instance of the class.

It also extracts the objects from the super class to avoid Code Duplication.

**class Shipment extends BoxWeight {**

This is used to create the derived class from the parent class and here the parent class is the “Boxweight” and the derived class is the “Shipment”.

It obtains all the members and member function of the class Boxweight to the class Shipment.

**Shipment(Shipment ob) {**

**// pass object to constructor**

**super(ob);**

It extracts the objects from the Parent class and here the “Super” keyword is used to extract the members of the Super class without duplicating the code.

**cost = ob.cost;**

Member of the Shipment Class

**roadcost=ob.roadcost;**

Member of the Shipment Class

This class contains its own members and methods in addition to the members and methods extracted from the parent class.

There are two parent class “Box” and “Boxweight”.

**}**

**//constructor when all parameters are specified**

**Shipment (double w, double h, double d, double m, double c, double c1, double c2)**

**{**

This is the Parameterized constructor and it is called when seven arguments are passed to the constructor when creating the object to invoke the instance of the class.

It also extracts the objects from the super class to avoid Code Duplication.

**super(w, h, d, m,c); // call superclass constructor**

**cost = c1;**

**roadcost=c2;**

**}**

**Shipment() { //default constructor**

**super();**

**cost = 5000;**

**roadcost=1000;**

**shippingcost=7000;**

This is the Default constructor and it is called when no arguments are passed to the constructor when creating the object to invoke the instance of the class.

It also extracts the objects from the super class to avoid Code Duplication.

**}**

**//constructor used when cube is created**

**Shipment(double len, double m, double c , double c1,double c2,String add , String rec)**

**{**

This is the Parameterized constructor and it is called when seven arguments are passed to the constructor when creating the object to invoke the instance of the class.

This Constructor is called when the arguments of the same data type are passed.

It also extracts the objects from the super class to avoid Code Duplication.

**super(len, m,c);**

This is used to extract the data and member from the Parent Class.

**cost = c1;**

**roadcost=c2;**

**Address=add;**

**recipient=rec;**

**}**

**double ship()**

**{**

**return shippingcost + roadcost + cost;**

**}**

This method returns the total cost which is extracted from the parent classes “Box” and “Boxweight”.

**void details()**

**{**

**System.out.print("The Recipient Name is "+recipient);**

**System.out.print("\nThe Recipient Address is "+Address);**

**}**

This method returns the Recipient and Address which is extracted from the parent classes “Box” and “Boxweight” respectively.

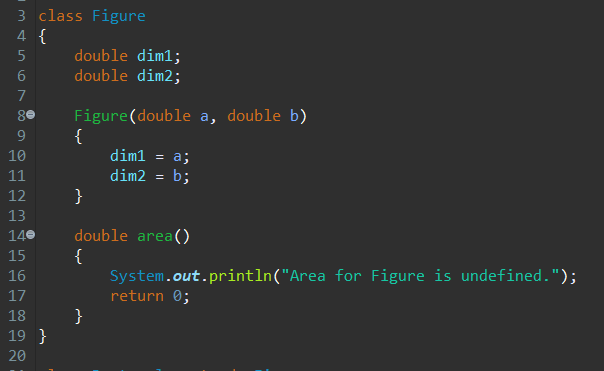
**DRIVER CLASS:**

**Shipment shipment1 = new Shipment (10, 20, 15, 5000, 10000,"Kollam","AAA Agency");**

This calls the Parameterized constructor.

**Shipment shipment2 = new Shipment (10, 40, 10, 10000, 10000,"Mumbai","Amrita");**

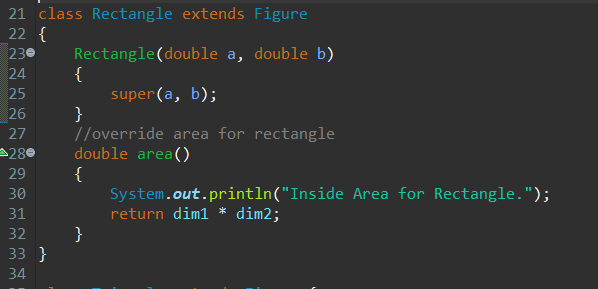
This calls the Parameterized constructor.

****

Here “Figure” is a class with two data members “dim1” and “dim2”. It also acts as a parent class or base class.

It contains a parameterized constructor with two arguments and then assigning the values of the arguments to the data members.

It contains the method “area()” which prints “Area For Figure is undefined” when it is called.

****

Here “Rectangle” is a class which inherits from the parent class “Figure”.

It also known as a sub class or derived class.

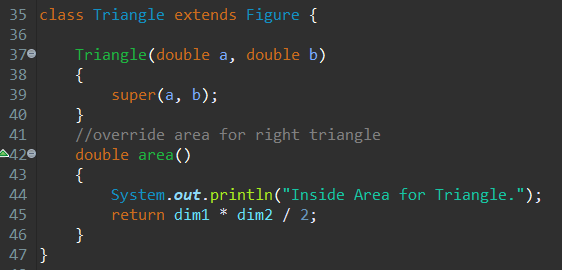
It contains a parameterized constructor with two arguments and those two arguments are extracted from its parent class using the “super” key word.

It contains the method “area()” which prints “Inside Area For Rectangle.” when it is called and returns the multiplication of two arguments.

Here this area overrides the existing area of the parent class and it displays the different information. This is because of “Method Overriding.”

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding.

When a method in a subclass has the same name, same parameters or signature, and same return type as a method in its super class, then the method in the subclass is said to override the method in the super class.



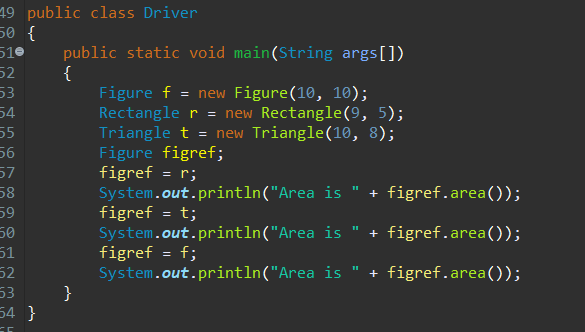
Here “Triangle” is a class which inherits from the parent class “Figure”.

It also known as a sub class or derived class.

It contains a parameterized constructor with two arguments and those two arguments are extracted from its parent class using the “super” key word.

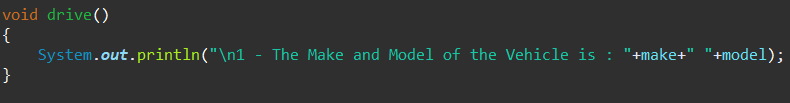
It contains the method “area()” which prints “Inside Area For Triangle.” when it is called and returns the area of the Triangle.

Here this area overrides the existing area of the parent class. This is because of “Method Overriding”.

`

Here we are creating three different objects for three different classes and calling the parameterized constructor and assigning the values to the members of the constructor.

We are calling the “area()” member function with three different objects and the calculation happening in the three different member function is different.

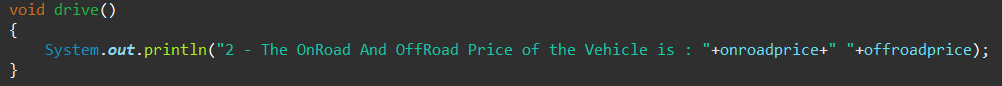


This is the member function of the Class Vehicle which is the Super class or the Parent class and the method is defined in the class.

Here we are going to illustrate the method over riding.

To do that we need to have the same Signature (ie) Data types, same method names and same return types, same arguments.

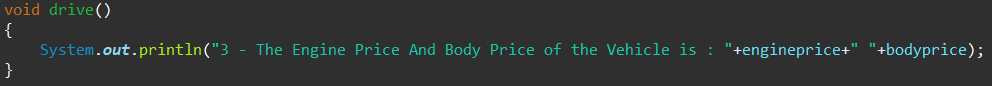
Only the content inside the method can change.



This is the method inside the class Road which extends to the class vehicle and hence it’s the sub class of the parent class “Vehicle”.

We have the same Signature (ie) Data types, same method names and same return types, same arguments.

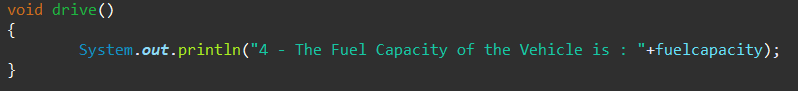
Only the content inside the method is changing.



This is another method inside the class “Fourwheel” which extends to the class vehicle and hence it’s the sub class of the parent class “Road”.

We have the same Signature (ie) Data types, same method names and same return types, same arguments.

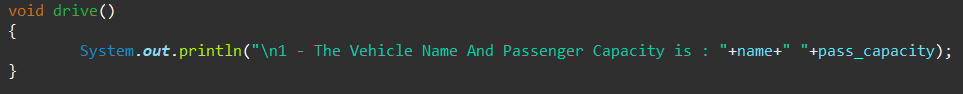
Only the content inside the method is changing.



This is another method inside the class “Twowheel” which extends to the class vehicle and hence it’s the sub class of the parent class “Fourwheel”.

We have the same Signature (ie) Data types, same method names and same return types, same arguments.

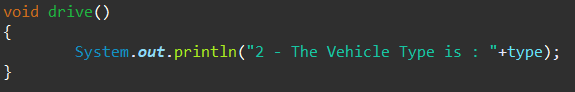
Only the content inside the method is changing.



This is another method inside the class “WaterBourne” which extends to the class vehicle and hence it’s the sub class of the parent class “Vehicle”.

We have the same Signature (ie) Data types, same method names and same return types, same arguments.

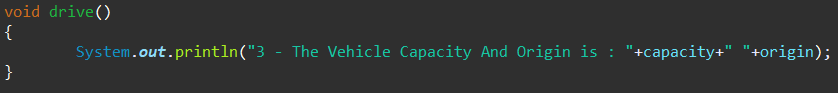
Only the content inside the method is changing.



This is another method inside the class “OceanVessel” which extends to the class vehicle and hence it’s the sub class of the parent class “Waterbourne”.

We have the same Signature (ie) Data types, same method names and same return types, same arguments.

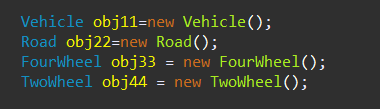
Only the content inside the method is changing.



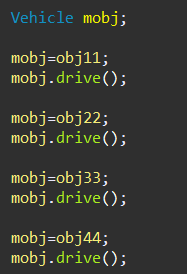
This is another method inside the class “CoastVessel” which extends to the class vehicle and hence it’s the sub class of the parent class “OceanVessel”.

We have the same Signature (ie) Data types, same method names and same return types, same arguments.

Only the content inside the method is changing.



This is creating the Object of various classes to invoke their members and member functions.



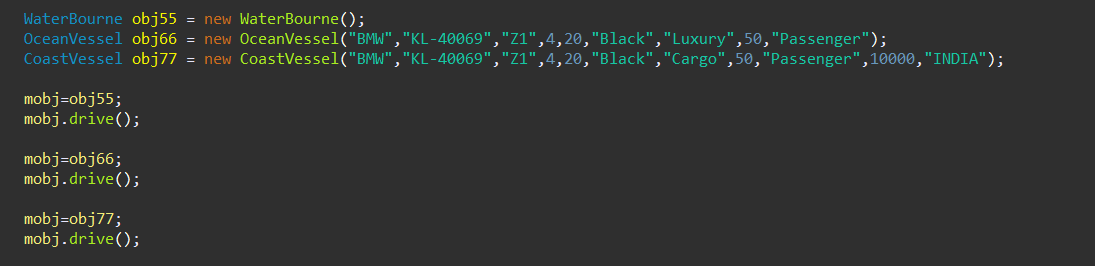
The first line is creating the object “mobj” with reference to the parent class.

The second and third line is representing the object “mobj” to the class “Vehicle” and calling the method “drive()”.

The Fourth and fifth line is representing the object “mobj” to the class “Road” and calling the method “drive()”.

The Sixth and Seventh line is representing the object “mobj” to the class “FourWheeler” and calling the method “drive()”.

The Eighth and Ninth line is representing the object “mobj” to the class “TwoWheeler” and calling the method “drive()”.



This is creating the Object of various classes to invoke their members and member functions by calling the Default Constructor in First Line and Parameterized Constructor in Second and Third Line.

The fifth line and sixth line is representing the object “mobj” to the class “WaterBourne” and calling the method “drive()”.

The Seventh and Eighth line is representing the object “mobj” to the class “OceanVessel” and calling the method “drive()”.

The Ninth line and tenth line is representing the object “mobj” to the class “Coastvessel” and calling the method “drive()”.

**THANKYOU!!**