1. **Inheritance**

   Here, properties of a class (super class/ parent class) to be inherited by another class (sub class / child class). Various types of inheritance are single, multilevel, hierarchical, hybrid. Multiple inheritance is not supported in Java. Inheritance supports code reuse.

   Task TO DO:

   - Modify the given example to form an example of hybrid inheritance. See what happens?
   - Show the use of **instanceof** Operator?
   - How is protected access modifier useful in inheritance?

2. **Polymorphism:**

   Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance. Overloading exhibits compile-time polymorphism , while Overriding shows runtime Overloading.

3. **Overloading:**

   The same function name can be used for different method(with different argument sets). In the program below, overload the add() function for adding the combination of integer and double numbers as shown in the main function.

4. **Simple Overriding:**

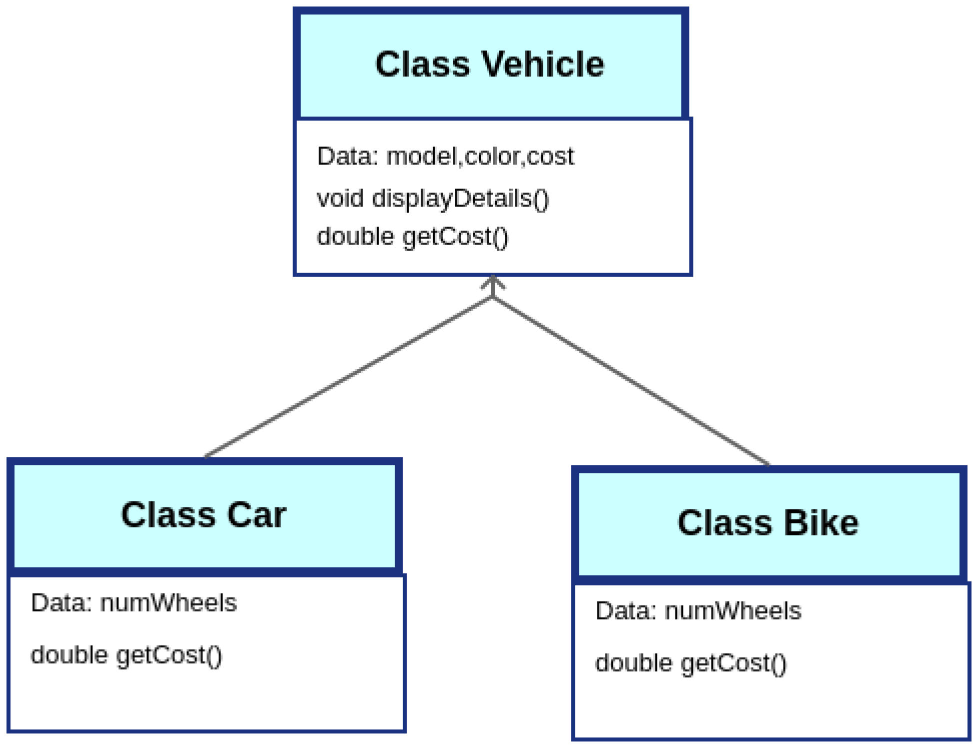
   In a class hierarchy when a method in a sub-class has the same name and type signature as in a method in its super class, then the method in the sub-class is said to override the method in the super class.

5. **Overriding with runtime polymorphism (Dynamic Method Dispatch):**

   - Upcasting: A call is made to an overridden method of child class via its parent type reference.
   - During Upcasting, the type of the object determines which method to be invoked. Making of this decision happens during runtime (by JVM) is called as Run time polymorphism.
   - Each time during the call of show() method, which method gets executed (Maps, WebMaps or PaperMaps) depends on the type of the object being referred to at the time of the call (i.e. during runtime).

```java
1  class Maps{
2  
3      void show()
4      {
5          System.out.println("Maps");
6      }
7  }
8  
9  class WebMaps extends Maps {
10     void show()
11     {
12         System.out.println("Web Maps: Bing Map, Google Map");
13     }
14 }
15 
16 
17 class  PaperMaps extends Maps {
18     void show()
19     {
20         System.out.println("Paper Maps: All Maps printed on paper");
21     }
22 }
23 
24 public class RunTimePolyDemo{
25 
26     public static void main (String[] args) {
27 
28         // We can have references of Maps type.
29         Maps m = new Maps();
30         WebMaps w = new WebMaps();
31         PaperMaps p = new PaperMaps();
32 
33         m = w ; // upcasting
34         m.show();
35 
36         m = p; // upcasting
37         m.show();
38         // the show() method is called via the reference variable of the
39         //Parent class. Since it refers to the subclass object and subclass
40         // method overrides the Parent class method, the subclass method
41         // is invoked at runtime.
42         //Each time during the call of show() method (e.g line 35 or line 38 ),
43         // which method gets executed (show() mehtod in Maps, WebMaps or PaperMaps)
44         // depends on the type of the object being referred to at the time of the
45         // call (i.e. during runtime):
46     }
47 }
```

Consider the figure below, there are there classes. Implement inheritance, function overloading (displayDetails() method) , and overriding (getCost() method).


Class Vehicle
Data: model,color,cost
void displayDetails()
double getCost()
Class Car
Data: numWheels
double getCost()
Class Bike
Data: numWheels
double getCost()