# Polymorphism

The dictionary definition of polymorphism: From *http://www.thefreedictionary.com/polymorphism*

**polymorphism** [ˌpɒlɪˈmɔːfɪzəm]

*n*

1. (Life Sciences & Allied Applications / Biology) *Biology*
   - **a.** the occurrence of more than one form of individual in a single species within an interbreeding population
   - **b.** the occurrence of more than one form in the individual polyps of a coelenterate colony
2. (Chemistry) the existence or formation of different types of crystal of the same chemical compound



**polymorphism** (pŏl'ē-môr'fĭz'əm)

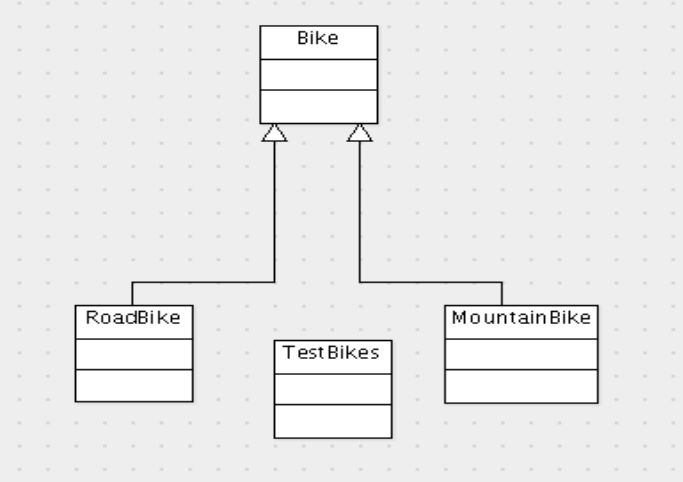
**1.** The existence of two or more different forms in an adult organism of the same species, as of an insect. In bees, the presence of queen, worker, and drone is an example of polymorphism. Differences between the sexes and between breeds of domesticated animals are not considered examples of polymorphism.

**2.** The crystallization of a compound in at least two distinct forms. Diamond and graphite, for example, are polymorphs of the element carbon. They both consist entirely of carbon but have different crystal structures and different physical properties.



This principle can also be applied to object-oriented programming and languages like the Java language. Subclasses of a class can define their own unique behaviors and yet share some of the same functionality of the parent class.

**Example:** Consider the following java classes.




```
class Bike {
   public String name(){
         return "Bike";
   }
}

class MountainBike extends Bike {
   public String name(){
         return "MountainBike";
   }
}

class RoadBike extends Bike {
   public String name(){
         return "RoadBike";
   }
}

class TestBike {

   public String getName(Bike b){
         return b.name();
   }

   public static void main(String[] args){
         Bike b1= new RoadBike();
         Bike b2= new MountainBike();
         TestBike tb= new TestBike();
         System.out.println(tb.getName(b1));
         System.out.println(tb.getName(b2));
   }
}
```

When getName() method is invoked, the Java virtual machine calls the appropriate method for the object that is referred to in each variable b1 and b2. It does not call the method that is defined by the variable's type, which is Bike. This behavior is referred to as virtual method invocation and demonstrates an aspect of the important *polymorphism* features in the Java language.

When getName() method is invoked, the Java virtual machine calls the appropriate method for the object that is referred to in each variable b1 and b2. It does not call the method that is defined by the variable's type, which is Bike. This behavior is referred to as virtual method invocation and demonstrates an aspect of the important polymorphism features in the Java language.