**OOP: Polymorphism in Java**

The ability of an object to take many forms.

We can create polymorphic objects when we have a parent-child relationship between classes or interfaces and their implementations.

Let’s say we have these classes

CAR

AUDI

BMW

Car class is parent class of BMW and Audi

We can create objects like this

Car car = new BMW();

// or

Car car2 = new Audi();

So you can see how Car object can be BMW and Audi as well. That’s why we say the ability of an object to take many forms.

Car car =

This part is a reference type. We always need to remember that reference type will decide what methods and properties available for the object.

= new BMW();   
= new Audi();

This is the actual object part. If BMW or Audi classes override the methods from the Car class. During the runtime, the overridden method will get called.

The polymorphism with interfaces and abstract classes will work exactly the same.

public class Car {  
 public String model;  
 public double sdeed;

public void drive() {  
 System.out.println("Car is driving");  
 }

public void stop() {  
 System.out.println("Car is stopping");  
 }

}

and let’s create a BMW class.

public class BMW extends Car {  
 @Override  
 public void drive() {  
 System.out.println("BMW is driving");  
 }

public void driveFast() {  
 System.out.println("BMW is driving fast");   
 }  
}

and the way we use

public class Main {  
 public static void main(String[] args) {  
 Car car = new BMW();

car.drive(); // BMW is driving  
   
 car.stop(); // Car is stopping

// car.driveFast(); does not compile  
 }  
}

* car.drive(); will print BMW is driving because the BMW class overrides the parent method. This is also known as runtime polymorphism because java thinks it will execute the method of the Car class but during the runtime, it will figure out that BMW overrides it and will execute BMW’s method.
* car.stop(); the output will be Car is stopping the parent class method will get called.
* car.driveFast(); will compile because. The reference type decides what methods are available for car the object. And Car does not have driveFast(); method.