

Introduction

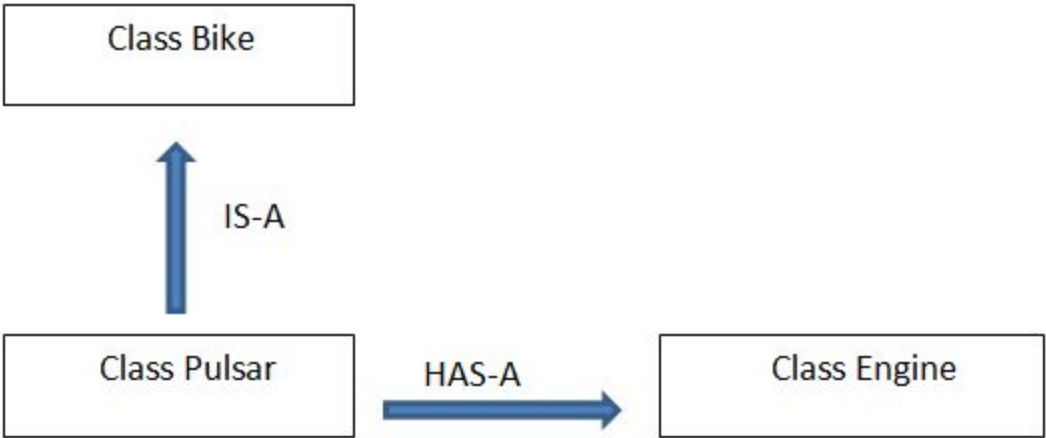
In Java, we can reuse our code using an Is-A relationship or using a Has-A relationship. An Is-A relationship is also known as inheritance and a Has-A relationship is also known as composition in Java.

Is-A Relationship in Java

In Java, an Is-A relationship depends on inheritance. Further inheritance is of two types, class inheritance and interface inheritance. It is used for code reusability in Java. For example, a Potato is a vegetable, a Bus is a vehicle, a Bulb is an electronic device and so on. One of the properties of inheritance is that inheritance is unidirectional in nature. Like we can say that a house is a building. But not all buildings are houses. We can easily determine an Is-A relationship in Java. When there is an extends or implement keyword in the class declaration in Java, then the specific class is said to be following the Is-A relationship.

Has-A Relationship in Java

In Java, a Has-A relationship is also known as composition. It is also used for code reusability in Java. In Java, a Has-A relationship simply means that an instance of one class has a reference to an instance of another class or an other instance of the same class. For example, a car has an engine, a dog has a tail and so on. In Java, there is no such keyword that implements a Has-A relationship. But we mostly use new keywords to implement a Has-A relationship in Java.



Example

```
01. package relationsdemo;
02. public class Bike
03. {
04.     private String color;
05.     private int maxSpeed;
06.     public void bikeInfo()
07.     {
08.         System.out.println("Bike Color= "+color + " Max Speed= " + maxSpeed);
09.     }
10.     public void setColor(String color)
11.     {
12.         this.color = color;
13.     }
14.     public void setMaxSpeed(int maxSpeed)
15.     {
16.         this.maxSpeed = maxSpeed;
17.     }
18. }
```

In the code above the Bike class has a few instance variables and methods.

```
01. package relationsdemo;
02. public class Pulsar extends Bike
03. {
04.     public void PulsarStartDemo()
05.     {
06.         Engine PulsarEngine = new Engine();
07.         PulsarEngine.stop();
08.     }
09. }
```

Pulsar is a type of bike that extends the Bike class that shows that Pulsar is a Bike. Pulsar also uses an Engine's method, stop, using composition. So it shows that a Pulsar has an Engine.

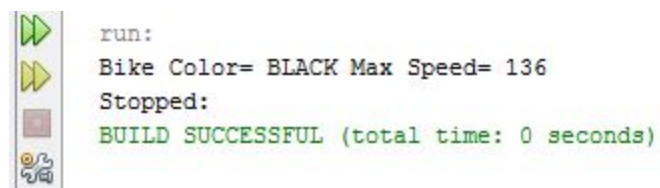
```
01. package relationsdemo;
02. public class Engine
03. {
04.     public void start()
05.     {
06.         System.out.println("Started:");
07.     }
08.     public void stop()
09.     {
10.         System.out.println("Stopped:");
11.     }
12. }
```

The Engine class has the two methods start() and stop() that are used by the Pulsar class.

```
01. package relationsdemo;
02. public class Demo
03. {
04.     public static void main(String[] args)
05.     {
06.         Pulsar myPulsar = new Pulsar();
07.         myPulsar.setColor("BLACK");
08.         myPulsar.setMaxSpeed(136);
09.         myPulsar.bikeInfo();
10.         myPulsar.PulsarStartDemo();
11.     }
12. }
```

In the code above we make an object of the Pulsar class and then initialize it. All the methods like setColor(), bikeInfo(), setMaxSpeed() are used here because of the Is-A relationship of the Pulsar class with the Bike class.

Output



```
run:
Bike Color= BLACK Max Speed= 136
Stopped:
BUILD SUCCESSFUL (total time: 0 seconds)
```