What is Is-A-Relationship in Java?

harshsethi2000

A relationship in Java means different relations between two or more classes. For example, if a class Bulb inherits another class Device, then we can say that Bulb is having is-a relationship with Device, which implies Bulb is a device.

In Java, we have two types of relationship:

1. **Is-A relationship:** Whenever one class inherits another class, it is called an IS-A relationship.
2. **Has-A relationship:** Whenever an instance of one class is used in another class, it is called HAS-A relationship.

Is-A relationship

IS-A Relationship is wholly related to Inheritance. For example – a kiwi is a fruit; a bulb is a device.

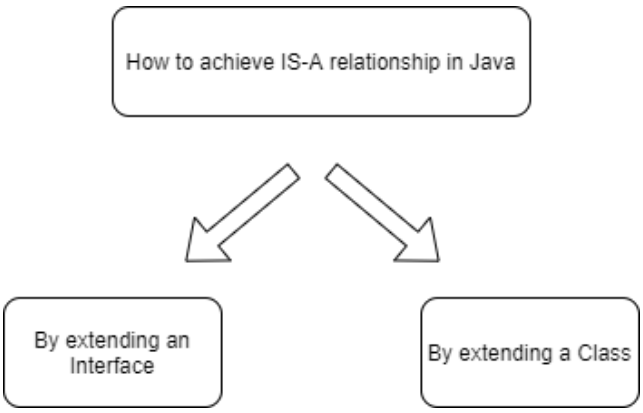
- IS-A relationship can simply be achieved by using extends Keyword.
- IS-A relationship is additionally used for code reusability in Java and to avoid code redundancy.
- IS-A relationship is unidirectional, which means we can say that a bulb is a device, but vice versa; a device is a bulb is not possible since all the devices are not bulbs.
- IS-A relationship is tightly coupled, which means changing one entity will affect another entity.

Advantage of IS-A relationship

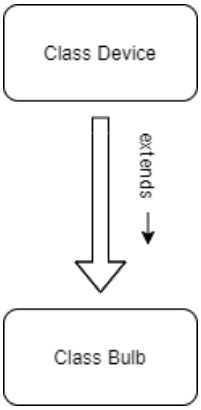
- Code Reusability.
- Reduce redundancy.

How to achieve IS-A relationship

IS-A relationship can simply be achieved by extending an interface or class by using extend keyword.



Let's understand the **IS-A relationship** with the help of a flowchart –



In the above flowchart, the class Bulb **extends** class Device, which implies that Device is the parent class of Bulb, and Class Bulb is said to have an Is-A relationship. Therefore we can say Bulb **Is-A** device.

Implementation of IS-A relationship

1. Class Device has a field named as deviceName.

2. Class Bulb extends Device that means Bulb is a type of Device.

3. Since Device is the parent class which implies that class Device can store the reference of an instance of class Bulb.

Example: Here is the implementation of the same, which is as follows:

Java

```
// Java program to demonstrate the
// working of the Is-A relationship

import java.io.*;

// parent class
class Device {

    private String deviceName;

    public void setDeviceName(String deviceName)
    {
        this.deviceName = deviceName;
    }

    public String getDeviceName()
    {
        return this.deviceName + " is a Device";
    }
}
```

```
// child class
class Bulb extends Device {
    public static void main(String gg[])
    {
        // parent class can store the reference
        // of instance of child classes
        Device device = new Bulb();

        // set the device name to bulb
        System.out.println("Device name is Bulb");
        device.setDeviceName("Bulb");

        // print the device name
        System.out.println(device.getDeviceName());
    }
}
```

Output

```
Device name is Bulb
Bulb is a Device
```

In the above java program Bulb class inherits the Device class. Therefore, we can say that Bulb is having an IS-A relationship with the class Device. Hence Bulb **is a** Device.

Last Updated : 01 Dec, 2021

Similar Reads

1. Happens-Before Relationship in Java
2. Difference Between java.sql.Time, java.sql.Timestamp and java.sql.Date in Java
3. How to Convert java.sql.Date to java.util.Date in Java?
4. Different Ways to Convert java.util.Date to java.time.LocalDate in Java
5. How to Convert java.util.Date to java.sql.Date in Java?
6. Java.util.BitSet class methods in Java with Examples | Set 2
7. Java.Lang.Float class in Java
8. Java.io.BufferedInputStream class in Java
9. Java.io.ObjectInputStream Class in Java | Set 1
10. Java.util.BitSet class in Java with Examples | Set 1

Related Tutorials

1. Spring MVC Tutorial
2. Spring Tutorial
3. Spring Boot Tutorial