STUDY ON INHERITANCE IN JAVA

Yamini Bhardwaj
Assistant professor, Department of Computer Science, Jayoti Vidyapeeth Women's University,
yaminibhardwaj23@gmail.com

Abstract:
Object-oriented programming languages are distinguished by inheritance, but its practical use is still not well understood. Programmers utilize inheritance for a variety of reasons, including sub typing, code reuse, enabling subclasses to modify the behavior of super classes, and simply classifying things.
The review article starts with an examination of JAVA OOP's ideas, such as inheritance, programming, and language technical notions using appropriate examples to give the reader a sense of the language. A general-purpose programming language called Java, which has a system programming influence, offers efficient low-level computations, data abstraction, object-oriented programming, and generic programming. Reusability of code is greatly influenced by inheritance.
Keywords: JAVA, OOP’s, Inheritance, reusability, Base class, Derived class, super class.

Introduction:

The concept of "inheritance" has been the subject of a lot of writing since the advent of the object-oriented paradigm. It is always preferable to reuse something that already exists as opposed to repeatedly making the same thing. This idea is supported by Java. Java classes can be utilized in a variety of ways. The simplest method for doing this is by reusing the properties of already existing classes while constructing new ones. Inheritance is the process through which a new class is derived from an existing one. The term "base class," "super class," or "parent class" is used to describe the old class. The term derived class, subclass, or child class is used to describe the new class. When an object inherits from a parent object, it takes on all of the parent object's properties and behaviors. The concept behind inheritance is that new classes can be built on top of older ones. You can reuse parent class fields and methods when you inherit from them, and you can also add new properties and methods. The parent-child relationship, also known as the IS-A relationship, is represented by inheritance.

Advantages to inheritance:

1. Reduce the amount of redundant code in a program.
2. If there is redundant code (variables and methods) in two related classes, the hierarchy can be refactored by shifting the redundant code up to the shared super class.
3. Better coding structure.
4. An improvement in code organization arises from moving common code to a super class.
5. Flexible changes to the code.

Syntax of java Inheritance:

```
Class  Baseclass_Name{
//declaring methods and fields
}
class Derivedclass_Name extends Baseclass_Name
 {
//declaring methods and fields
}
```

When you create a new class that derives from an existing class, you use the extends keyword.

Types of Inheritance in JAVA:

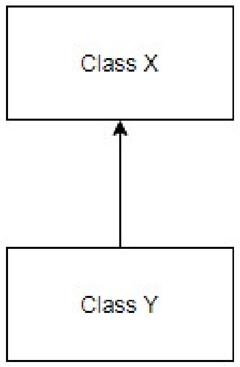
There are following types of Inheritance supported by JAVA:
1. Single Inheritance
2. Multilevel Inheritance
3. Multiple Inheritance
4. Hybrid Inheritance
5. Hierarchical Inheritance

Super Keyword:

- To refer to an immediate parent class instance variable, use the super keyword.
- You can call the immediate parent class method with the super keyword.
- Constructor of the immediate parent class may be called using super ().

1. **Single Inheritance:**

   It's incredibly simple to understand single inheritance. Single inheritance is the term used when a class only extends one other class. Class Y just extends class X, as can be seen in the flowchart below. In this instance, X is the parent class of Y, and Y would be X's child class.

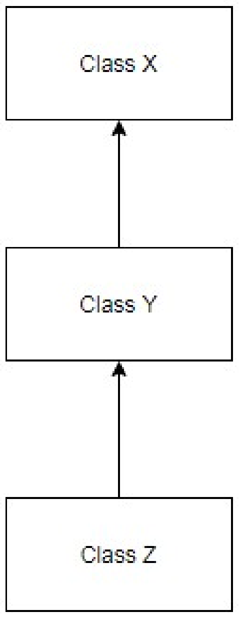


Structure of Single Inheritance

Syntax:
```
Class  X
{
//code
}
Class Y extends X
{
//code
}
```
Example 1:

2. **Multilevel Inheritance:**

   Multilevel inheritance is an OO technology feature that allows one to inherit from a derived class, making that derived class the base class for the new object. As you can see in below flow diagram Z is subclass or child class of Y and Y is a child class of X. For more details and example refer – Multilevel inheritance in Java.



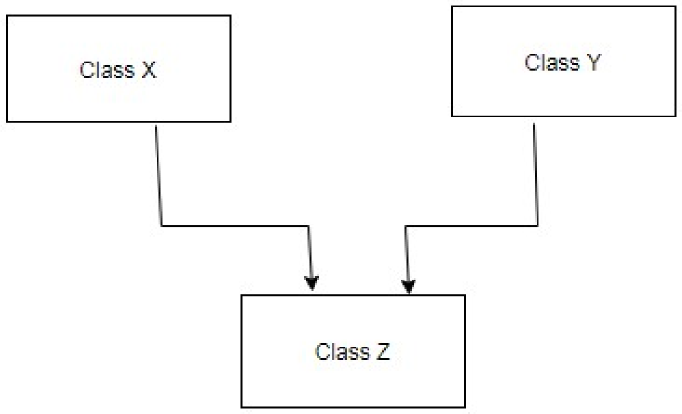
Structure of Multilevel Inheritance

Syntax:

```
class X
{
//code
 }
class Y extends X {
//code
}
class Z extends Y{
//code
}
```

3. **Multiple Inheritance:**

A class extending (or inheriting) from more than one base class is referred to as having "multiple inheritance". One base class or parent was a concept in the earlier lessons on inheritance. Multiple inheritance has the drawback of requiring the derived class to handle its dependencies on two base classes.

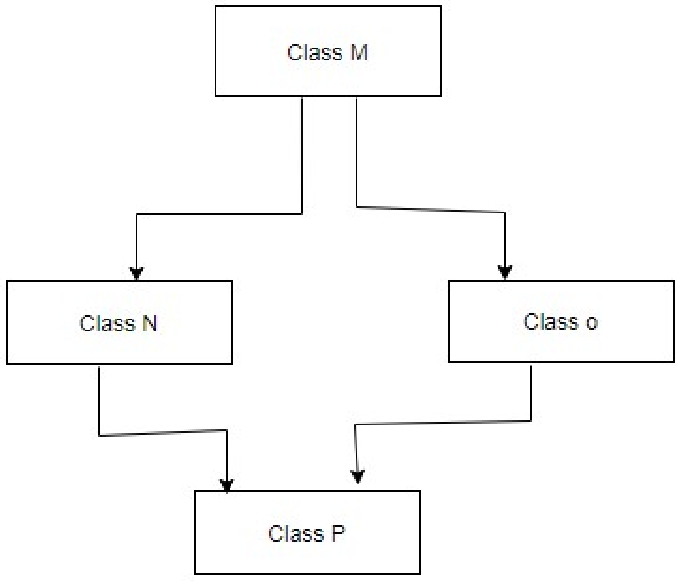


Structure of Multiple Inheritance

Note: 1. Software projects hardly ever use multiple inheritances. Multiple inheritance frequently causes issues in the hierarchy. When the class is extended further, this leads to unneeded complexity.
2. Multiple inheritance is not supported by the majority of the new OO languages, including Small Talk, Java, and C#. C++ supports multiple inheritances.

4. **Hybrid Inheritance:**

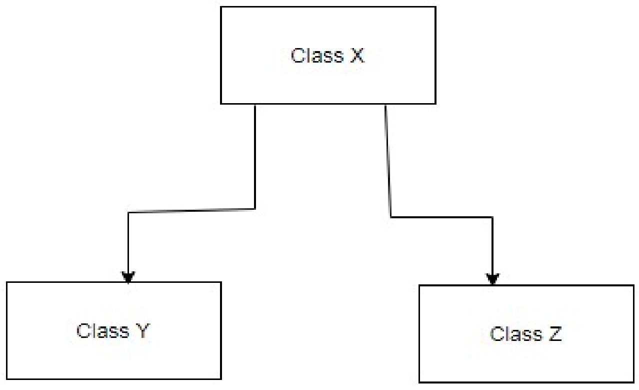
Hybrid inheritance can be defined as a blend of single inheritance and multiple inheritance in simple terms. The flowchart shown below is typical. The same manner that multiple inheritance can be performed in Java, so can a hybrid inheritance! employing interfaces Yes, you heard correctly. Java supports multiple and hybrid inheritance by using interfaces.



Structure of Hybrid Inheritance

5. Hierarchical Inheritance:

When multiple classes inherit from a single class, this is also known as hierarchical inheritance.



Structure of Hierarchical Inheritance

Syntax:

```
class A
{
 //code
}
class B extends A
{
//code
}
class c extends A{
//code
}
```

Implementation of multiple inheritance in Java:

The primary advantages of inheritance, parent class reuse, and protocol conformance, as well as how they may be accomplished in Java, have been discussed in the section above. Multiple parent class reuse and simultaneous adherence to all of their protocols are two of multiple inheritance's primary advantages. Because multiple inheritance is not supported in Java, these advantages cannot be directly obtained.

Limitations of the interface delegation method:

Both of the main advantages of multiple inheritance can be achieved with the interface-delegation technique. There are several elements that we have left out, and sometimes workarounds are required for the technique to be used in practice. A scaffolding class could be required because, for instance, the delegation object's protected fields and methods can only be accessed by extending classes. However, the interface-delegation method generally has practical advantages.

Conclusion:

In this paper, inheritance as it relates to software is extensively discussed, but there seems to be little on how it is actually employed. According to the numerous websites, blogs, and stories in the trade press, there appears to be a great deal of confusion about what it is and how to utilize it.

At the same time, criticisms of inheritance are made fairly openly in articles with provocative headings like "Why inheritance is wicked" or "Inheritance is evil, and must be eradicated."

Although Java does not support multiple inheritance, it does introduce an interface feature that does. This may be employed to accomplish the advantageous outcomes of multiple inheritance, according to certain theories. In this study, we investigated how this may be done. We started off by demonstrating the usage of delegation to reuse classes. Then, we identified the effects that could be accomplished solely through interfaces.

Reference:

1. https://beginnersbook.com
2. https://www.javatpoint.com
3. Arnold, K., Gosling, J., 1998. The Java Programming Language, The Java Series, second ed. Addison-Wesley, Reading,...