JAVA

What you learn ?

Java	
➢ What is inheritance?	➢ Types of inheritance
➢ Problem with Multiple inheritance	➢ Aggregation

Inheritance

Inheritance allows a class to inherit the properties and behavior of another class.

Inheritance represents the **IS-A relationship**.

- **Why use inheritance in java**
  - For Method Overriding (so runtime polymorphism can be achieved).
  - For Code Reusability.

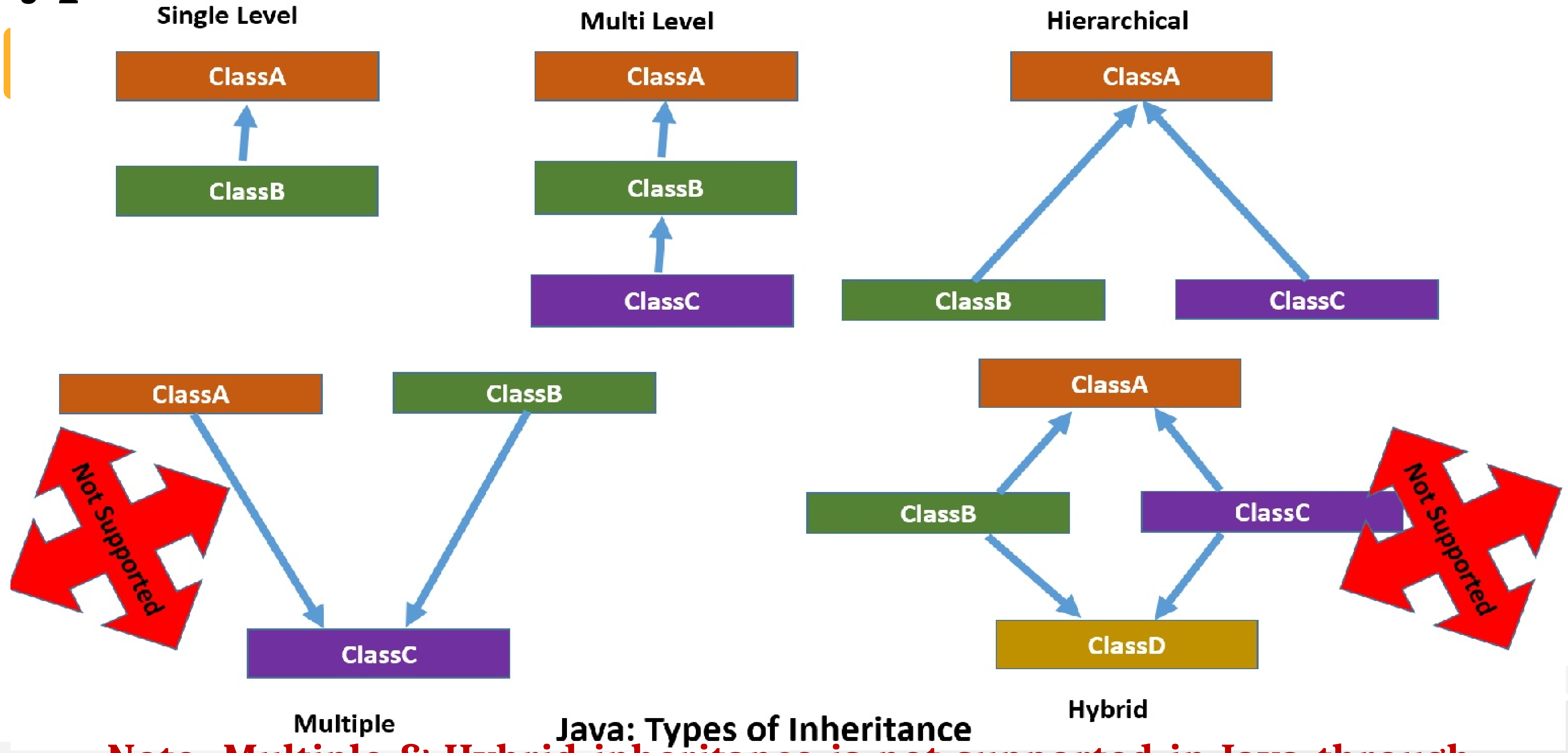
- **Super Class/Parent Class/Base Class:** Superclass is the class from where a subclass inherits the features.
- **Sub Class/Child Class/Derived Class:** Subclass is a class which inherits the other class
- **extends** keyword is used in the subclass declaration. The subclass can then access the public and protected members of the superclass, including fields and methods.

Super

In Java, the super keyword is used to refer to the parent class or superclass of a subclass. Here are some common uses of the super keyword:

- **To call a parent class constructor:** The super() constructor call is used to invoke a constructor of the parent class. This is used in a subclass constructor to initialize the inherited fields of the parent class before initializing the subclass fields.
- **To call a parent class method:** The super. prefix is used to call a method of the parent class from the subclass. This is used to access or override a method of the parent class in the subclass.
- **To access a parent class field:** The super. prefix is used to access a field of the parent class from the subclass. This is used to access or override a field of the parent class in the subclass.

Types of inheritance



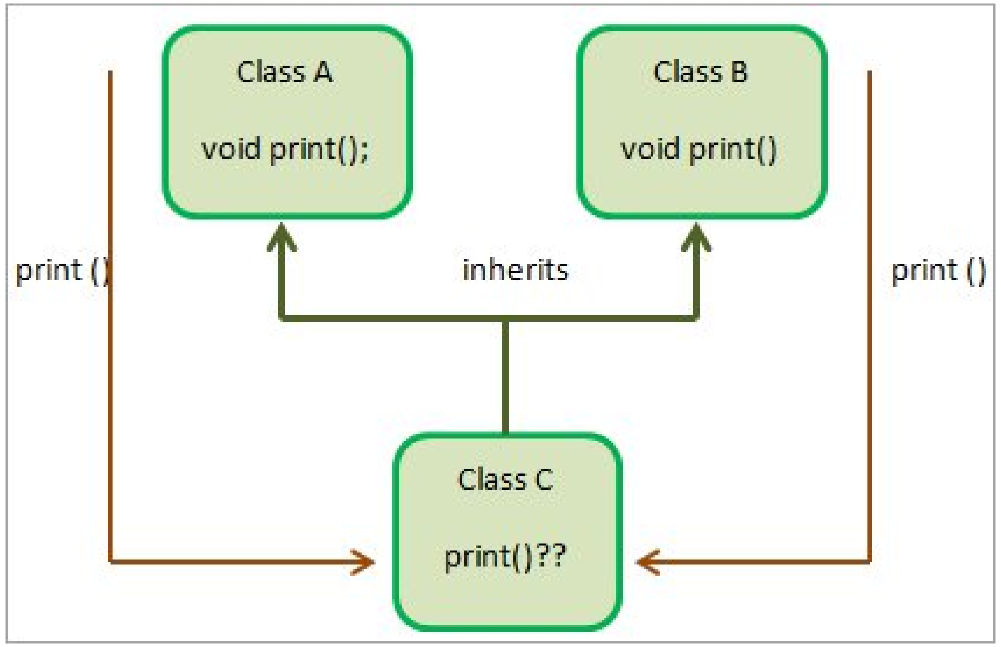
Note: Multiple & Hybrid inheritance is not supported in Java through class.

Problem with Multiple Inheritance

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.

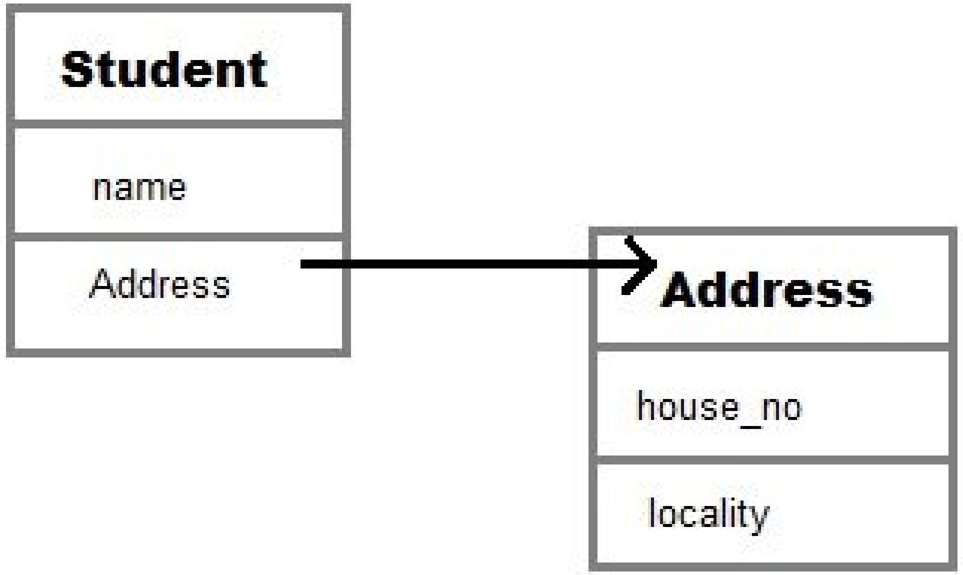


Aggregation / Has-A Relationship

In object-oriented programming, the "Has-A" relationship refers to a type of association between classes, where one class has a reference to another class as a member variable. This is also known as composition.

```
class Employee{
int id;
String name;
Address address;//Address is a class
...
}
```

In such case, Employee has an entity reference address, so relationship is Employee HAS-A address.



Method Overriding

- **Method overriding** is a feature in object-oriented programming that allows a subclass to provide its own implementation of a method that is already defined in its superclass.
- The method in the subclass must have the **same name, return type, and parameters as the method in the superclass,** and it must be marked with the **@Override** annotation.
- When a method is called on an object of the subclass, the JVM first checks if the subclass has overridden the method. If it has, the JVM calls the overridden method in the subclass. If it has not, the JVM calls the method in the superclass.
- Runtime polymorphism, also known as dynamic polymorphism, is a feature in object-oriented programming that allows objects of a subclass to be treated as objects of their superclass. This is achieved through method overriding, where a subclass provides its own implementation of a method that is already defined in its superclass.

Method Overriding

- **Dynamic method dispatch** is a mechanism in object-oriented programming that allows a method call on a superclass reference variable to be resolved at runtime based on the actual type of the object. This is also known as runtime polymorphism or dynamic polymorphism.
- In Java, dynamic method dispatch is achieved through method overriding. When a method is called on a superclass reference variable, the JVM looks for the method in the superclass. If the method is overridden in the subclass, the JVM calls the overridden method in the subclass instead.

Method Overloading

Method overloading is a feature in object-oriented programming that allows a class to have multiple methods with the same name but different parameters.

- The methods must differ in their parameter types, number, or order. When a method is called, the JVM determines which version of the method to call based on the arguments passed to it.
- Method overloading allows classes to provide multiple versions of a method with different parameter lists, which can make the code more readable and easier to use. It is often used in utility classes, where the same operation may need to be performed on different types of data.

final Keyword

The final keyword can be used in several contexts to create variables, methods, and classes that cannot be changed or overridden. .

- **Final variables:** A final variable is a constant value that cannot be modified after it has been assigned a value. To create a final variable, you use the final keyword.
- **Final methods:** A final method is a method that cannot be overridden by a subclass. This is useful when you want to ensure that a method's behavior cannot be changed. To create a final method, you use the final keyword.
- **Final classes:** A final class is a class that cannot be extended by another class. This is useful when you want to ensure that a class's behavior cannot be changed or overridden. To create a final class, you use the final keyword.

The final keyword can be used to create constants, prevent methods from being overridden, and prevent classes from being extended.

Thanks

Anirudha Gaikwad