Object Oriented Analysis and Design with Java

UE19CS353

Dr. L. Kamatchi Priya and Prof. Sindhu R Pai
Department of Computer Science and Engineering

PES UNIVERSITY ONLINE

UE19CS353: Object Oriented Analysis and Design with Java

Inheritance

Dr. L. Kamatchi Priya and Prof. Sindhu R Pai

Department of Computer Science and Engineering

Inheritance - Agenda

- Introduction
- Types
- Declaration of sub-class
- Member Access
- Coding examples – Demo
- Constructor calls
- Usage of super
- Overriding
- Points to think!
- References

Introduction

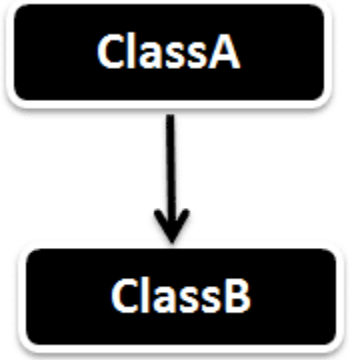
- Allows the creation of hierarchical classifications.
- We can create a general class that defines traits common to a set of related items.
- This class can then be inherited by other, more specific classes, each adding those things that are unique to it.
- Java terminologies:

  super class: The class that is inherited – Also known as parent class/base class

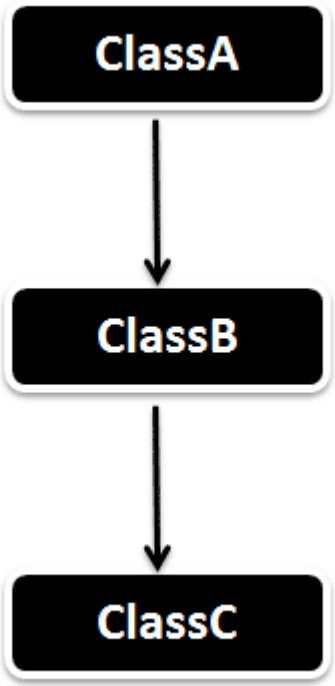
  sub class: The class that does the inheriting – Also known as derived class/ extended class/ child class

Note: A subclass is a specialized version of a super class. Inherits all of the members defined by the super class and adds its own, unique elements.

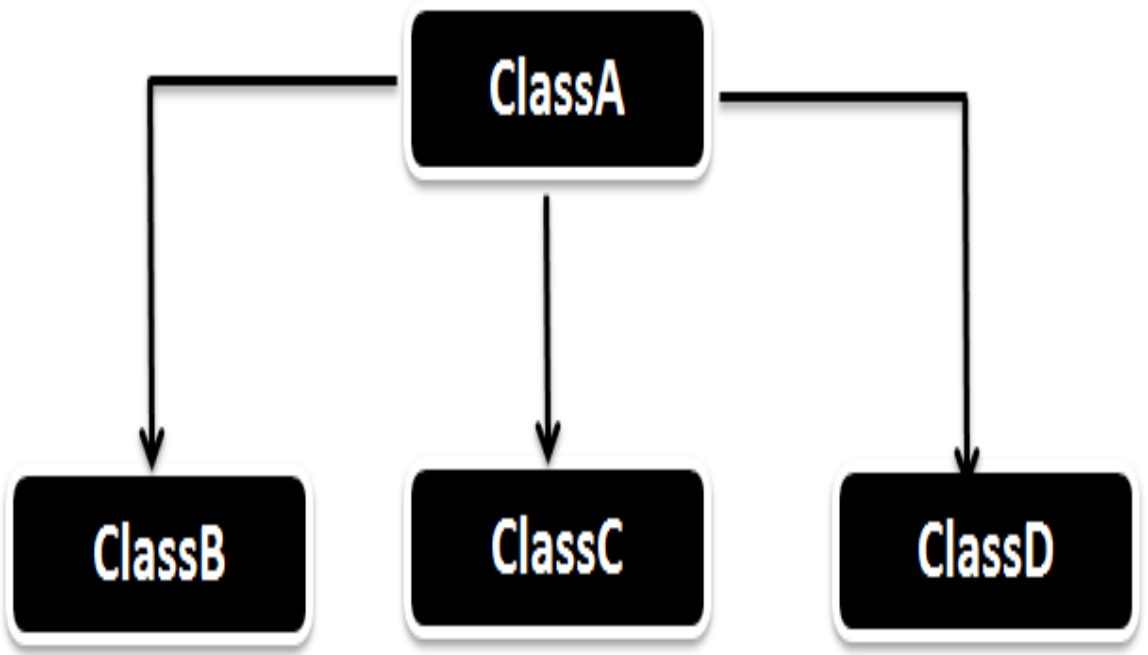
Types of Inheritance



Single Inheritance



Multi level Inheritance



Hierarchical Inheritance

Declaration of a sub class

- General form of **sub-class declaration: extends** keyword usage

```
class subclass-name extends superclass-name {
  // body of class
}
```

- Java does not support the inheritance of multiple super classes into a single subclass

- subclass becomes a superclass of another subclass

- No class can be a superclass of itself

Member Access

- **private access:** Member that has been declared as private will remain private to its class. It is not accessible by any code outside its class, including subclasses

- **Default access:** When a member does not have an explicit access modifier, it is visible to subclasses as well as to other classes in the same package.

- **Protected access:** When an element to be seen outside your current package, but only to classes that subclass your class directly

Member Access in detail

	Private	No Modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Constructor calls in detail

- If a class contains no constructor declarations, then a default constructor is implicitly declared.

- Each **constructor calls its direct superclass constructor**, which calls its direct superclass constructor, until **Object's constructor is called**.

- Coding examples

Usage of super keyword

- **super : A reference variable which is used to refer immediate parent class object**.
- On creation of instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.
- Used in three ways:

  To refer immediate parent class instance variable.

  To invoke immediate parent class method.

  super() can be used to invoke immediate parent class constructor
- Coding examples

Overriding

- Provides runtime polymorphic behavior and known as **dynamic method dispatch/ Late binding**
- A language feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super classes
- The **implementation in the subclass overrides the implementation in the superclass** by providing a method that **has same name, same parameters or signature, and same return type as the method in the parent class**.
- The **version of a method that is executed** will be determined by **the object that is used to invoke it.**

Overriding example code

```
class Animal
{
    void eats(){System.out.println("it is eating!!");}
    void sleeps(){System.out.println("it is sleeping!");}
}
    class Dog extends Animal{
        @Override
        void eats()
        {
            System.out.println("Dog is eating");
        }
    }
```

@override: Used when the subclass method overrides its superclass method.

Points to think!!

- What happens if we call a method in the ctor?
- What happens if we call a method in a static method of the class?
- What happens if we call a private method in the ctor?
- Can we create an instance of a super class and can we refer it by a sub class reference?
- Can we perform typecasting(upcasting/downcasting) on the base class object referred by sub class reference?
- If we do not add the @override annotation, will the subclass not override the method in the super class?

THANK YOU

Dr. L. Kamatchi Priya and Prof. Sindhu R Pai

Department of Computer Science and Engineering

priyal@pes.edu

sindhurpai@pes.edu

+91 8277606459