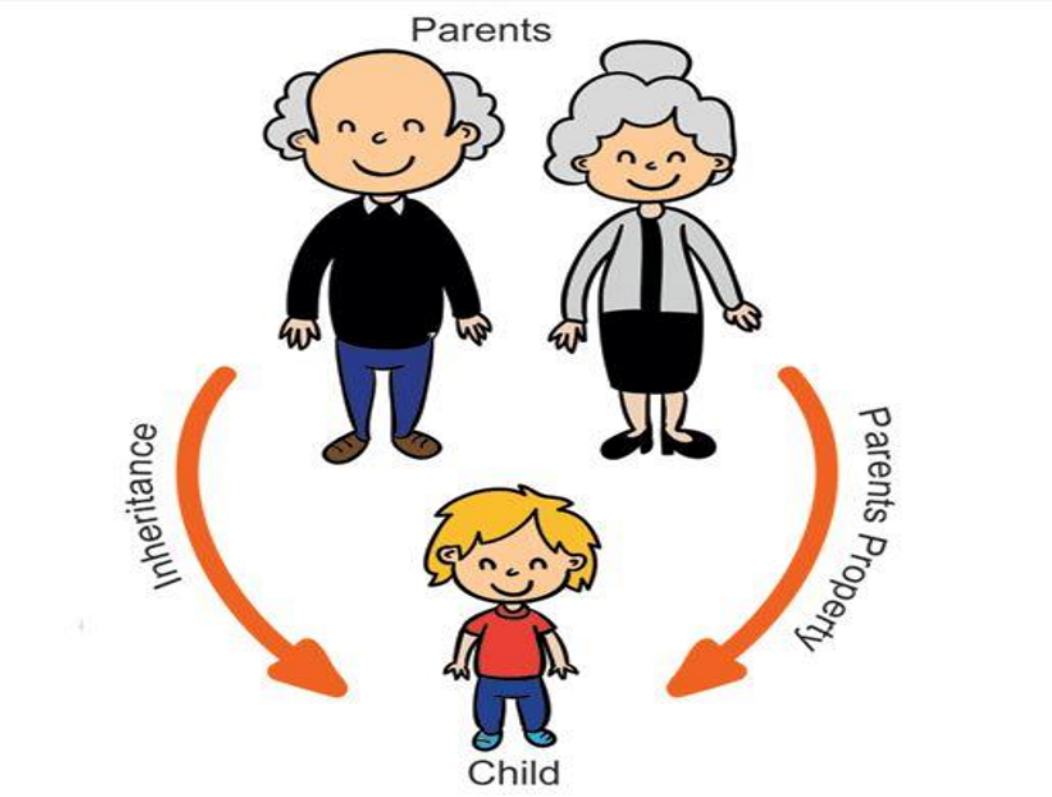
Inheritance

W.A E.M Weerasinghe

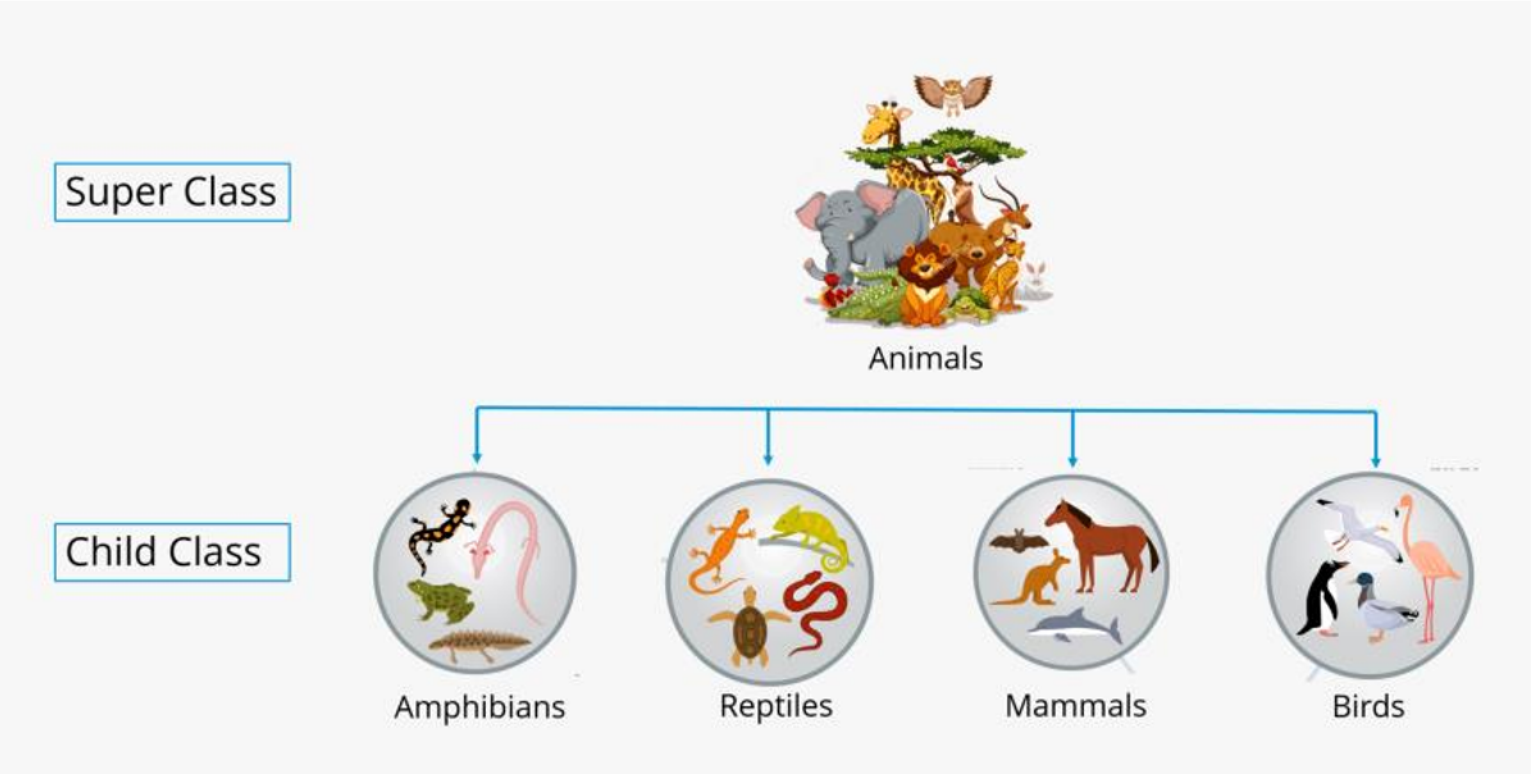
Inheritance

- **Inheritance in Java** is a mechanism in which one object obtains all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).
- The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.
- Inheritance represents the **IS-A relationship** which is also known as a *parent-child* relationship.

Inheritance



Inheritance



Why use Inheritance in JAVA

- For Code Reusability
- You can minimize the length of duplicate code
- Due to reducing the length of code, redundancy of the application is also reduced
- Make application code more flexible

Terms used in Inheritance

- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a *derived class, extended class, or child class*.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a *base class or a parent class.*
- **Reusability:** As the name specifies, **reusability is a mechanism** which facilitates you to reuse the fields and methods of the existing class when you create a new class. **You can use the same fields and methods already defined in the previous class.**

The Syntax of JAVA Inheritance

```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

Single Inheritance

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

- When a class inherits another class, it is known as a *single inheritance*.
- In the example given , Dog class inherits the Animal class, so there is the single inheritance.

Multilevel Inheritance

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

- When there is a **chain of inheritance**, it is known as *multilevel inheritance*.
- As you can see in the example given, BabyDog class inherits the Dog class which again inherits the Animal class, so there is a multilevel inheritance.

Hierarchical Inheritance

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

- When two or more classes inherits a single class, it is known as *hierarchical inheritance*.
- In the example, Dog and Cat classes inherits the Animal class, so there is hierarchical inheritance.

Types Of Inheritance

Single Inheritance
Class A
↑
Class B
	public class A {
	.....
	}
	public class B **extends** A {
	.....
	}

Multi Level Inheritance Class A ↑ Class B ↑ Class C	public class A { .....} public class B **extends** A {.....} public class C **extends** B {..... }
Hierarchical Inheritance Class A ↑ ↑ Class B Class C	public class A { .....} public class B **extends** A {.....} public class C **extends** A {..... }

Multiple Inheritance
Class A Class B
↖ ↗
Class C
	public class A { .....}
	public class B {.....}
	public class C **extends** A,B {
	.....
	} // Java does not support mutiple Inheritance

Multiple Inheritance

- Java supports *single inheritance*, meaning that a derived class can have only one parent class
- ***Multiple inheritance* allows a class to be derived from two or more classes, inheriting the members of all parents**
- Java does not support multiple inheritance
- In most cases, the use of interfaces gives us aspects of multiple inheritance without the overhead