Inheritance:

* When a child class inherit the property of it parent class known as Inheritance.

Representation:

class childclassname extends parentclassname{

}

* Inheritance is achieved by using super() statement.
* Used to call parent/super class constructor based on parameter.
* Super () should be the first statement inside the constructor body.

Note:

* In inheritance only non-static members are involves .
* super() and this() can't be written together.

**Important Points**

1. Code reuse is the most important benefit of inheritance because subclasses inherits the variables and methods of superclass.
2. [Private](https://www.digitalocean.com/community/tutorials/java-access-modifiers) members of superclass are not directly accessible to subclass. As in this example, Animal variable noOfLegs is not accessible to Cat class but it can be indirectly accessible via getter and setter methods.
3. Superclass members with default access is accessible to subclass ONLY if they are in same package.
4. Superclass [constructors](https://www.digitalocean.com/community/tutorials/constructor-in-java) are not inherited by subclass.
5. If superclass doesn’t have default constructor, then subclass also needs to have an explicit constructor defined. Else it will throw compile time exception. In the subclass constructor, call to superclass constructor is mandatory in this case and it should be the first statement in the subclass constructor.
6. [**Java doesn’t support multiple inheritance**](https://www.digitalocean.com/community/tutorials/multiple-inheritance-in-java), a subclass can extends only one class. Animal class is implicitly extending Object class and Cat is extending Animal class but due to java inheritance transitive nature, Cat class also extends Object class.
7. We can create an instance of subclass and then assign it to superclass variable, this is called **upcasting**. Below is a simple example of upcasting:
8. Cat c = new Cat(); //subclass instance
9. Animal a = c; //upcasting, it's fine since Cat is also an Animal
10. When an instance of Superclass is assigned to a Subclass variable, then it’s called **downcasting**. We need to explicitly cast this to Subclass. For example;
11. Cat c = new Cat();
12. Animal a = c;
13. Cat c1 = (Cat) a; //explicit casting, works fine because "c" is actually of type Cat

Note that Compiler won’t complain even if we are doing it wrong, because of explicit casting. Below are some of the cases where it will throw ClassCastException at runtime.

Dog d = new Dog();

Animal a = d;

Cat c1 = (Cat) a; //ClassCastException at runtime

Animal a1 = new Animal();

Cat c2 = (Cat) a1; //ClassCastException because a1 is actually of type Animal at runtime

1. We can override the method of Superclass in the Subclass. However we should always annotate overridden method with [@Override annotation](https://www.digitalocean.com/community/tutorials/java-override-method-overriding). The compiler will know that we are overriding a method and if something changes in the superclass method, we will get a compile-time error rather than getting unwanted results at the runtime.
2. We can call the superclass methods and access superclass variables using **super** keyword. It comes handy when we have the same name variable/method in the subclass but we want to access the superclass variable/method. This is also used when Constructors are defined in the superclass and subclass and we have to explicitly call the superclass constructor.
3. We can use instanceof instruction to check the inheritance between objects, let’s see this with below example.

```

Cat c = new Cat();

Dog d = new Dog();

Animal an = c;

boolean flag = c instanceof Cat; //normal case, returns true

flag = c instanceof Animal; // returns true since c is-an Animal too

flag = an instanceof Cat; //returns true because a is of type Cat at runtime

flag = an instanceof Dog; //returns false for obvious reasons.

```

1. We can’t extend Final classes in java.
2. If you are not going to use Superclass in the code i.e your Superclass is just a base to keep reusable code then you can keep them as [Abstract class](https://www.digitalocean.com/community/tutorials/abstract-class-in-java) to avoid unnecessary instantiation by client classes. It will also restrict the instance creation of base class.