



INHERITANCE

JAVA (oop concept)



MAY 3, 2018

V.DILUXSHAN

Sabaragamuwa University of SriLanka

The process by which one class acquires the properties(data members) and functionalities(methods) of another class is called **inheritance**. The aim of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from the another class.

Child Class:

The class that extends the features of another class is known as child class, sub class or derived class.

Parent Class:

The class whose properties and functionalities are used(inherited) by another class is known as parent class, super class or Base class.

Syntax: Inheritance in Java

To inherit a class we use extends keyword. Here class XYZ is child class and class ABC is parent class. The class XYZ is inheriting the properties and methods of ABC class.

```
class XYZ extends ABC
{
}
```

Inheritance Example

In this example, we have a base class `Teacher` and a sub class `PhysicsTeacher`. Since class `PhysicsTeacher` extends the designation and college properties and `work()` method from base class, we need not to declare these properties and method in sub class.

Here we have `collegeName`, `designation` and `work()` method which are common to all the teachers so we have declared them in the base class, this way the child classes like `MathTeacher`, `MusicTeacher` and `PhysicsTeacher` do not need to write this code and can be used directly from base class.

```
class Teacher {
    String designation = "Teacher";
    String collegeName = "Beginnersbook";
    void does(){
        System.out.println("Teaching");
    }
}

public class PhysicsTeacher extends Teacher{
    String mainSubject = "Physics";
    public static void main(String args[]){
        PhysicsTeacher obj = new PhysicsTeacher();
        System.out.println(obj.collegeName);
        System.out.println(obj.designation);
        System.out.println(obj.mainSubject);
    }
}
```

```
        obj.does();
    }
}
```

Output:

```
Beginnersbook
Teacher
Physics
Teaching
```

Based on the above example we can say that `PhysicsTeacher` **IS-A** `Teacher`. This means that a child class has IS-A relationship with the parent class. This inheritance is known as **IS-A relationship** between child and parent class

Note:

The derived class inherits all the members and methods that are declared as public or protected. If the members or methods of super class are declared as private then the derived class cannot use them directly. The private members can be accessed only in its own class. Such private members can only be accessed using public or protected getter and setter methods of super class as shown in the example below.

```
class Teacher {
    private String designation = "Teacher";
    private String collegeName = "Beginnersbook";
    public String getDesignation() {
        return designation;
    }
    protected void setDesignation(String designation) {
        this.designation = designation;
    }
    protected String getCollegeName() {
        return collegeName;
    }
    protected void setCollegeName(String collegeName) {
        this.collegeName = collegeName;
    }
    void does(){
        System.out.println("Teaching");
    }
}

public class JavaExample extends Teacher{
    String mainSubject = "Physics";
    public static void main(String args[]){
        JavaExample obj = new JavaExample();
        /* Note: we are not accessing the data members
        * directly we are using public getter method
        * to access the private members of parent class
        */
        System.out.println(obj.getCollegeName());
        System.out.println(obj.getDesignation());
        System.out.println(obj.mainSubject);
        obj.does();
    }
}
```

```
}
```

The output is:

```
Beginnersbook
```

```
Teacher
```

```
Physics
```

```
Teaching
```

The important point to note in the above example is that the child class is able to access the private members of parent class through **protected methods** of parent class. When we make a instance variable(data member) or method **protected**, this means that they are accessible only in the class itself and in child class. These public, protected, private etc. are all access specifiers and we will discuss them in the coming tutorials.