# Lesson 5

# *Inheritance*

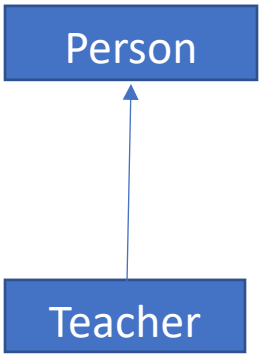
By the end of this lesson you will learn:

- Inheritance definition and example
- Different types of Inheritance
- Constructor chaining
- this and super keyword
- Method Overriding in inheritance

---

## Inheritance

Inheritance is the process where one class possess the properties of another class. Inheritance is used for code reusability. To implement parent-child relationship.




Here Teacher class inherits Person class. Person class is called base class, super class or parent class and Teacher class is called sub class, derived class or child class.

- The subclass inherits all data attributes of its superclass
- The subclass inherits all methods of its superclass
- The subclass inherits all associations of its superclass

The subclass can:

- Add new functionality
- Use inherited functionality
- Override inherited functionality

Inheritance is declared using the "extends" keyword.

***Syntax for inheritance:***

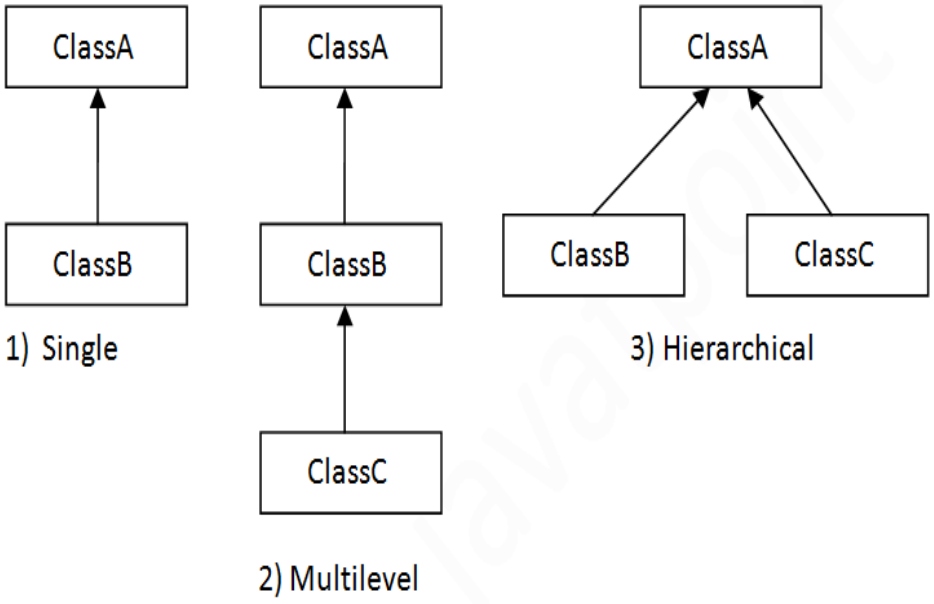
To inherit the person class from teacher class you need to write the following syntax

```
public class Person
{
        private String name;
        private Date dob;
        [...]
    }


public class Employee extends Person
{
        private int employeID;
        private int salary;
        private Date startDate;
        [...]
}


Employee anEmployee = new Employee();
```

- ✓ Here in the first line of the employee class we have used extends keyword to inherit person class.
- ✓ Then we have created object of employee class.
- ✓ From theory we know that this object can access methods of both classes.

There are different types of inheritance in java.

- Single inheritance
- Multilevel inheritance
- Hierarchical inheritance




**Single level inheritance:**

In this type of inheritance there is a single parent and single child class.

**Multilevel Inheritance:** In this inheritance the parent of one class is child of another class.

**Hierarchical inheritance:** In this inheritance there are multiple child class that inherits one parent class.

There is no limit to the number of subclasses a class can have. There is no limit to the depth of the class tree. Each Java class has one (and only one) superclass. C++ allows for multiple inheritance.

**Constructor to Initialize:** Classes use constructors to initialize instance variables. When a subclass object is created, its constructor is called. It is the responsibility of the subclass constructor to invoke the appropriate superclass constructors so that the instance variables defined in the superclass are properly initialized.

Lets see one example for the explanation:

```
class Person
{
        private String name;
```

```
	private int age;
	private String gender;
	Person()
	{
	System.out.println("Inside parent class empty cons");
	}
	Person(String name,int age,String gender)
	{
		System.out.println("Inside parent class param
cons");
		this.name=name;
		this.age=age;
		this.gender=gender;

	}
	public void setName(String name)
	{
		this.name=name;
	}
	public void setAge(int age)
	{
		this.age=age;
	}
	public void setGender(String gender)
	{
		this.gender=gender;
	}
	public String getName()
	{
```

```
		return name;
	}
	public int getAge()
	{
		return age;
	}
	public String getGender()
	{
		return gender;
	}

}
class Teacher extends Person
{
	private String id;
	private double salary;
	Teacher()
	{
	System.out.println("Inside child class empty cons");
	}
	Teacher(String id,double salary)
	{
		System.out.println("Inside child class param cons");
		this.id=id;
		this.salary=salary;

	}
	public void setId(String id)
```

```
	{
		this.id=id;
	}
	public void setSalary(double salary)
	{
		this.salary=salary;
	}
	public String getId()
	{
		return id;
	}
	public double getSalary()
	{
		return salary;
	}
	public void showInfo()
	{
		System.out.println("Name is : "+getName());
		System.out.println("Age is : "+getAge());
		System.out.println("gender is : "+getGender());
		System.out.println("Id is : "+id);
		System.out.println("Salary is : "+salary);

	}

}
class Start
{
```

```
    public static void main(String args[])
    {
        Teacher t1=new Teacher("11-1",1000);

    }
}
```

Here output will be:

*Inside parent class empty cons*

*Inside child class param cons*

So the object of teacher class is invoking the empty constructor of Person class.

**this keyword in constructor:** this can be used to call the constructor of its own class. Lets see the above example using this keyword.

```
Person()
{
System.out.println("Inside parent class");
}
Person(String name,int age,String gender)
{
    this(name,age);
    this.gender=gender;
}
Person(String name,int age)
{
    this.name=name;
    this.age=age;
}
```

- ✓ Here we have written three constructor of person class.
- ✓ One is empty and another two is parameterized.
- ✓ This constructor “Person(String name,int age,String gender)” is passing its first two parameter to its another constructor.
- ✓ Here first “Person(String name,int age)” constructor will be executed and then “Person(String name,int age,String gender)” will be executed if we write the following inside main:

```
class Start
{
    public static void main(String args[])
    {
        Person p1=new Person("Shakib",28,"Male");

    }
}
```

**Super keyword in constructor:**

Superclass constructors can be called using the "super" keyword in a manner similar to "this". It must be the first line of code in the constructor. If a call to super is not made, the system will automatically attempt to invoke the no-argument constructor of the superclass.

Lets see the above person teacher class to understand super keyword:

```
class Person
{
    private String name;
    private int age;
    private String gender;
    Person()
    {
    System.out.println("Inside parent class");
```

```
    }
    Person(String name,int age,String gender)
    {
        this(name,age);
        this.gender=gender;
    }
    Person(String name,int age)
    {
        this.name=name;
        this.age=age;
    }
    [……]
    }

class Teacher extends Person
{
    private String id;
    private double salary;
    Teacher()
    {
    System.out.println("Inside child class");
    }
    Teacher(String name,int age,String gender,String id,double salary)
    {
        super(name,age,gender);
        this.id=id;
        this.salary=salary;
    }
```

```
        [……..]
}

class Start
{
        public static void main(String args[])
        {
        Teacher t1=new Teacher("Shakib",28,"Male","11-1",1000);
        }
}
```

Here the parameterized constructor of parent class will be called as we used super keyword in the child class constructor.

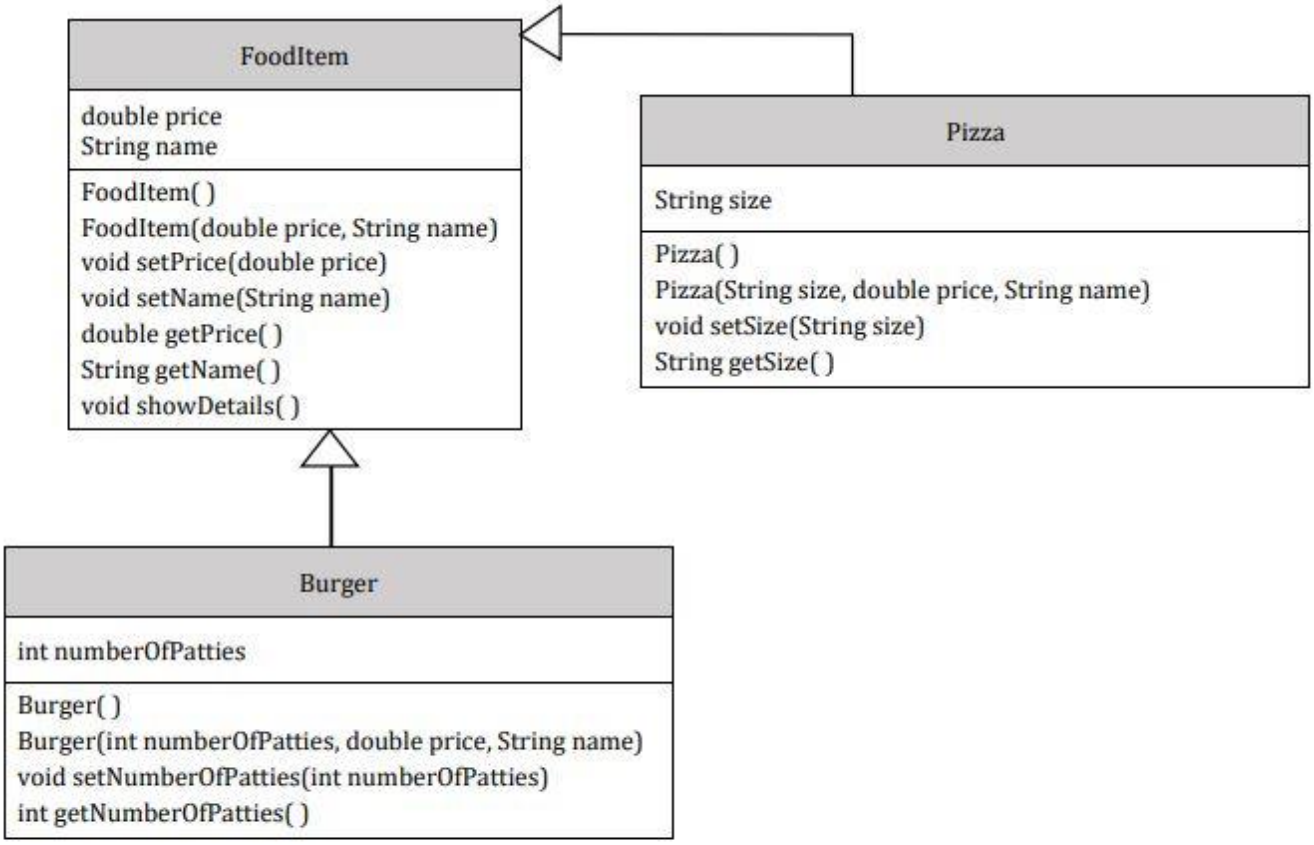
**Method Overriding in child class:** If you write the same method in both parent and child class then this is called method overriding.

```
public class BankAccount
{
        private String ownersName;
        private int accountNumber;
        protected float balance;

        public void deposit(float anAmount)
        {
                if (anAmount>0.0)
                        balance = balance + anAmount;
        }
        public void withdraw(float anAmount)
        {
```

```
            if ((anAmount>0.0) && (balance>anAmount))
                balance = balance - anAmount;
        }

        public float getBalance()
        {
            return balance;
        }
}

public class OverdraftAccount extends BankAccount
{
        private float limit;

        public void withdraw(float anAmount) // Overriding method
        {
            if ((anAmount>0.0) && (getBalance()+limit>anAmount))
                balance = balance - anAmount;
        }

}
```

- ✓ Here withdraw() method is written in both parent and child class.
- ✓ Object of child class will invoke its own withdraw method and object of parent class will invoke its own withdraw method.

Exercise:




Write a class **Start** that contains the main( ) method. Inside the main( ) create two objects of **Burger** and two objects of **Pizza.** Illustrate the purpose of all the methods and constructors.