Inheritance

Inheritance

- An important pillar/main feature of OOP. It is the mechanism in java by which **one class is allow to inherit the features**(fields and methods) **of another class**.
- Also known as “is-a” relationship

Inheritance-Important terminology

- **Super Class:** also known as base class or a parent class. It has its own attributes and behaviors
- **Sub Class:** The class that inherits the other class features. also known as derived class, extended class, or child class. The subclass can add its own fields and methods in addition to the superclass fields and methods.
- **Reusability:** Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

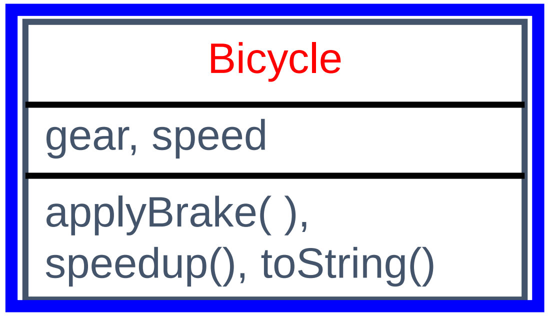
Keyword: extends

Syntax :

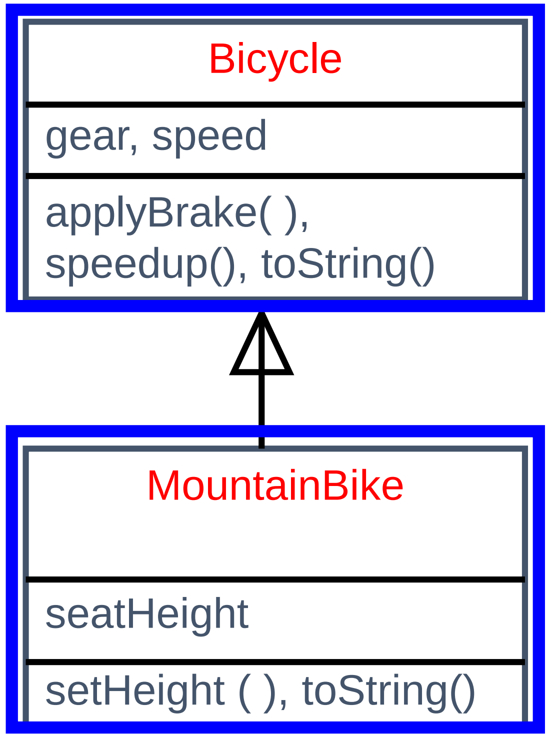
```
class derived-class extends base-class
{
   //methods and fields
}
```

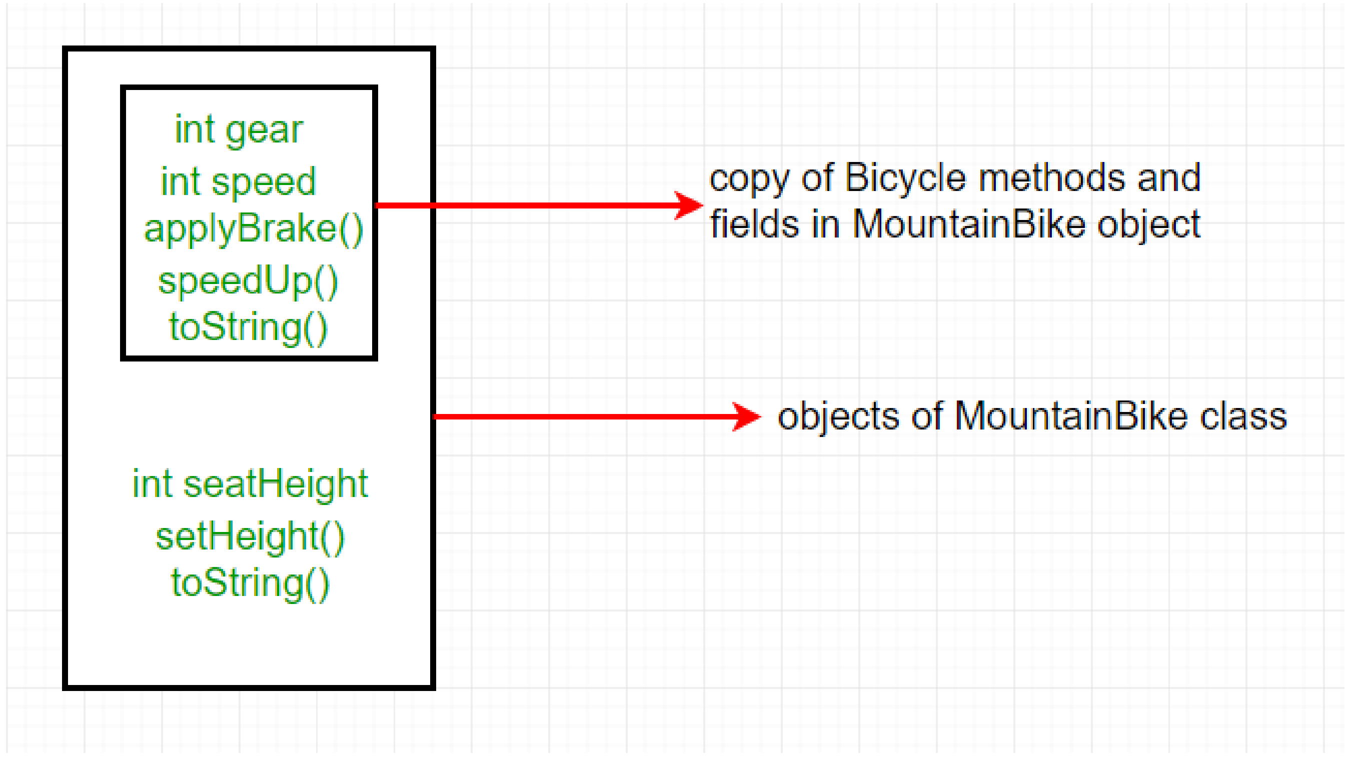
Case: Bicycle

```
class Bicycle // base class
{
         public int gear;
         public int speed;

         public Bicycle(int gear, int speed) //constructor
                  this.gear = gear;
                  this.speed = speed;
         }

         public void applyBrake(int decrement){
                  speed -= decrement;
         }

         public void speedUp(int increment)
         {
                  speed += increment;
         }

         public String toString()  {
                  return("No of gears are "+gear
                        +"\n“ + "speed of bicycle is "+speed);
         }
}
```



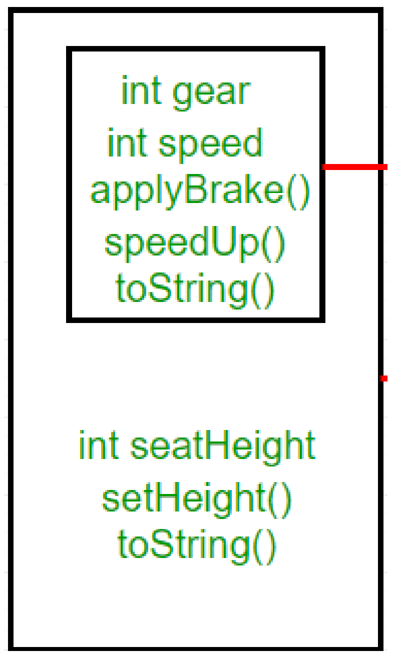
Case: Bicycle




int gear
int speed
applyBrake()
speedUp()
toString()
int seatHeight
setHeight()
toString()
copy of Bicycle methods and fields in MountainBike object
objects of MountainBike class

```
class MountainBike extends Bicycle // derived class
{
          public int seatHeight;

          public MountainBike(int gear,int speed, int startHeight)
          {
                    // invoking base-class(Bicycle) constructor
                    super(gear, speed);
                    seatHeight = startHeight;
          }

          public void setHeight(int newValue) // the MountainBike subclass adds one more method
          {
                    seatHeight = newValue;
          }

          // overriding toString() method of Bicycle to print more info
          public String toString()
          {
                    return (super.toString()+
                                        "\nseat height is "+seatHeight);
          }

}
```

MountainBike



Draw the relationship of the two class below:

```
import java.util.*;
import java.lang.*;
import java.io.*;

class one {
	public void print_geek() {
		System.out.println("Geeks");
	}
}

class two extends one {
	public void print_for() {
		System.out.println("for");
	}
}

public class Main {
	public static void main(String[] args) {
		two g = new two();
		g.print_geek();
		g.print_for();
		g.print_geek();
	}
}
```

Inheritance and constructors

constructor of base class with no argument gets automatically called in derived class constructor.

```
class Base {
         Base() {
                  System.out.println("Base Class Constructor Called ");
         }
}

class Derived extends Base {
         Derived() {
                  System.out.println("Derived Class Constructor Called ");
         }
}

public class Main {
         public static void main(String[] args) {
                  Derived d = new Derived();
         }
}
```

```
class Base {
		int x;
		Base(int _x) {
			x = _x;
		}
}

class Derived extends Base {
		int y;
		Derived(int _x, int _y) {
			super(_x);
			y = _y;
		}

		void Display() {
			System.out.println("x = "+x+", y = "+y);
		}
}

public class Main {
		public static void main(String[] args) {
			Derived d = new Derived(10, 20);
			d.Display();
		}
}
```

To call parameterized contructor of base class using super().

The point to note is **base class constructor call must be the first line in derived class constructor**.