Use of **this**  keyword in constructor:

* **this** keyword is used within a constructor or within a non-static method.
* It should not be used with static method because static is not associated with any objects.
* **this** keyword represents current instance.
* **this** keyword is used to access non-static member of the current instance.

Syntax: this.non-staticMemberName.

**Constructor Overloading:**

Creating multiple constructors with unique arguments list is called as constructor overloading. When constructor is overloaded, a specific constructor will be executed based on inputs given.

**Use of “call to this” statement in a constructor---this() :**

* this() is used to call current class constructor within a constructor.
* To call current class constructor within a constructor, insert a call to this( this())
* Constructor call must be the 1st statement inside a constructor
* Call to this statement calls a specific constructor based on the arguments.

**When we have to call a constructor within a constructor?**

When the code is repeating in the constructor then call a constructor using this() statement

**INHERITANCE:**

Acquiring members from one class to another class is called as inheritance. Its used for re-usability

* **Super class/Base class:** The class from which members are acquiring is called as super class
* **Sub class/derived class:** The class to which members are acquired is called as sub class
* To Inherit the members use “extends” keyword
* In java sub class extends super class
* Private and static members cannot be inherited
* Constructor can’t be inherited

**Can a class extend more than one class?**

Ans: No. Its not supported in java

A class extends multiple classes (class C extends A, B) is called as multiple inheritance. Multiple inheritance is not supported using classes in java.

**this** and **this()**

**super** and **super()**

**Use of “call to super” statement--super() :**

* Call to super is used to call super class constructor from sub class constructor.
* Since constructor call must be the 1st statement within a constructor, call to super should be the 1st statement.
* When there is inheritance between classes, java implicitly calls default super class constructor if sub-class constructor does not have explicit call.
* Java can add implicit call only for default constructor.
* If there is no default constructor in super class then programmer has to explicitly call parameterized super class constructor using super keyword otherwise constructor chain breaks.