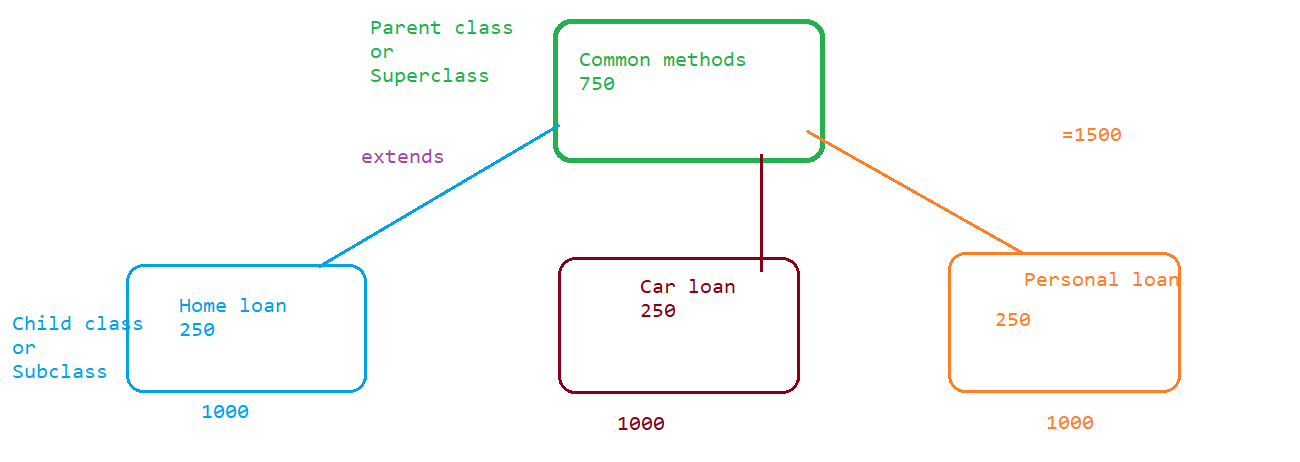
Inheritance:



1. It is an oops concept which allow the user to achieve reusability by having a parent and child relationship using extends keyword.

2. Through inheritance child class get the access of all the properties (methods and variables) of parent class.

3. Type of method (static or non-static) doesn’t matter to follow the rules of inheritance.

Example:

**public** **class** Parent {

**public** **void** home()

{

System.***out***.println("Home method from parent class");

}

**public** **void** car()

{

System.***out***.println("car method from parent class");

}

**public** **void** furniture()

{

System.***out***.println("furniture method from parent class");

}

**public** **static** **void** property()

{

System.***out***.println("Property method from parent class");

}

**public** **class** Child **extends** Parent{

**public** **void** bike()

{

home();

car();

System.***out***.println("Bike method from child class");

}

**public** **static** **void** main(String[] args) {

Child c = **new** Child();

c.home();

c.bike();

c.car();

*property*();

Parent p = **new** Parent();

p.home();

}

Output after executing the child class:

Home method from parent class

Home method from parent class

car method from parent class

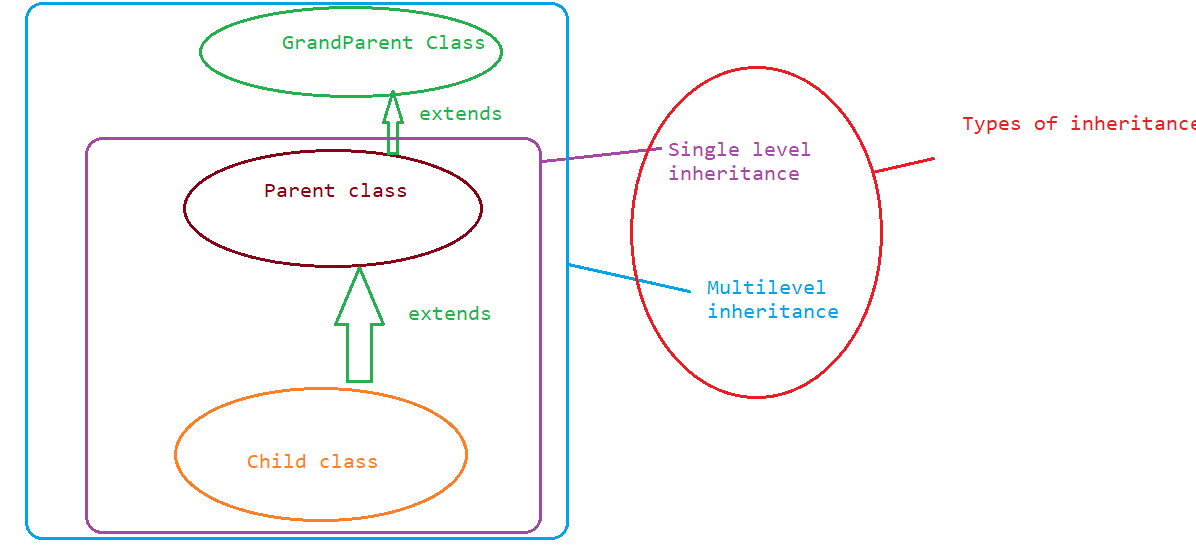
Bike method from child class

car method from parent class

Property method from parent class

Home method from parent class

4. Multilevel inheritance is possible in which we can have a child having parent and that parent can have parent this represents multilevel inheritance.



Example of multilevel inheritance:

**public** **class** GrandParent {

**public** **void** farm()

{

System.***out***.println("Farm method from grand parent class");

}

}

**public** **class** Parent **extends** GrandParent {

**public** **void** home()

{

System.***out***.println("Home method from parent class");

}

**public** **void** car()

{

System.***out***.println("car method from parent class");

}

**public** **void** furniture()

{

System.***out***.println("furniture method from parent class");

}

**public** **static** **void** property()

{

System.***out***.println("Property method from parent class");

}

}

**public** **class** Child **extends** Parent{

**public** **void** bike()

{

home();

car();

System.***out***.println("Bike method from child class");

}

**public** **static** **void** main(String[] args) {

Child c = **new** Child();

c.home();

c.bike();

c.car();

*property*();

c.farm();

Parent p = **new** Parent();

p.home();

}

Output from Child class:

Home method from parent class

Home method from parent class

car method from parent class

Bike method from child class

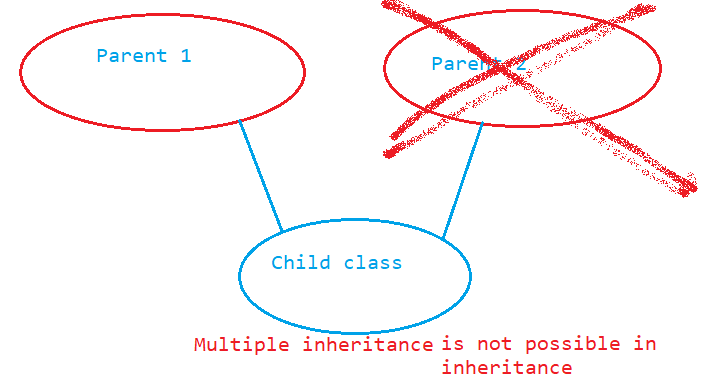
car method from parent class

Property method from parent class

Farm method from grand parent class

Home method from parent class

5. One parent class can have multiple child but a child cannot have multiple parents.



6. Inheritane with respect to variables: Inheritance is applicable to all the variables irrespective of their type(static or non-static).

Example:

**public** **class** Parent **extends** GrandParent {

String s = "non static variable";

**static** String *j* = "Staticvariable";

**public** **class** Child **extends** Parent{

**public** **void** bike()

{

System.***out***.println(s);// accessing non static variable from Parent class

System.***out***.println(**this**.s);// accessing non static variable from Parent class

System.***out***.println(*j*);// accessing static variable from Parent class

}

**public** **static** **void** main(String[] args) {

Child c = **new** Child();

System.***out***.println(c.s);

System.***out***.println(*j*);

}

Super keyword: This is a keyword which is used to access the global variables ( static and non static) of parent class inside the non static area of child class.

Example:

**public** **class** Parent **extends** GrandParent {

String s = "non static variable parent class";

**static** String *j* = "Staticvariable parent class";

}

**public** **class** Child **extends** Parent{

String s = "child class variable";

**static** String *j* = "static variable";

**public** **void** bike()

{

System.***out***.println(s);// accessing non static variable from child class

System.***out***.println(**this**.s);// accessing non static variable from child class

System.***out***.println(*j*);// accessing static variable from Parent class

System.***out***.println(**super**.s);// accessing non static variable from parent class

}

**public** **static** **void** main(String[] args) {

Child c = **new** Child();

Parent p = **new** Parent();

System.***out***.println(c.s);// child class s variable

System.***out***.println(*j*);// child class j variable

System.***out***.println(p.s);// parent class s variable

System.***out***.println(Parent.*j*);// parent class j variable

}

Output:

child class variable

static variable

non static variable parent class

Staticvariable parent class

IS A Relationship: Whenever we are extending a class and acquire the properties of parent class it represents IS A Relationship.Example is Inheritance.

HAS A Relationship: Whenever we create an object of a class and access its method then it represents HAS A relationship.

Example is to create the object and access the properties of other class.

Inheritance with respect to constructor:

Constructor doesn’t follow inheritance but it by-default calls the parent class constructor once we call the child class constructor.

Example:

**public** **class** A {

**public** A()

{

System.***out***.println("A class constructor");

}

}**public** **class** B **extends** A

{

**public** B()

{

System.***out***.println("B class constructor");

}

**public** **static** **void** main(String[] args) {

B b = **new** B();

}

}

Output:

A class constructor

B class constructor

Example 2:

**public** **class** A {

**public** A()

{

System.***out***.println("A class constructor");

}

}

**public** **class** B **extends** A

{

**public** B(**int** i)

{

System.***out***.println("B class constructor");

}

**public** **static** **void** main(String[] args) {

B b = **new** B();

}

}

Note : Here on the creation of object of B class we get an error saying we don’t have zero argument constructor available but it is available in the parent class and it is not accessible here hence constructor doesn’t follow inheritance.

**Through inheritance we have achieved Reusability.**

Assignment :

