**Important points about overriding.**

* If the overridden is **public**, then the overriding method must be **public**; otherwise, a compile-time error occurs.
* If the overridden method is **protected,** then the overriding method must be **protected or public**; otherwise, a compile-time error occurs.
* If the overridden method has **default** (package) access, then the overriding method **must not** be **private**; otherwise, a compile-time error occurs.
* **If the overridden method has Private (package) access, we cant override it.**

**Above overriding rule same for abstract method also.**

**Abstract method can be public/protected/default but not private.**

**because private not inherited.** **The derive class cannot access the private method (even using *super*).**

**Override Private Method:-**

You can't override a private method, but you can introduce one in a derived class without a problem. Private method can be call or access using object of that class also(See Example).The derive class cannot access the private method (even using ***super***)

**public** **class** Base

{

**private** **void** show()

{

System.***out***.println("Parent ");// **can not access using super or object of**  **that class also.**

}

}

**class** MyClass

{

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

Base b1=**new** Base();

**b1.show();//C.T Error**

}

}

**Static and Final Overloading in Derived Class:**

1. A method declared **final** cannot be **overridden and can be reclared**. A method declared static cannot be **overridden** but static **can be re-declared(private method also reclared but can not override)**.

2. We can not call Static method using this() or super() but we can call final method using this() and **super.method\_name(paramer);**

3. First think, Static and final method overloading is happened within class.

4. We can overload static and final method in Derived Class from Base Class.

5. Static and Final both method can be called using Derived class object

Program: 1 Static overload in Derived Class

**public** **class** Base

{

**static** **void** show(**float** x)

{

System.***out***.println("Parent");

}

**static** **void** show(**double** x)

{

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

}

}

**class** Derived **extends** Base

{

**static** **void** show(**int** x)

{

//super.show(10.f); //C.T Errror

System.***out***.println("Base");

}

}

**class** MyClass

{

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

Derived b1=**new** Derived();

b1.*show*(10);

}

Output:Base

Program: 2:**Why static method can not be overridden in Java, because they are resolved during compile time.**  
  
Basically when we use static method same name with same signature in Derived Class then we can see that it override but it not actually override we just introduced new method in Derived Class it already resolved at C.T not R.T.

**public** **class** Base

{

**static** **void** show(**int** x)

{

System.***out***.println("Parent "+x);

}

}

**class** Derived **extends** Base

{

**static** **void** show(**int** x)

{

//super.show(10);//C.T Error

System.***out***.println("Base "+x);

}

}

**class** MyClass

{

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

Base b1=**new** Derived();

b1.*show*(10);

}

}

Output: Parent 10

**Final Method:**

1. Final method can not be Override in Derived Class but overload in derived Class:

2. We can not override final method but we can call from using super();

**public** **class** Base

{

**final** **void** show(**int** x)

{

System.***out***.println("Parent "+x);

}

}

**class** Derived **extends** Base

{

**final** **void** show(**int** x**) //Error**

{

System.***out***.println("Base "+x);

}

}

Program: var-args method is looser than widen.

**public** **class** Base

{

**void** show(**int** a,**int** b,**int** c)

{

System.***out***.println("choose int");

}

**void** show(**byte**... b)

{

System.***out***.println("choose byte ");

}

}

**class** MyClass

{

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

**byte** b=5;

Base b1=**new** Base();

b1.show();

b1.show(b);

b1.show(b,b);

b1.show(b,b,b);

}

}

Ouput

choose byte

choose byte

choose byte

choose int

**Program: if we use runtime polymorphism but not override method then parent class method invoke.**

**See**

**public** **class** Base

{

**void** show()

{

System.***out***.println("Parent");

}

}

**class** Derived **extends** Base

{

//no method override or defind

}

**class** MyClass

{

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

**Base b1=new Derived();//Dynamic Binding**

b1.show();}

}

Output: Parent