Dt : 18/8/2023

faq:

define static method Overriding process?

=>There is no concept of Static method Overriding process in Java,because the static method memories in classes and available in Classes.

faq:

define Method Hiding process?

=>when we have same static method Signature in PClass and CClass,then PClass-Static-Method is hided by the CClass-Static-method while execution process,is known as Method Hiding process.

Note:

=>Method Hiding process means the execution-control cannot reach PClass-Static-method for execution,becuase the same static-method available in CClass and which is executed.

faq:

can we access static members of class using Object reference?

=>Yes,we can access static members of Class using Object reference, because the Object reference belongs to class.

faq:

define Empty-Object-reference?

=>when we create object for the class holding only static-members,then Empty-Object-reference is created.

==========

Note:

=>After Object creation process,we identify the following:

1.Object

2.Object reference

3.Object reference Variable

1.Object:

=>The memory generated to hold Instance members of Class is known as Object.

2.Object reference:

=>The address location where the object is created is known as Object reference.

3.Object reference Variable:

=>The NonPrimitive datatype variable which is holding Object reference

is known as Object reference variable or Object Name.

==========

Ex:

ProjectName : Inheritance_App3

packages,

p1 : PClass.java

```
package p1;
public class PClass {
   public static void m(int x) {
        System.out.println("====PClass static
m(x)====");
        System.out.println("The value x:"+x);
   }
}
```

p1 : CClass.java

```
package p1;
public class CClass extends PClass{
     public static void m(int x) {
           System.out.println("====CClass static
m(x)====");
           System.out.println("The value x:"+x);
     }
}
```

p2 : DemoInheritance3.java(MainClass)

```
package p2;
import p1.*;
public class DemoInheritance3 {
    public static void main(String[] args) {
```

```
        System.out.println("*****ClassName*****");
        CClass.m(123);
        System.out.println("*****ObjectName*****");
        CClass ob = new CClass();
        ob.m(129);
    }
}
```

o/p:

******ClassName******

====CClass static m(x)====

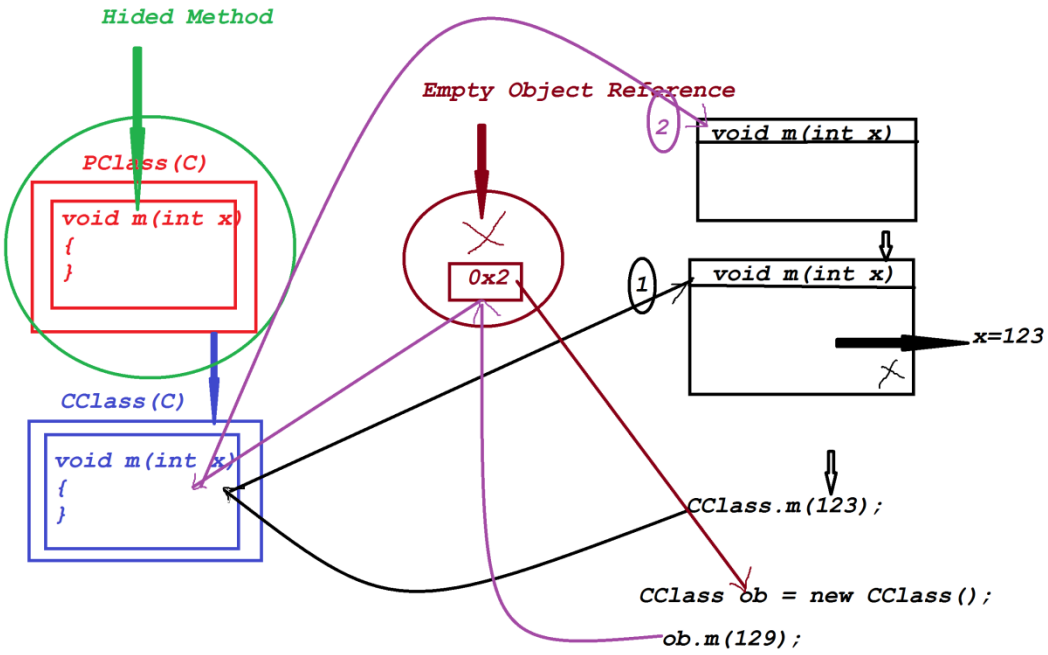
The value x:123

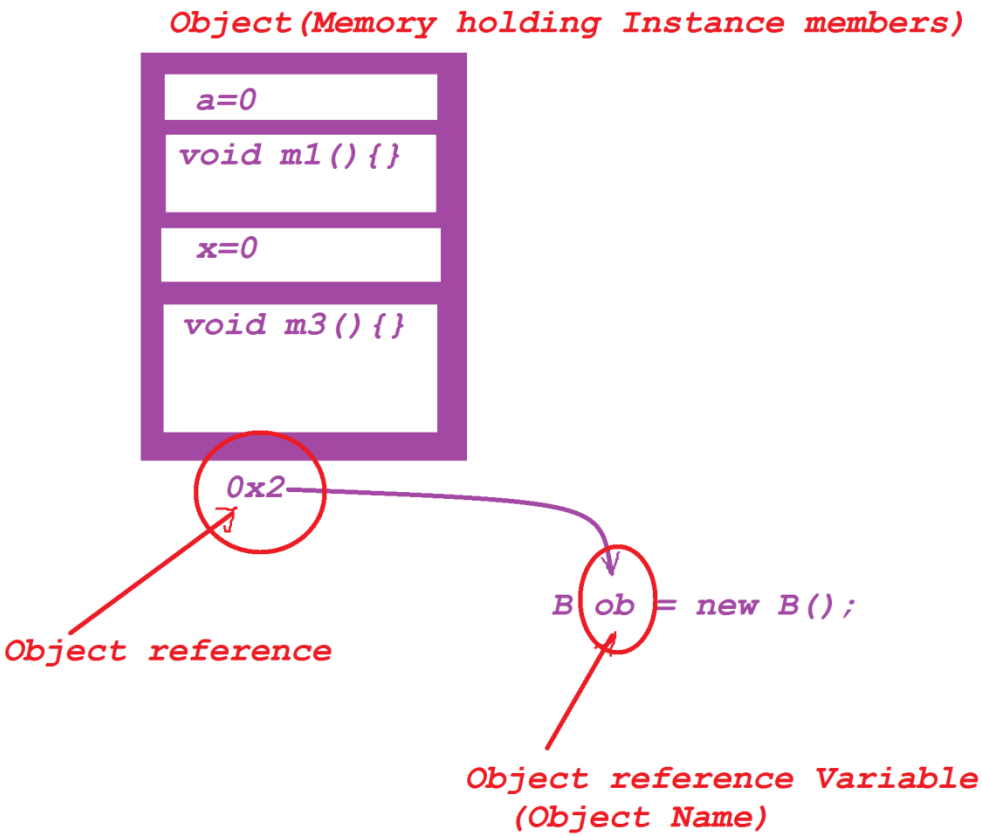
******ObjectName******

====CClass static m(x)====

The value x:129

Diagrams:


Hided Method
Empty Object Reference
PClass(C)
void m(int x)
{
}
CClass(C)
void m(int x)
{
}
0x2
2
1
void m(int x)
void m(int x)
x=123
CClass.m(123);
CClass ob = new CClass();
ob.m(129);



==========

Summary:

(i)Same Instance method signature in PClass and CClass is known as Method Overriding process.

(ii)Same Static method signature in PClass and CClass is known as

Method Hiding Process.

==

Case-2 : Constructors from the PClass/SuperClass

(i)0-parameter constructor from the PClass/SuperClass

=>when we have 0-parameter constructor in PClass then Compiler at Compilation stage will add "super()" to the CClass Constructor and which is internally PClass Con_Call.

ProjectName: Inheritance_App4

packages,

p1 : PClass.java

```
package p1;
public class PClass {
      public PClass() {
       System.out.println("****PClass-Constructor****");
       }
}
```

p1 : CClass.java

```
package p1;
public class CClass extends PClass{
     public CClass() {
      super();
      System.out.println("****CClass-Constructor****");
      }
}
```

p2 : DemoInheritance4.java(MainClass)

```
package p2;
import p1.*;
public class DemoInheritance4 {
     public static void main(String[] args) {
        CClass ob = new CClass();//CClass_Con_Call
     }
}
```

o/p:

*****PClass-Constructor*****

*****CClass-Constructor*****

(ii)Parameterized Constructor from the PClass/SuperClass

=>when we have parameterized constructor in PClass,then we must add "super()" to the CClass constructor to pass parameter to PClass_con.

ProjectName: Inheritance_App5

packages,

p1 : PClass.java

```
package p1;
public class PClass {
      public PClass(int a) {
       System.out.println("****PClass-Constructor****");
       System.out.println("The value a:"+a);
       }
}
```

p1 : CClass.java

```
package p1;
public class CClass extends PClass{
    public CClass(int k) {
     super(k);//PClass_Con_Call
     }
}
```

p2 : DemoInheritance5.java(MainClass)

```
package p2;
import p1.*;
public class DemoInheritance5 {
    public static void main(String[] args) {
       CClass ob = new CClass(123);//CClass_Con_Call
     }
}
```

o/p:

*****PClass-Constructor******

The value a:123

****