***Dt : 27/9/2022***

***Some important methods of java.lang.String:***

***(i)toCharArray()***

***public char[] toCharArray();***

***(ii)startsWith()***

***public boolean startsWith(java.lang.String);***

***(iii)endsWith()***

***public boolean endsWith(java.lang.String);***

***==============================================================***

***Assignment:(Solution)***

***Construct the program for the following layout and analize the***

***output:***

***PClass.java***

***package test;***

***public class PClass {***

***public static int a=10;***

***static {***

***System.out.println("=====PClass Static block====");***

***System.out.println("a:"+a);***

***}***

***public static void m1() {***

***System.out.println("====PClass Static m1()=====");***

***System.out.println("a:"+a);***

***}***

***}***

***CClass.java***

***package test;***

***public class CClass extends PClass{***

***public static int b=20;***

***static {***

***System.out.println("=====CClass Static block====");***

***System.out.println("a:"+a);***

***}***

***public static void m2() {***

***System.out.println("====CClass Static m2()=====");***

***System.out.println("a:"+a);***

***}***

***}***

***DemoInheritance2.java(MainClass)***

***package maccess;***

***import test.\*;***

***public class DemoInheritance2 {***

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

***CClass ob = new CClass();***

***CClass.m1();***

***CClass.m2();***

***}***

***}***

***o/p:***

***=====PClass Static block====***

***a:10***

***====PClass Static m1()=====***

***a:10***

***=====CClass Static block====***

***a:10***

***====CClass Static m2()=====***

***a:10***

***=======================================================***

***faq:***

***can we access static members using Object reference?***

***=>Yes,we can access static members of class using Object reference,***

***because the object reference generated for class and belongs to***

***class.***

***faq:***

***define 'Empty Object reference'?***

***=>The object reference which is generated for a class holding***

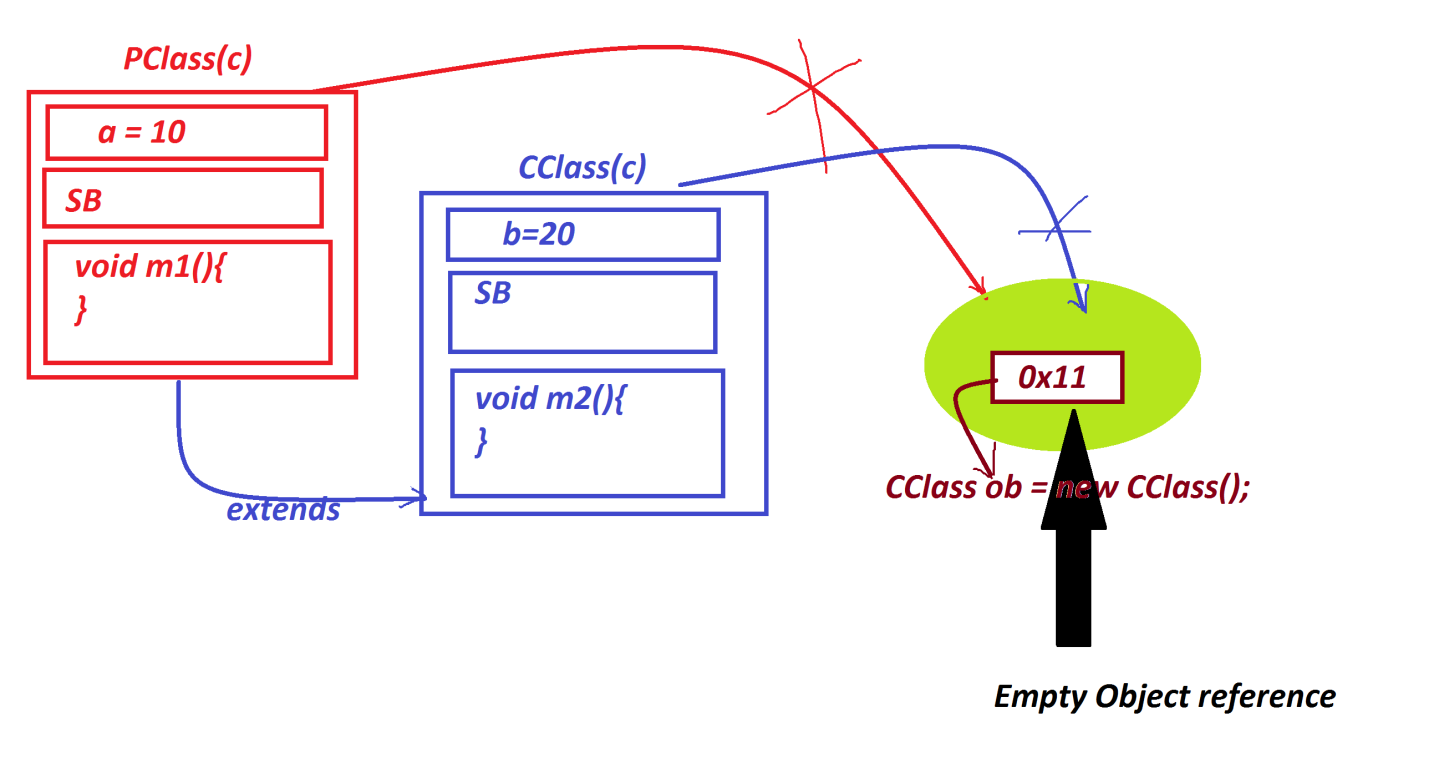
***only static members is known as 'Empty Object reference'.***

***faq:***

***In wt situation we create object for a class?***

***=>when class is holding instance members and if we want to access***

***them,then we create object for a class.***

******

***=================================================================***

***Case-2 : Constructors from the PClass/SuperClass***

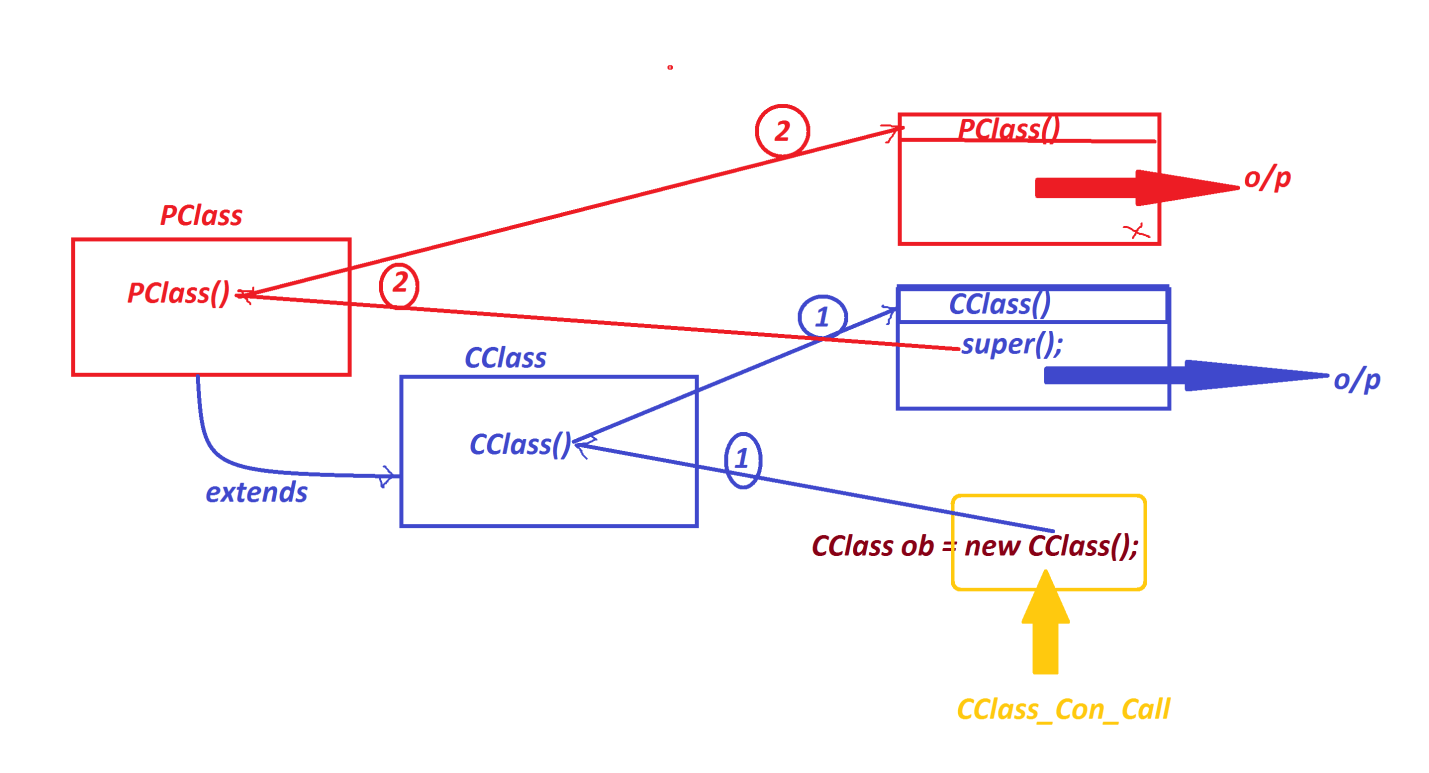
***(i)0-parameter constructor in PClass/SuperClass***

***=>when there is 0-parameter constructor in PClass then the***

***compiler at compilation stage will add "super()" to the CClass***

***constructor and which is PClass\_Con\_Call.***

***Diagram:***

******

***Ex:***

***PClass.java***

***package test;***

***public class PClass {***

***public PClass() {***

***System.out.println("====PClass()=====");***

***}***

***}***

***CClass.java***

***package test;***

***public class CClass extends PClass{***

***public CClass() {***

***System.out.println("====CClass()====");***

***}***

***}***

***DemoInheritance3.java(MianClass)***

***package maccess;***

***import test.\*;***

***public class DemoInheritance3 {***

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

***CClass ob = new CClass();//CClass\_Con\_Call***

***}***

***}***

***o/p:***

***====PClass()=====***

***====CClass()====***

***=============================================================***

***(ii)Parameterized Constructor in PClass/SuperClass***

***=>When there is parameterized constructor in PClass,then we***

***must add "super()" to the CClass constructor to pass parameters to***

***the PClass Constructor.***

***Ex:***

***PClass.java***

***package test;***

***public class PClass {***

***public PClass(int x) {***

***System.out.println("====PClass()=====");***

***System.out.println("The value x:"+x);***

***}***

***}***

***CClass.java***

***package test;***

***public class CClass extends PClass{***

***public CClass(int p) {***

***super(p);//PClass\_Con\_Call***

***}***

***}***

***DemoInheritance4.java(MainClass)***

***package maccess;***

***import test.\*;***

***public class DemoInheritance4 {***

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

***CClass ob = new CClass(123);//CClass\_Con\_Call***

***}***

***}***

***o/p:***

***====PClass()=====***

***The value x:123***

***============================================================***

***Note:***

***=>"super()" is pre-defined format to call constructors from the***

***PClass.***

***faq:***

***define Constructor Chaining process?***

***=>The process of calling constructor from Constructor is known as***

***Constructor Chaining process or Constructor Interlinking process.***

***===============================================================***

***faq:***

***define Method Overriding process?***

***=>The method with same method signature in PClass and CClass,then***

***the PClass method is replaced by CClass method while object creation***

***process is known as Method Overriding process or Method replacement***

***process.***

***=>Same method Signature means,***

***same return\_type***

***same method\_name***

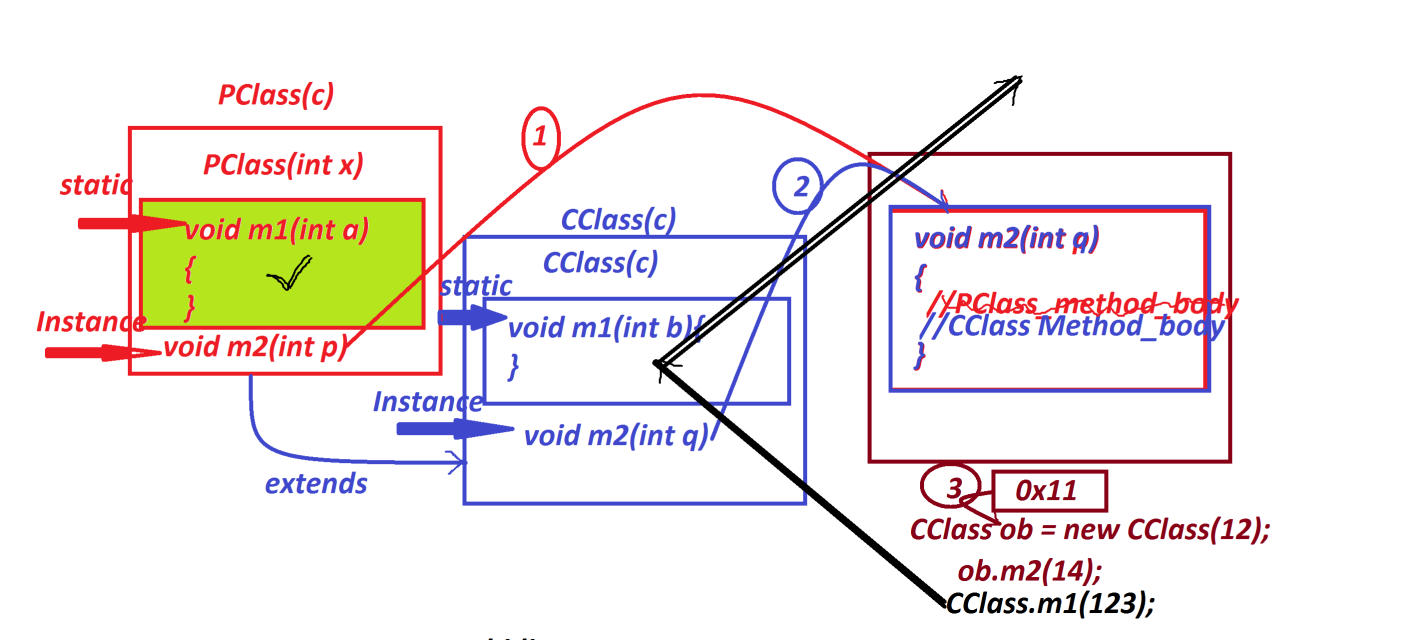
***same para\_list***

***same para\_type***

***Dt : 28/9/2022***

***Ex :***

***Diagram:***

******

***PClass.java***

***package test;***

***public class PClass {***

***public PClass(int x) {***

***System.out.println("====PClass(x)====");***

***System.out.println("The value x:"+x);***

***}***

***public static void m1(int a) {***

***System.out.println("====PClass static m1(a)====");***

***System.out.println("The value a:"+a);***

***}***

***public void m2(int p) {***

***System.out.println("====PClass Instance m2(p)====");***

***System.out.println("The value p:"+p);***

***}***

***}***

***CClass.java***

***package test;***

***public class CClass extends PClass{***

***public CClass(int x) {***

***super(x);***

***}***

***public static void m1(int b) {***

***System.out.println("====CClass static m1(b)====");***

***System.out.println("The value a:"+b);***

***}***

***public void m2(int q) {***

***System.out.println("====CClass Instance m2(q)====");***

***System.out.println("The value q:"+q);***

***}***

***}***

***DemoInheritance5.java(MainClass)***

***package maccess;***

***import test.\*;***

***public class DemoInheritance5 {***

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

***CClass ob = new CClass(12);//CClass\_Con\_Call***

***CClass.m1(123);***

***ob.m2(234);***

***}***

***}***

***o/p:***

***====PClass(x)====***

***The value x:12***

***====CClass static m1(b)====***

***The value a:123***

***====CClass Instance m2(q)====***

***The value q:234***

***=======================================================***

***Note:***

***=>There is no concept of Constructor Overriding process.***

***=>There is no concept of static method Overriding process,because***

***static methods will get memory in classes and available in***

***classes.***

***=>Instance method Overriding is posible,because these instance***

***methods will get the memory within the object while object***

***creation process.***

***==============================================================***

***faq:***

***define Method Hiding process?***

***=>The process in which the execution control cannot reach method***

***for execution is known as Method Hiding process.***

***Note:***

***=>when we have same static method signature in PClass and CClass,***

***then PClass method is replaced by CClass method while execution***

***process.***

***========================================================***

***Summary:***

***=>when we have same instance method signature in PClass and CClass***

***is known as Method Overriding process.***

***=>when we have same static method Signature in PClass and CClass is***

***known as Method Hiding process.***

***===========================================================***

***faq:***

***define Method Overloading process?***

***=>More than one method with same method name but differentiated***

***by their para\_list or para\_type is known as Method Overloading***

***process.***

***case-1 : Constructor Overloading process***

***=>More than one constructor differentiated by their para\_list or***

***para\_type is known as Constructor Overloading process.***

***Ex:***

***Display.java***

***package test;***

***public class Display***

***{***

***public Display(int x,int y)***

***{***

***this(x);//Con\_call\_from\_same\_call***

***System.out.println("====Display(x,y)====");***

***System.out.println("The value y:"+y);***

***}***

***public Display(int x)***

***{***

***System.out.println("====Display(x)====");***

***System.out.println("The value x:"+x);***

***}***

***}***

***DemoInheritance6.java(MainClass)***

***package maccess;***

***import test.Display;***

***public class DemoInheritance6 {***

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

***Display d2 = new Display(13,14);//Con\_with\_2\_para***

***}***

***}***

***o/p:***

***====Display(x)====***

***The value x:13***

***====Display(x,y)====***

***The value y:14***

***=======================================================***

***faq:***

***wt is the diff b/w***

***(i)super()***

***(ii)this()***

***(i)super():***

***=>super() is used to execute constructors from the PClass or***

***SuperClass.***

***(ii)this():***

***=>this() is used to execute constructors from the Same class or***

***Current running class.***

***=========================================================***