Dt: 28/12/2020

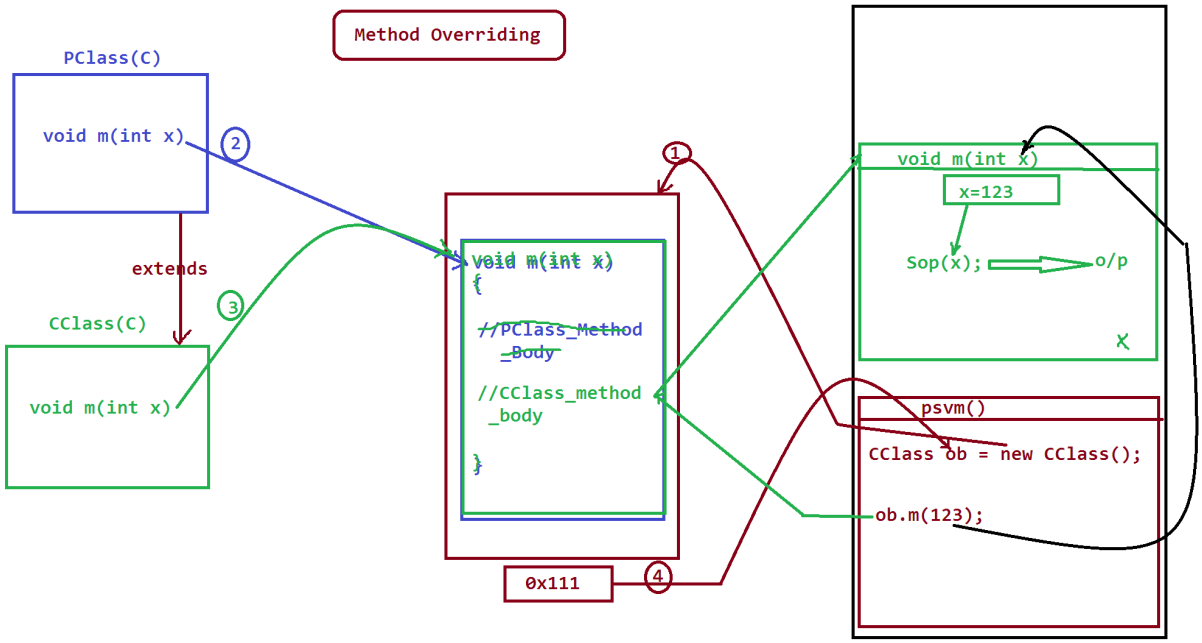
Execution flow of above program:

ClassFiles:

PClass.class

CClass.class

Inheritance2.class



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.

Exp program:

```
class PClass
{
	void m(int x)
	{
System.out.println("====PClass===");
System.out.println("The value x:"+x);
	}
}
class CClass extends PClass
{
	void m(int y,int z)
	{
System.out.println("====CClass===");
System.out.println("The value y:"+y);
System.out.println("The value z:"+z);
	}
}
class Inheritance3 //MainClass
{
	public static void main(String[] args)
	{
CClass ob = new CClass();//Normal Inheritance
ob.m(12);
ob.m(13,14);
	}
```

```
}
```

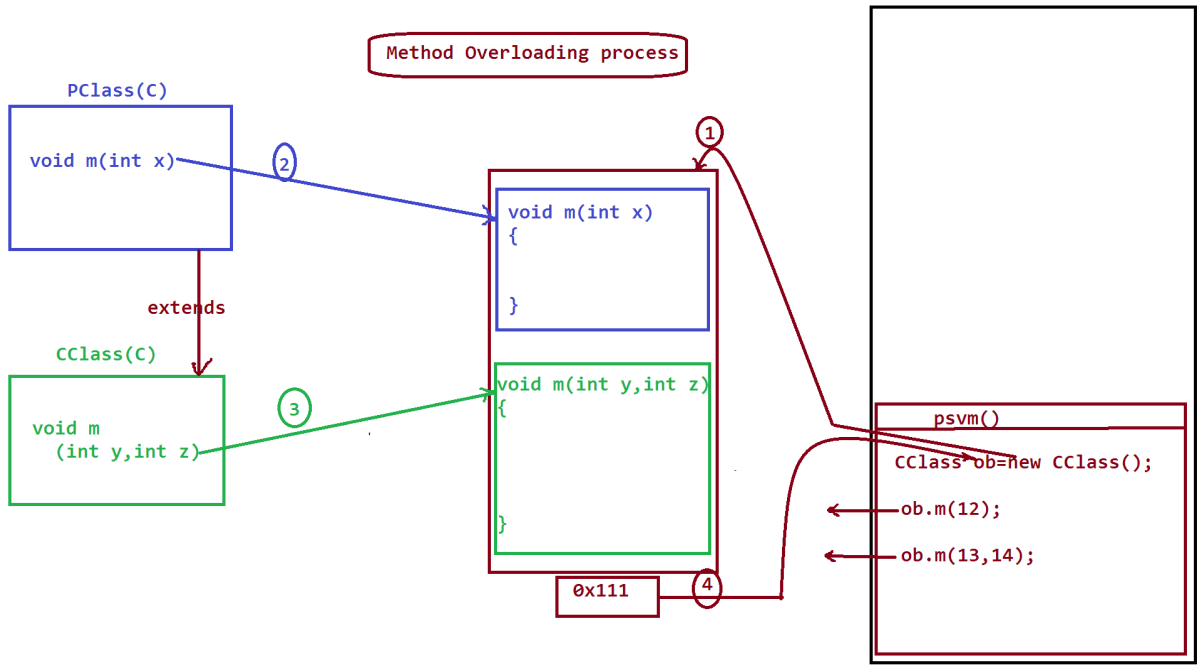
Execution flow of above program:

ClassFiles:

PClass.class

CClass.class

Inheritance3.class



faq:

define Generalization process?

=>The process in which one reference is created and the reference is binded with all the members of PClass and only Overriding members from the CClass is known as Generalization process.

=>We use the folllowing syntax to acheive Generalization process:

PClass ob = new CClass();

Exp program:

```
class PClass
{
	void m1()//Overrided method
	{
System.out.println("====PClass m1()====");
	}
	void m2()
	{
System.out.println("====PClass m2()====");
	}
}
class CClass extends PClass
{
	void m1()//Overriding method
	{
System.out.println("====CClass m1()====");
	}
	void m3()
	{
System.out.println("====CClass m3()====");
	}
}
class Inheritance4 //MainClass
```

```
{
        public static void main(String[] args)
        {
PClass ob = new CClass();
ob.m1();
ob.m2();
//ob.m3();//Compilation Error
        }
}
```

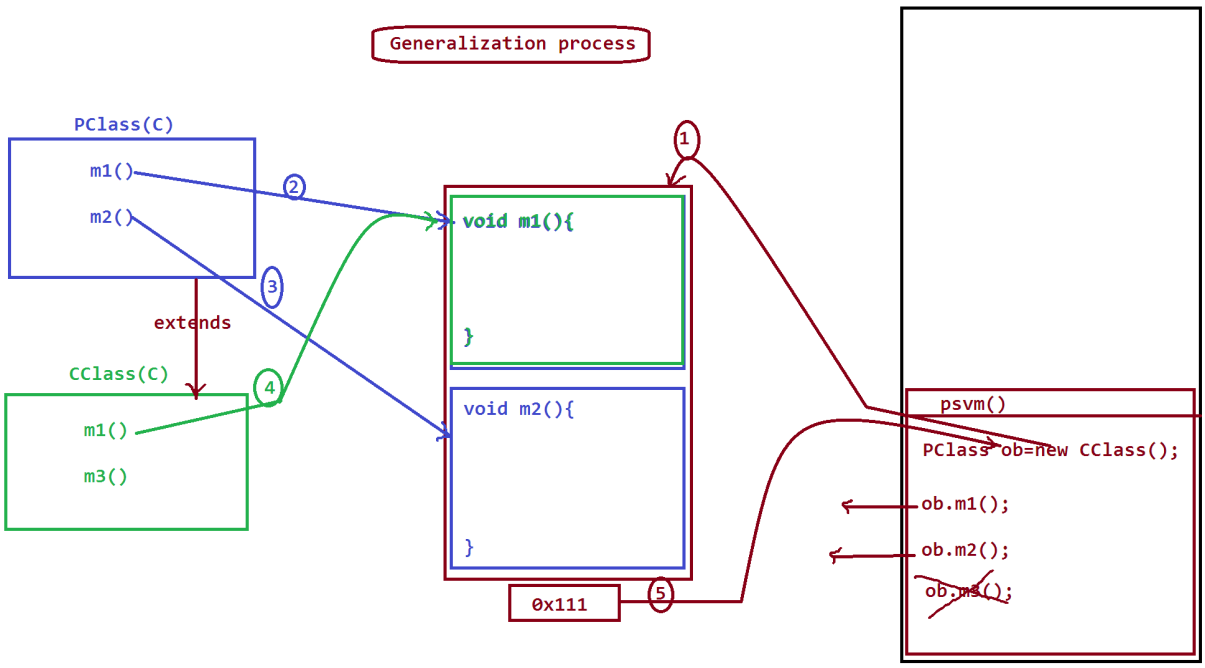
Execution flow of above program:

ClassFiles:

PClass.class

CClass.class

Inheritance4.class



Dt : 29/12/2020

Note:

=>This Generalization process can be acheived by declaring the CClass without name known as "Anonymous InnerClass as Class Extention".

syntax:

```
class PClass
{
 //PClass_body
}
PClass ob = new PClass()
{
 //CClass_body
};
```

Exp program:

```
class PClass
{
        void m1()//Overrided method
        {
System.out.println("====PClass m1()====");
        }
        void m2()
        {
System.out.println("====PClass m2()====");
        }
```

```
}
class Inheritance5 //MainClass
{
        public static void main(String[] args)
        {
PClass ob = new PClass()
{
        void m1()//Overriding method
        {
System.out.println("====CClass m1()====");
        }
        void m3()
        {
System.out.println("====CClass m3()====");
        }
};

ob.m1();
ob.m2();
//ob.m3();//Compilation Error
        }
}
```

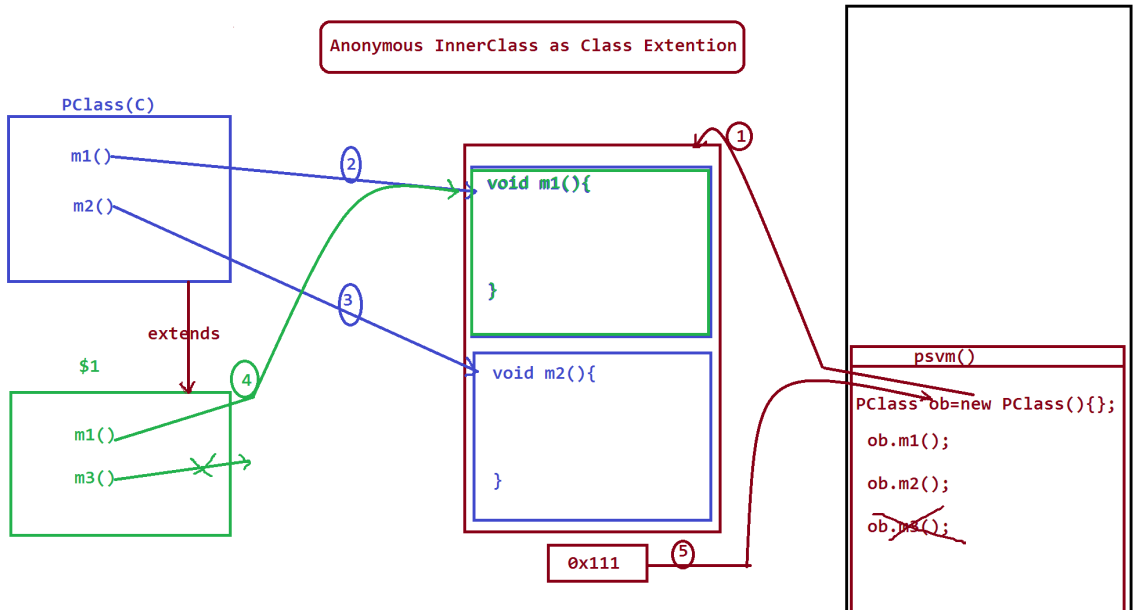
Execution flow of above program:

ClassFiles:

PClass.class

Inheritance5.class(MainClass)

Inheritance5$1.class



Summary of Inheritance:

Model-1 : Normal Inheritance

CClass ob = new CClass();

Model-2 : Generalization process

PClass ob = new CClass();

Model-3 : Anonymous InnerClass as Class extention.(Generalization)

PClass ob = new PClass(){};

---

faq:

wt is the diff b/w

(i)super

(ii)this

(i)super:

=>'super' keyword is used to access the variables and methods from the PClass or SuperClass.

(ii)this:

=>'this' keyword is used to access the variables and methods from the Same class.

Note:

=>'super' and 'this' keywords are used when we have same variables and methods in PClass and CClass.

Exp program:

```
import java.util.Scanner;
class PClass
{
	int a;
	void m()
	{
System.out.println("===PClass m()===");
System.out.println("The value a:"+a);
	}
}
class CClass extends PClass
{
	int a;
```

```
	void m()
	{
System.out.println("===CClass m()===");
System.out.println("The value a:"+a);
	}
	void load(int x,int y)
	{
		super.a=x;
		this.a=y;
	}
	void dis()
	{
		super.m();
		this.m();
	}
}
class Inheritance6 //MainClass
{
	public static void main(String[] args)
	{
Scanner s = new Scanner(System.in);
CClass ob = new CClass();
System.out.println("Enter the value for PClass:");
int x = s.nextInt();
System.out.println("Enter the value for CClass:");
```

```
int y = s.nextInt();
ob.load(x,y);
ob.dis();
        }
}
```
