**Dt : 21/12/2020**

**Note:**

**=>break statement is used to stop the switch-case statement and also used to stop the Iterative statements like while and for.**

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

**Exp program2(DRef2.java)**

```
import java.util.Scanner;
class Address //SubClass
{
        String hNo,sName,city;
        int pinCode;
        void getAddress()
        {
System.out.println
        ("hNo:"+hNo+"\nsName:"+sName+"\ncity:"+city+"\npinCode:"+pinCode);
        }
}
class Employee//Subclass
{
        String name,id;
        Address ad = new Address();
        void getEmployee()
        {
System.out.println("Name:"+name+"\nid:"+id);
      ad.getAddress();
```

```
        }
}
class Read //SubClass
{
        void read(Scanner s,Employee e)
        {
System.out.println("Enter the name:");
e.name = s.nextLine();
System.out.println("Enter the Id:");
e.id = s.nextLine();
System.out.println("Enter the hNo:");
e.ad.hNo = s.nextLine();
System.out.println("Enter the sName:");
e.ad.sName = s.nextLine();
System.out.println("Enter the city:");
e.ad.city = s.nextLine();
System.out.println("Enter the pinCode:");
e.ad.pinCode = s.nextInt();
        }
}
class Display //SubClass
{
        void dis(Employee e)
        {
                e.getEmployee();
```

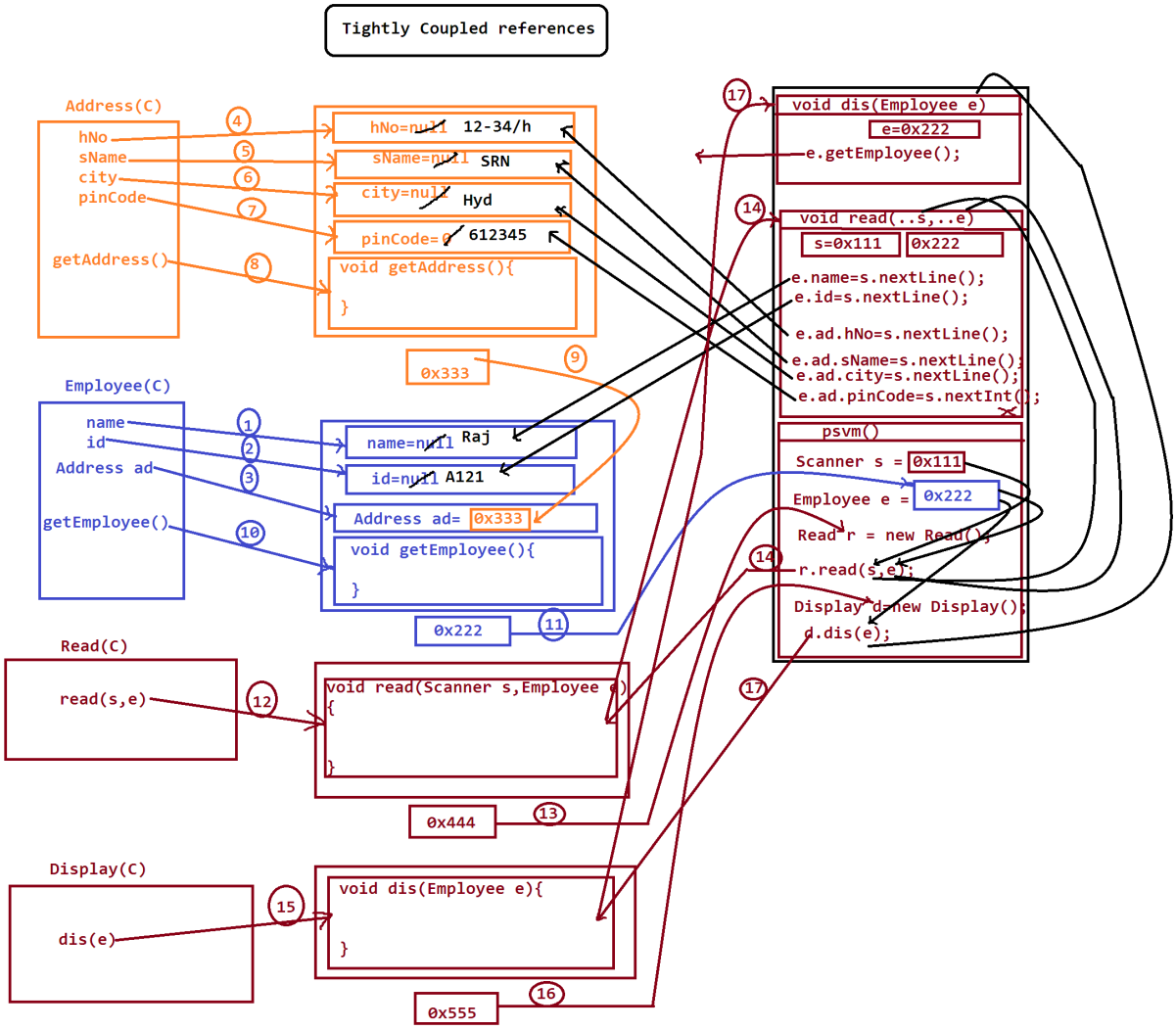
```
        }
}
class  DRef2 //MainClass
{
        public static void main(String[] args)
        {
Scanner s = new Scanner(System.in);
Employee e = new Employee();
Read r = new Read();
r.read(s,e);//method call
Display d = new Display();
d.dis(e);
        }
}
```

**Dt : 23/12/2020**

**Execution flow of above program:**

**ClassFiles:**

- **Address.class**
- **Employee.class**
- **Read.class**
- **Display.class**
- **DRef2.class(MainClass)**




**=>References in Java are categorized into two types:**

**1.Tightly Coupled references**

**2.Loosly Coupled references.**

**1.Tightly Coupled references:**

**=>In Tightly Coupled references the objects which are linked together are dependent objects.**

**Exp:**

**DRef2.java**

**Note:**

**=>In DRef2.java while Employee class object creation,the Address class object is created.which means Address class object is depending on Employee class object.**

**2.Loosly Coupled references.:**

**=>In Loosly Coupled references the Objects which are linked together are Independent objects.**

**Exp:**

**DRef1.java**

**Note:**

**=>In DRef1.java the Balance class Object can be created without creating WithDraw class object,which means Balance Class Object is independent from WithDraw class object.**

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

**Note:**

**=>Through references concept we can link the objects.**

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

***imp**

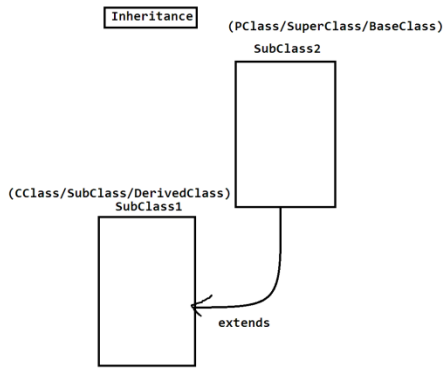
**Inheritance in Java:**

**=>The process of linking two classes using 'extends' keyword is known as Inheritance process.**

**Digram:**

**syntax:**

```
class SubClass2
{
 //members
}
class SubClass1 extends SubClass2
{
 //members
}
```




**=>In Inheritance process through 'extends' keyword the members of one class are available to another class.**

**=>In Normal Inheritance process we always create object for CClass as**

follows:

CClass ob = new CClass();

---

Inheritance Case-1:

NonStatic members from the PClass or SuperClass

Exp program:

```
class SubClass2 //PClass
{
	int b;
	void m2()
	{
System.out.println("===PClass m2()===");
System.out.println("The value b:"+b);
System.out.println("The value a:"+a);
	}

	{
System.out.println("===PClass NonStatic block===");
	}
}
class SubClass1 extends SubClass2 //CClass
{
	int a;
```

```
        void m1()
        {
System.out.println("===CClass m1()===");
System.out.println("The value a:"+a);
System.out.println("The value b:"+b);
        }

        {
System.out.println("===CClass NonStatic block===");
        }
}
class Inheritance1 //MainClass
{
        public static void main(String[] args)
        {
SubClass1 ob = new SubClass1();//Normal Inheritance process
ob.b=13;
ob.a=12;
ob.m2();
ob.m1();
        }
}
```

**Dt : 26/12/2020**

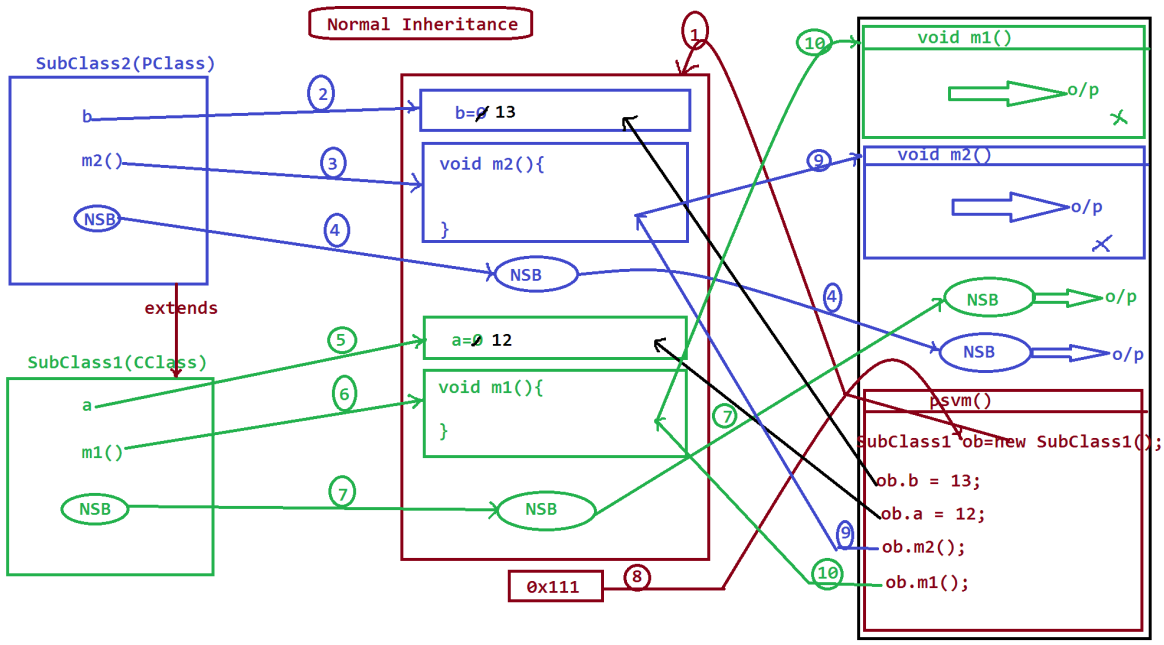
**Execution flow of above program:**

**ClassFiles:**

**SubClass2.class**

**SubClass1.class**

**Inheritance1.class**




**Note:**

**=>In normal Inheritance process one reference is created and the ref is binded with all the members of PClass and all the members of CClass.**

**=>In Inheritance process the PClass is loaded onto MethodArea first and then the CClass is loaded.**

**=>In Inheritance process while Object creation the PClass members are binded first and then the CClass members are binded.**

**=>In Inheritance process through 'extends' keyword the PClass members are available to CClass,in this process the CClass can access all the members of PClass,but the PClass cannot access the members of CClass.**

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

**faq:**

**define Method Overriding process?**

**=>The method with same method signature in PClass and CClass,then PClass method is replaced by the CClass method while object creation is known as Method Overriding process.**

**Same method Signature means**

- **same return_type**
- **same method_name**
- **same para_list**
- **same para_type**

**Exp program:**

```
class PClass
{
        void m(int x)//Overrided method
        {
System.out.println("===PClass==");
System.out.println("The value x:"+x);
        }
}
class CClass extends PClass
{
        void m(int x)//Overriding method
        {
```

```
System.out.println("===CClass===");
System.out.println("The value x:"+x);
        }
}
class Inheritance2 //MainClass
{
        public static void main(String[] args)
        {
CClass ob = new CClass();//Normal Inheritance process
ob.m(123);
        }
}
```