Concept of Inheritance Java and Java Polymorphism

Inheritance in java

- When a "Is-A" relationship exists between two classes we use Inheritance
- The parent class is termed super class and the inherited class is the sub class>
- The keyword extend java is used by the sub class to inherit the features of super class
- Inheritance is important since it leads to reusability of code

Inheritance Java Example

```
class Doctor
// Instance Variables and Methods for the Doctor Class
}

class Surgeon extends Doctor{
// Inherits instance variables & methods of the doctor class
//may have variables and methods of its own.
}
```

Method Overriding

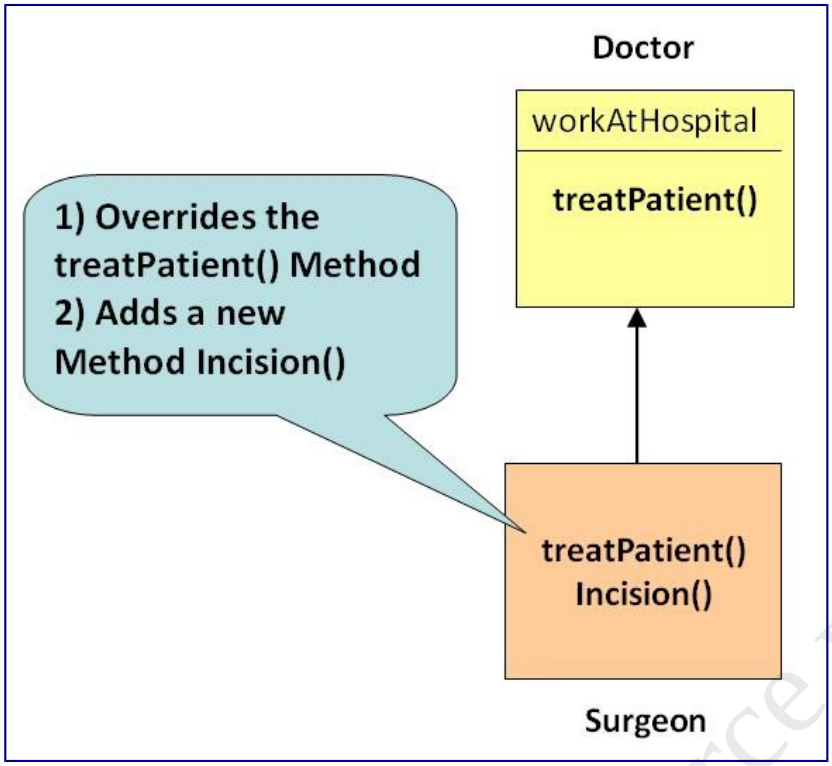
Redefining a super class method in a sub class is called method overriding

Rules for Method Overriding

- The method signature i.e. method name, parameter list and return type have to match exactly.
- The overridden method can widen the accessibility but not narrow it, i.e. if it is private in the base class, the child class can make it public but not vice versa.

Example

```
Doctor doctorObj = new Doctor()
doctorObj.treatPatient()
// treatPatient method
// in class Doctor will be executed
Surgeon surgeonObj = new Surgeon();
surgeonObj.treatPatient()
// treatPatient  method
// in class Surgeon  will be executed
```



Dynamic Polymorphism Java

A reference variable of the super class can refer to a sub class object

```
Doctor obj = new Surgeon();
```

Consider the statement

```
obj.treatPatient();
```

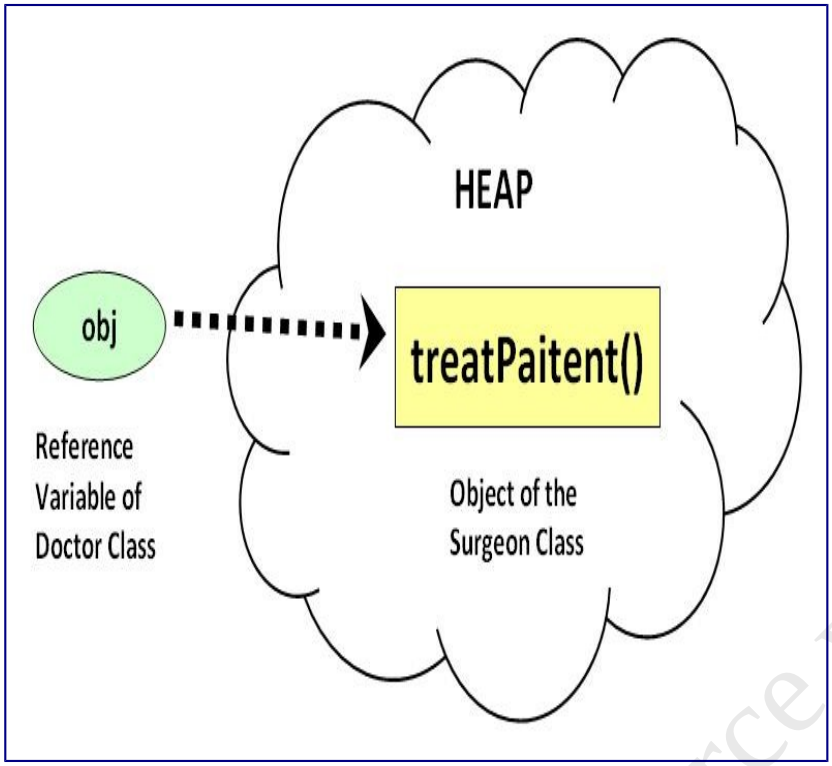
Here the reference variable "obj" is of the parent class , but the object it is poiting to is of the child class (as show in diagram).

obj.treatPatient() will execute treatPatient() method of the sub-class - Surgeon

If a base class reference is used to call a method, the method to be invoked is decided by the JVM, depending on the object the reference is pointing to

For example, even though obj is a reference to Doctor, it calls the method of Surgeon, as it points to a Surgeon object

This is decided during run-time and hence termed ***dynamic*** or ***run-time polymorphism***



Super

What if the treatPatient method in the Surgeon class wants to do the functionality defined in Doctor class and then perform its own specific functionality?

In this case keyword **super** can be used to access methods of the parent class from the child class.

The treatPatient method in the Surgeon class could be written as:

```
treatPatient(){
   super.treatPatient();
     //add code specific to Surgeon
}
```

The keyword super can be used to access any data member or methods of the super class in the sub class.

Assignment:-To learn Inheritance , Polymorphism & super keyword

Step 1) Copy the following code into an Editor

```
public class Test{
     public static void main(String args[]){
        X x= new X();
       Y y = new  Y();
       y.m2();
      //x.m1();
      //y.m1();
     //x = y;// parent pointing to object of child
     //x.m1() ;
```

```
     //y.a=10;
   }

}
class X{
   private int a;
   int b;
      public void m1(){
       System.out.println("This is method m1 of class X");
     }
}

class Y extends X{
      int c; // new instance variable of class Y
         public void m1(){
            // overriden method
            System.out.println("This is method m1 of class Y");
        }
       public void m2(){
           super.m1();
           System.out.println("This is method m2 of class Y");
      }
}
```

Step 2) Save , Compile & Run the code. Observe the output.

Step 3) Uncomments lines # 6-9. Save , Compile & Run the code. Observe the output.

Step 4) Uncomment line # 10 . Save & Compile the code.

Step 5) Error = ? This is because sub-class can not access private members of the super class.

Difference between Overloading and overriding

Method overloading: Method overloading is in the same class , where more than one method have the same name but different signatures

Ex

```
void sum (int a , int b);
void sum (int a , int b, int c);
void sum (float a, double b);
```

Method overriding : Method overriding is when one of the **methods** in the **super class** is **redefined** in the **sub-class**. In this case the signature of the method remains the same.

Ex

```
class X{
  public int sum(){
     // some code
  }
}

class Y extends X{
  public int sum(){
```

```
    //overridden method
   //signature is same
 }
}
```

Difference between static & Dynamic polymorphism

Static Polymorphism : It relates to method overloading .Errors ,if any, are resolved at compile time. Since the code is not executed during execution , the name static.

Ex:

```
void sum (int a , int b);
void sum (float a, double b);
int sum (int a, int b); //compiler gives error.
```

Dynamic Polymorphism: It relates to method overriding.

In case a reference variable is calling an overridden method, the method to be invoked is determined by the object ,your reference variable is pointing to. This is can be only determined at run -time when code in under execution , hence the name dynamic.