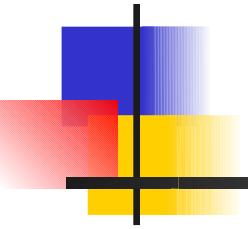
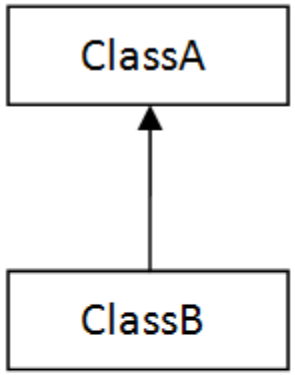


Network Programming

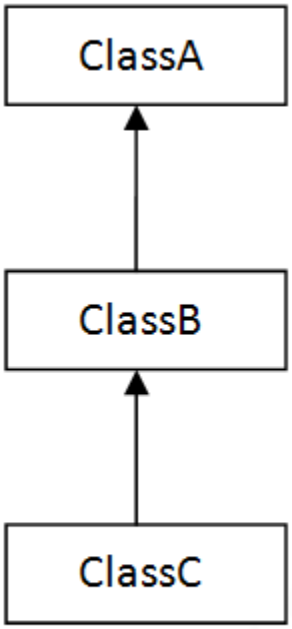
Java – 2

Practice Programming

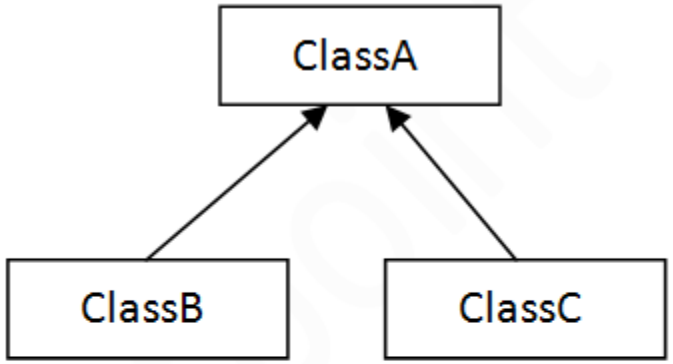
Types of inheritance in java



1) Single

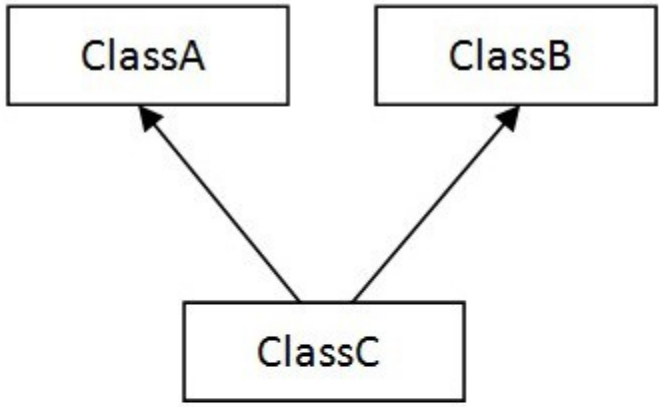


2) Multilevel

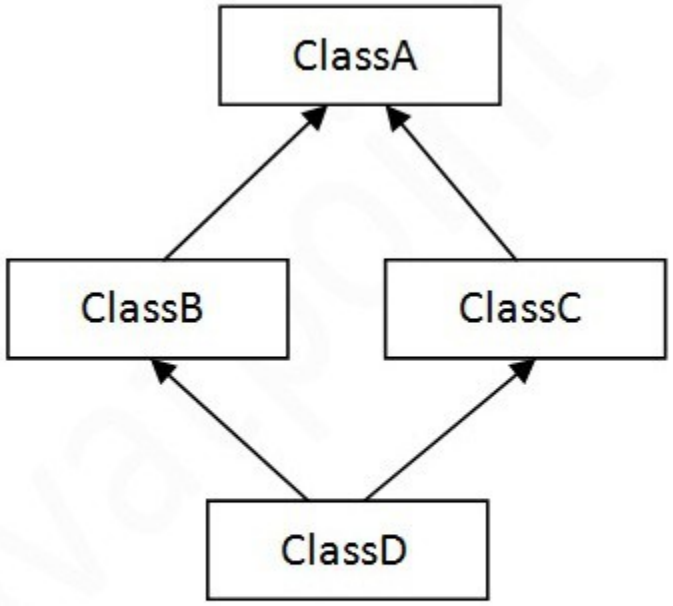


3) Hierarchical

Types of inheritance in java



4) Multiple



5) Hybrid

Example

```
public class Person  {
        String name;
        Person(String n)  {
                name = "Person:  " + n;
        }
 }
class Mother extends Person  {
        Mother(String n)  {
                 Person(n);              // super(n);
                name = "Mother:  " + n;
        }
        void FeedChildren( )  {
                System.out.println(name + " is feeding the kids ...");
        }
}
class Wife extends Person  {
        Wife(String n)  {
                 Person(n);              // super(n);
                name = "Wife:  " + n;
        }
        void CallHusband( )  {
                System.out.println(name + " is calling the husband ...");
        }}
```

Example

```
public class Test  {
        public static void main(String args[])  {
                Person p = new Person("Patty");
                Mother m = new Mother("Mary");
                Wife w = new Wife("Wilma");
                System.out.println("p is a " + p.name);
                System.out.println("m is a " + m.name);
                System.out.println("w is a " + w.name);
                m.FeedChildren();
                w.CallHusband();
        }
}
```

```
C:> java Test
p is a  Person:  Patty
m is a Mother: Mary
w is a Wife:  Wilma
```

Example

```
public class Person  {
        String name;
        Person(String n)  {
                name = "Person:  " + n;
        }
 }
interface Mother   {
        void FeedChildren( ) ;
}
interface  Wife   {
        void CallHusband( )  ;
}
class WifeAndMother extends Person implements Wife, Mother  {
        WifeAndMother(String n)  {
                super(n);
                name = "Wife and mother:  " + n;
        }
        public void FeedChildren()  {
                System.out.println(name + " is feeding the children.");
        }
        public void CallHusband()  {
                System.out.println(name + " is calling her husband.");
        }
}
```

Example

```
public class Test  {
        public static void main(String args[])  {
                Person p = new Person("Patty");
                WifeAndMother w = new WifeAndMother("Wendy");
                System.out.println("p is a " + p.name);
                System.out.println("w is a " + w.name);
                w.FeedChildren( );
                w.CallHusband( );
        }
}
```

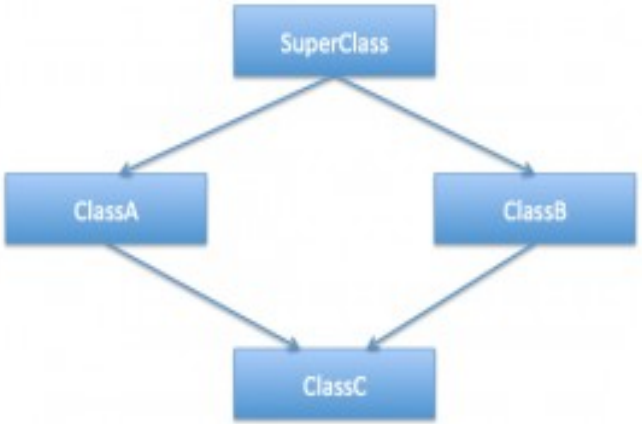
```
C:> java Test
p is a  Person:  Patty
w is a  Wife and mother: Wendy
Wendy is feeding the children.
Wendy is calling her husband.
```

```
public class ChainingDemo {
        public ChainingDemo(){
                        System.out.println("Default constructor");
        }
        public ChainingDemo(String str){
                        this();
                        System.out.println("single param");
        }
        public ChainingDemo(String str, int num){
                        this("Hello");
                        System.out.println("double args");
        }
        public ChainingDemo(int num1, int num2, int num3){
                        this("Hello", 2);
                        System.out.println("three args");
        }
        public static void main(String args[]){
                        ChainingDemo obj = new ChainingDemo(5,5,15);
        }
}
```

```
C:> java ChainingDemo
        Default constructor
        single param
        double args
        three args
```

```
public abstract class SuperClass {
        public abstract void doSomething();
}
public class ClassA extends SuperClass{
        @Override
        public void doSomething(){
                System.out.println(" do A ");
        }
        public void methodA() { }
}
public class ClassB extends SuperClass{
        @Override
        public void doSomething(){
                System.out.println(" do B ");
        }
        public void methodB() { }
}
public class ClassC extends ClassA, ClassB{
        public void test(){
                //calling super class method
                doSomething();
        }
}
```

Diamond Problem



Multiple Inheritance in Interfaces

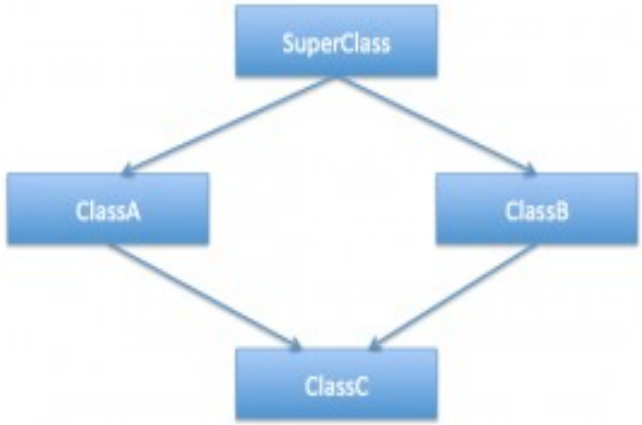
```
public interface InterfaceA {
        public void doSomething();
}

public interface InterfaceB {
        public void doSomething();
}

public interface InterfaceC extends InterfaceA, InterfaceB {
        public void doSomething();
}
```

Multiple Inheritance in Interfaces

```
public class InterfacesImpl implements InterfaceA, InterfaceB, InterfaceC {
        @Override
        public void doSomething(){
                System.out.println("display something");
        }
        public static void main(String[] args) {
                InterfaceA objA = new InterfacesImpl();
                InterfaceB objB = new InterfacesImpl();
                InterfaceC objC = new InterfacesImpl();

                objA.doSomething();          //all the method calls
                objB.doSomething();          //below are going
                objC.doSomething();          //to same concrete
        }                                    //implementation

}
```

Composition

```
public class ClassC {

        ClassA objA = new ClassA();
        ClassB objB = new ClassB();

        public void test(){
                objA.doSomething();
        }
        public void methodA(){
                objA.methodA();
        }
        public void methodB(){
                objB.methodB();
        }
}
```



Composition vs Inheritance

- Best practices of java programming is to “favor composition over inheritance”.
  - Multiple level of class inheritance and superclass may not be controlled by us. “ third party”
  - Exposing all the superclass methods to the client. “ security holes”
  - the method invocation not flexible. We can make the method invocation flexible and make it dynamic. “compile time binding”

References

- **Inheritance in Java**
- http://www.javatpoint.com/inheritance-in-java

- **What is Inheritance in Java Programming?**
- http://beginnersbook.com/2013/03/inheritance-in-java/

- **What Is an Interface?**
- https://docs.oracle.com/javase/tutorial/java/concepts/interface.html

- **What is the difference between abstract class and interface?**
- http://www.javatpoint.com/difference-between-abstract-class-and-interface

References

- **Inheritance in Java**
- http://www.javatpoint.com/inheritance-in-java