Interfaces in Java

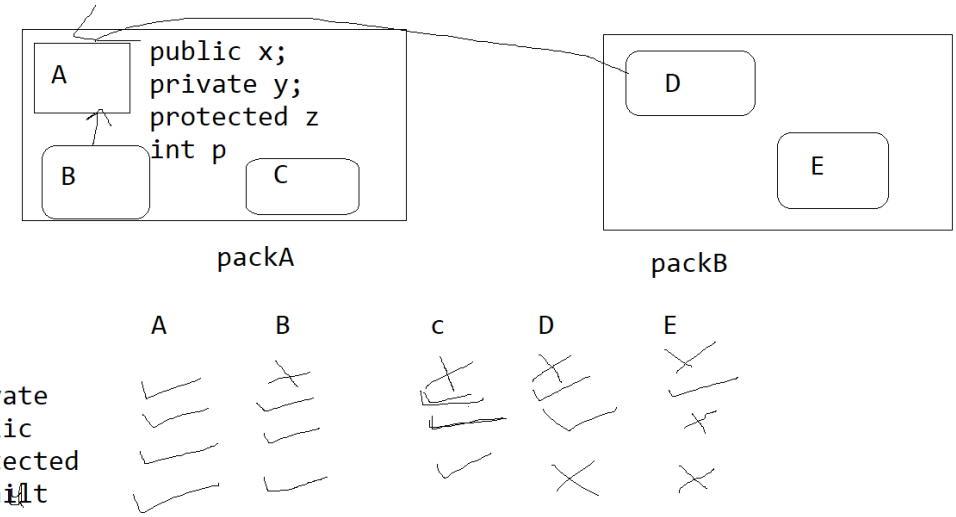
- Every function in interfaces is by default public and abstract.
- Every variable in interfaces is by default public static final.
- In interface If you want to write function definition, then the function must be default. This facility is added in version 8.
- In interface you may write static methods, it is also added in version 8.
- One interface can extend any number of interfaces, but one class can extend only one class but can implements many interfaces.
- Interface is a contract between the class and interface.

Interfaces	Abstract classes
It does not represent ISA relation	It represents ISA relation with child.
All the variables are by default public static final	Members can be public, protected, private or static or final
You cannot add constructor in the interface	You can add constructor in the abstract class
You cannot override methods of Object class	You can override methods of Object class
One interface can extend many interfaces	But one class can extend only one class, but can implements any number of interfaces

Packages

In java packages are like folders in OS.

1. It helps to keep related files together; hence organization is better.
2. Importing related classes becomes easy.
3. Naming collision can be avoided.



What is functional Interface?

- Any interface which has only one abstract function then it is functional interface
- A functional interface may have some default functions, and some static functions also, but should have only one abstract function.

If you have a functional interfaces, and if you want to override only abstract method, then you may use lambda function

//@FunctionalInterface
public interface MyInterface {
int compare(int x,int y);
default void f11() {
System.out.println("in f11");
}
public static void m1() {
 	
System.out.println("in m1 static method");
}
 	
}
	Class TestInterface{
	p.s v main(String[] args){
	MyInterface ob=(x,y)->{return x>y?x:y;};
	System.out.println(ob.compare(12,15));
	}

What is generics?

- If to a function / interface/class. You are passing data type as parameter then it is called as generics.
- It increases reusability of the code.

package com.demo.interfaces; public interface MyGenericInterface<T> { T compare(T x,T y);	MyGenericInterface<Integer> ob1=(x,y)->{return x>y?x:y;}; MyGenericInterface<String> ob2=(x,y)->{int s=x.compareTo(y); if(s<0)

}	return x; else return y;};