Interfaces

1. All methods in interfaces are by default public and abstract.
2. variables in the interface are by default public static and final, these variable donot get inherited, to use them use interfacename.variablename
3. we can write static methods in interface, these methods do not get inherited, to call these methods use interfacename.methodname
4. We can add private methods in interface, (from java 1.9 onward)
5. To add method with implementation in interface is allowed from java 1.8 onward, but the method should be default.
6. default methods can be overridden, to call interface default method in the class.
   I1.super.m11();
7. One class can extend only one class but can implement multiple interfaces.
8. One interface can extend multiple number of interfaces.
9. In inheritance if the access specifier in parent is protected, then in child it can be public. but vice versa is not true
10. If on class implements more than one interface, and both interface has default method with same name, then it is mandatory to override default method.

class A{
    abstract protected void m1();
}
class B extends {
   public void m1(){
    ....
  }
}
This is allowed.
	class A{
	    abstract protected void m1();
	}
	class B extends {
	   protectd void m1(){
	    ....
	  }
	}
	This is allowed.
		class A{
		    abstract public void m1();
		}
		class B extends {
		   protected void m1(){
		    ....
		  }
		}
		This is not allowed.

differences between Interfaces and Abstract classes

Abstract class	Interfaces
we can write methods with implementation	method with implementation has to be default method
Constructor can be written in abstract class	We cannot write constructor in interface
Better to use when ISA relationship is there	It is a contract between interface and the class
We can extend only one class	we can imlements more than on interfaces One intreface can extend more than one interface
we can declare members in the class	variables are by default public static final
we can use private, protected, public and default access specifiers	we can use only public or private access specifiers.
we can override Object class methods	We cannot override Object class method

Rule :
In inheritance if you want to call any child specific method then use explicit typecasting for parent reference.

Employee e=new SalariedEmp(...........)
((SalariedEmp)e)method4() --→ method4 is only in SalariedEmp, not in Employee class

Eaxmaple

```
class A

Class B extends A implements I1

class C extends B
```

A ob=new C();
System.out.println(ob instanceof A); ---→ True
System.out.println(ob instanceof B); ---→ True
System.out.println(ob instanceof C); ---→ True

Why to use packages(Advantages

1. Better organization of classes, all related classes can be in the same package
2. Importing all related classes is easy
3. Avoid naming conflict, we may create 2 classes with same name in different packages