

EXAMPLE

```
public abstract class Person {  
  
    private String name;  
  
    public Person(String nm) {  
        this.name = nm;  
    }  
  
    //abstract method  
    public abstract void work();  
  
    public void changeName(String newName) {  
        this.name = newName;  
    }  
}
```

Person.java

```
public class Employee extends Person{  
  
    private int empId;  
  
    public Employee(String nm, int id) {  
        super(nm);  
        this.empId = id;  
    }  
    @Override  
    public void work() {  
        if(empId == 0){  
            System.out.println("Not working");  
        }else{  
            System.out.println("Working as employee!!");  
        }  
    }  
}
```

Employee.java

ABSTRACT VS INTERFACE

Doesn't support multiple inheritance.

Abstract class can contains access modifiers for the subs, functions, and properties.

Abstract class can contains constructors.

Abstract class can contains attributes.

Variables have no restrictions.

Abstract class can contains both abstract and standard (non abstract) methods.

Methods have no restrictions, but abstract methods can't be static.

In subclass use the keyword "extends".

Support multiple inheritance.

Cannot have access modifiers by default, everything is assumed as public.

Doesn't contains constructors.

Doesn't contains attributes.

Variables in interface are "public static final" by default.

Can only contains abstract method and default method*.

Methods must be "public abstract", abstract methods also can't be static.

In subclass use the keyword "implements".

*When we extend an interface that contains a default method, we can perform:

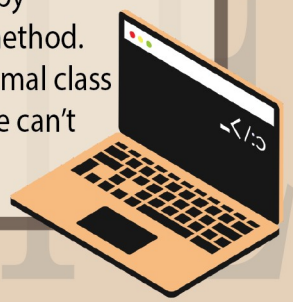
1. Not override the default method and will inherit the default method.
2. Override the default method similar to other methods we override in subclass.
3. Redecclare default method as abstract, which force subclass to override it.

```
public interface Shape {  
  
    //implicitly public,  
    static and final public  
    String LABEL = "Shape";  
  
    //interface methods are  
    implicitly abstract and  
    public  
    void draw();  
}
```

Shape.java

FUNCTION

Abstract class and interface both can have abstract method, the function of abstract method is to make us easier to change what a method does in each subclasses that inherits the superclass (abstract or interface class) by overriding the superclass' method. The difference between normal class is that abstract and interface can't be instantiated.



EXAMPLE

```
public class Circle implements Shape {  
  
    private double radius;  
  
    public Circle(double r){  
        this.radius = r;  
    }  
  
    @Override  
    public void draw() {  
        System.out.println("Drawing Circle");  
    }  
}
```

Circle.java

SOURCES

