

# Nesne Yönelimli Programlama

*Abstract class, Interface*

Hüseyin Ahmetoğlu

# Soyutlama

- ▶ Soyutlama, uygulama ayrıntılarını gizleme ve kullanıcıya yalnızca işlevselliği gösterme işlemidir.
- ▶ Başka bir şekilde, kullanıcıya yalnızca önemli şeyleri gösteri.
- ▶ SMS gönderdiğinizizi düşünelim. Siz sadece metni yazarsınız mesajın nasıl iletildiği ile ilgilenmezsiniz. Soyutlama tam olarak bunu sağlar.
- ▶ Soyutlama, nesnenin nasıl yaptığına değil, ne yaptığına odaklanmanızı sağlar.
- ▶ Java'da soyutlamaya ulaşmanın iki yolu vardır.
  - Soyut sınıf (%0 ila %100)
  - Arayüz (%100)

## Abstract class (Soyut Sınıf)

- ▶ Soyut olarak tanımlanan sınıfa soyut sınıf denir. Soyut ve soyut olmayan yöntemlere sahip olabilir.
- ▶ Soyut bir sınıf, **abstract** anahtar kelimesi ile bildirilmelidir.
- ▶ Soyut ve soyut olmayan metotlara sahip olabilir.
- ▶ Soyut sınıfın bir örneği oluşturulamaz.
- ▶ Yapıcılara ve statik metotlara sahip olabilir.
- ▶ Alt sınıfı, metot gövdesini değiştirmemeye zorlayacak son metotlara sahip olabilir.
- ▶ Soyut olarak bildirilen ve uygulaması olmayan bir yöntem, soyut bir yöntem olarak bilinir.

```
abstract class A{}
```

```
abstract void printStatus();//no method body and abstract
```

```
abstract class Bike{
    abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
    Bike obj = new Honda4();
    obj.run();
}
}
```

```
running safely
```

```
abstract class Shape{
abstract void draw();
}
//In real scenario, implementation is provided by others i.e. unknown by end user
class Rectangle extends Shape{
void draw(){System.out.println("drawing rectangle");}
}
class Circle1 extends Shape{
void draw(){System.out.println("drawing circle");}
}
//In real scenario, method is called by programmer or user
class TestAbstraction1{
public static void main(String args[]){
Shape s=new Circle1();//In a real scenario, object is provided through method, e.g., getShape() method
s.draw();
}
}
```

```
drawing circle
```

```
abstract class Bank{
    abstract int getRateOfInterest();
}

class SBI extends Bank{
    int getRateOfInterest(){return 7;}
}

class PNB extends Bank{
    int getRateOfInterest(){return 8;}
}

class TestBank{
    public static void main(String args[]){
        Bank b;
        b=new SBI();
        System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
        b=new PNB();
        System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
    }}
}
```

```
Rate of Interest is: 7 %
Rate of Interest is: 8 %
```

```
//Example of an abstract class that has abstract and non-abstract methods
abstract class Bike{
    Bike(){System.out.println("bike is created");}
    abstract void run();
    void changeGear(){System.out.println("gear changed");}
}
//Creating a Child class which inherits Abstract class
class Honda extends Bike{
    void run(){System.out.println("running safely..");}
}
//Creating a Test class which calls abstract and non-abstract methods
class TestAbstraction2{
    public static void main(String args[]){
        Bike obj = new Honda();
        obj.run();
        obj.changeGear();
    }
}
```

```
bike is created
running safely..
gear changed
```

- ▶ Kural: Bir sınıfta soyut bir metot varsa o sınıf soyut olmalıdır.
- ▶ Kural: Soyut bir metodu olan soyut bir sınıftan miras alıyorsanız, metodun uygulamasını sağlamanız veya bu sınıfı soyutlamanız gerekir.

```
class Bike12{  
    abstract void run();  
}
```

```
compile time error
```



# Interface (Arayüzler)

- ▶ Java'daki bir arayüz, bir sınıfın planıdır. Statik sabitlere ve soyut metotlara sahiptir.
- ▶ Java'da soyutlama ve çoklu kalıtım elde etmek için kullanılır.
- ▶ Arayüz içerisindeki soyut metotların gövdesi boştur.
- ▶ Java Arayüzü ayrıca IS-A ilişkisini temsil eder.
- ▶ Soyut sınıf gibi somutlaştırılmaz. Yani örneği oluşturulamaz.

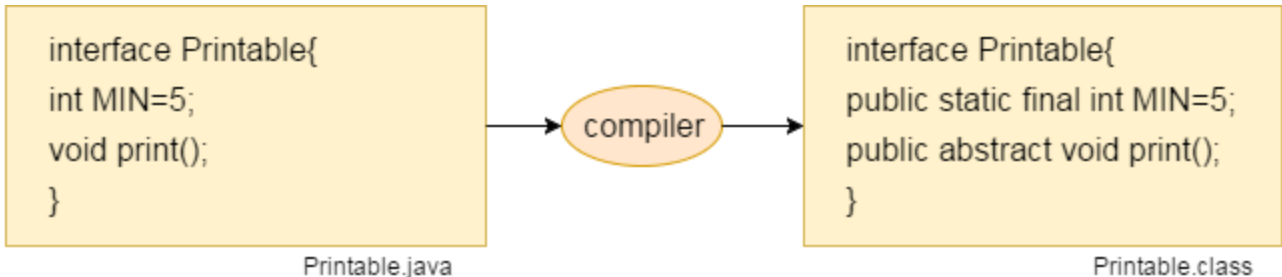
# Java arayüzünü neden kullanmalı?

- ▶ Soyutlama elde etmek için kullanılır.
- ▶ Arayüz ile çoklu kalıtımın işlevselliğini destekleyebiliriz.
- ▶ Gevşek bağlantı elde etmek için kullanılabilir.

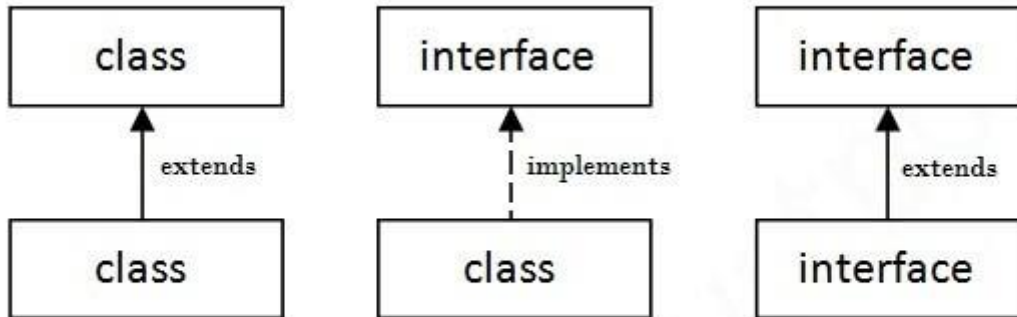
```
interface <interface_name>{  
  
    // declare constant fields  
  
    // declare methods that abstract  
  
    // by default.  
}
```

# Derleyici ve Interface

- ▶ Java derleyicisi, interface metotlarından önce public ve abstract anahtar sözcüklerini ekler. Ayrıca veri üyelerinden önce public, static ve final anahtar kelimelerini ekler.



# Interface ve Class



```
interface printable{  
    void print();  
}  
class A6 implements printable{  
    public void print(){System.out.println("Hello");}  
  
    public static void main(String args[]){  
        A6 obj = new A6();  
        obj.print();  
    }  
}
```

Hello

```
//Interface declaration: by first user
interface Drawable{
void draw();
}

//Implementation: by second user
class Rectangle implements Drawable{
public void draw(){System.out.println("drawing rectangle");}
}
class Circle implements Drawable{
public void draw(){System.out.println("drawing circle");}
}

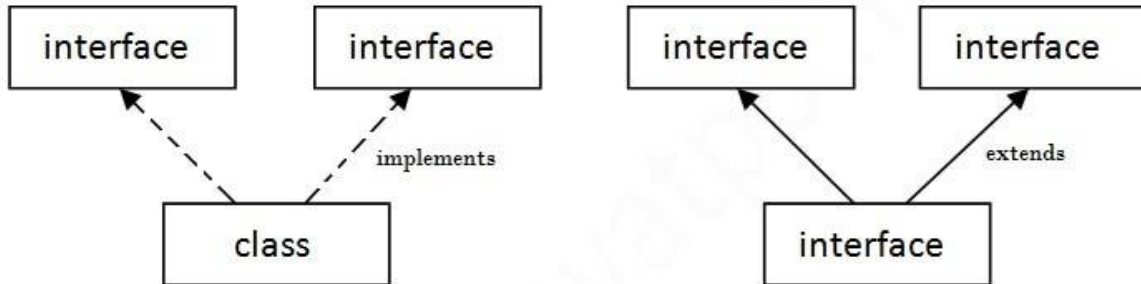
//Using interface: by third user
class TestInterface1{
public static void main(String args[]){
Drawable d=new Circle();//In real scenario, object is provided by method e.g. getDrawable()
d.draw();
}}
```

```
drawing circle
```

```
interface Bank{
    float rateOfInterest();
}
class SBI implements Bank{
    public float rateOfInterest(){return 9.15f;}
}
class PNB implements Bank{
    public float rateOfInterest(){return 9.7f;}
}
class TestInterface2{
    public static void main(String[] args){
        Bank b=new SBI();
        System.out.println("ROI: "+b.rateOfInterest());
    }
}
```

ROI: 9.15

# Arayüzle Java'da çoklu kalıtım



## Multiple Inheritance in Java



```
interface Printable{
void print();
}
interface Showable{
void show();
}
class A7 implements Printable,Showable{
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}

public static void main(String args[]){
A7 obj = new A7();
obj.print();
obj.show();
}
}
```

```
Output:Hello
       Welcome
```

```
interface Printable{  
void print();  
}  
interface Showable{  
void print();  
}  
  
class TestInterface3 implements Printable, Showable{  
public void print(){System.out.println("Hello");}  
public static void main(String args[]){  
TestInterface3 obj = new TestInterface3();  
obj.print();  
}  
}
```



Hello

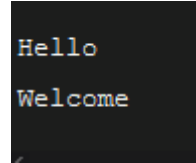
# Arayüzlerde Kalıtım

```
interface Printable{
    void print();
}

interface Showable extends Printable{
    void show();
}

class TestInterface4 implements Showable{
    public void print(){System.out.println("Hello");}
    public void show(){System.out.println("Welcome");}

    public static void main(String args[]){
        TestInterface4 obj = new TestInterface4();
        obj.print();
        obj.show();
    }
}
```



```
Hello
Welcome
```

# Default Method

```
interface Drawable{
    void draw();
    default void msg(){System.out.println("default method");}
}
class Rectangle implements Drawable{
    public void draw(){System.out.println("drawing rectangle");}
}
class TestInterfaceDefault{
    public static void main(String args[]){
        Drawable d=new Rectangle();
        d.draw();
        d.msg();
    }
}
```

```
drawing rectangle
default method
```

## Static Method

```
interface Drawable{
void draw();
static int cube(int x){return x*x*x;}
}
class Rectangle implements Drawable{
public void draw(){System.out.println("drawing rectangle");}
}

class TestInterfaceStatic{
public static void main(String args[]){
Drawable d=new Rectangle();
d.draw();
System.out.println(Drawable.cube(3));
}}
```

```
drawing rectangle
27
```

# abstract class ve interface

*Soyut sınıf ve arabirimin ikisi de somutlaştırılmaz.*

- ▶ Soyut sınıf, soyut ve soyut olmayan metotlara sahip olabilir.
- ▶ Soyut sınıf, çoklu kalıtımı desteklemez.
- ▶ Soyut sınıf, final, final olmayan, static ve static olmayan değişkenlere sahip olabilir.
- ▶ **Soyut bir sınıf** , başka bir Java sınıfından miras alabilir ve birden çok Java arabirimini implemente edebilir.
- ▶ Bir Java soyut sınıfı, private, protected vb. gibi sınıf üyelerine sahip olabilir.
- ▶ Arayüz sadece soyut metotlara sahip olabilir. Java 8'den bu yana, varsayılan ve statik metotlara da sahip olabilir.
- ▶ Arayüz çoklu kalıtımı destekler.
- ▶ Arayüz sadece static ve final değişkenlere sahiptir.
- ▶ Bir **arabirim** yalnızca başka bir Java arabiriminden miras alabilir.
- ▶ Java arabiriminin üyeleri varsayılan olarak public tir.

```
interface A{
void a();
void b();
void c();
void d();
}
```

```
abstract class B implements A{
public void c(){System.out.println("I am c");}
}
```

```
class M extends B{
public void a(){System.out.println("I am a");}
public void b(){System.out.println("I am b");}
public void d(){System.out.println("I am d");}
}
```

```
class Test5{
public static void main(String args[]){
A a=new M();
a.a();
a.b();
a.c();
a.d();
}}
```

```
Output:I am a
       I am b
       I am c
       I am d
```

## KAYNAKLAR

- ▶ Java Tutorial | Learn Java - javatpoint. (2021, March 21). Retrieved from <https://www.javatpoint.com/java-tutorial>