



# 1학기 고급자바 실습

## week 1-1

김민진(18)

김지희(18)

문의 메일 : [genie02166@duksung.ac.kr](mailto:genie02166@duksung.ac.kr)

# Part1. 인터페이스 기본 개념

1) 인터페이스 선언

2-1) 인터페이스 구현 – Television 구현 클래스

2-2) 인터페이스 구현 – Audio 구현 클래스

3) 다중 인터페이스 구현을 위한 다른 인터페이스 선언

4) 다중 인터페이스 구현 클래스 – SmartTelevision

5) 인터페이스 사용

\* RemoteControlExample 출력 결과

6) 필드 다형성을 위한 Tire 인터페이스와

HankookTire, KumhoTire 구현 클래스

7) Car 클래스

- 인터페이스 타입 필드 선언과 초기 구현 객체 대입

8) 필드 다형성 테스트

\*\* CarExample 출력 결과

9) 인터페이스 배열로 구현 객체 관리

10) 필드 다형성 테스트

\*\*\* CarExample2 출력 결과

# 1) 인터페이스 선언

```
RemoteControl.java Television.java Audio.java Searchable.java SmartTelevision.java RemoteControlExample.java
1 package week1;
2
3 public interface RemoteControl {
4
5     public int MAX_VOLUME = 10;
6     public int MIN_VOLUME = 10;
7
8     //추상 메소드 - 메소드 선언부만 작성
9     public void turnOn();
10    public void turnOff();
11    public void setVolume(int volume);
12
13    //디폴트 메소드
14    default void setMute(boolean mute) {
15        if(mute) {
16            System.out.println("무음 처리합니다.");
17        }else {
18            System.out.println("무음 해제합니다.");
19        }
20    }
21
22    //정적 메소드
23    static void changeBattery() {
24        System.out.println("건전지를 교환합니다.");
25    }
26 }
27
```

## 2-1) 인터페이스 구현 – Television 구현 클래스

```
RemoteControl.java Television.java Audio.java Searchable.java SmartTelevision.java RemoteControlExample.java
1 package week1;
2
3 public class Television implements RemoteControl{
4
5     //필드
6     private int volume;
7
8     //turnOn() 추상 메소드의 실제 메소드
9     public void turnOn() {
10         System.out.println("TV를 켭니다.");
11     }
12     //turnOff() 추상 메소드의 실제 메소드
13     public void turnOff() {
14         System.out.println("TV를 끕니다.");
15     }
16     //setVolume() 추상 메소드의 실제 메소드
17     public void setVolume(int volume) {
18         if(volume>RemoteControl.MAX_VOLUME) {
19             this.volume = RemoteControl.MAX_VOLUME;
20         }else if(volume<RemoteControl.MIN_VOLUME) {
21             this.volume = RemoteControl.MIN_VOLUME;
22         }else {
23             this.volume = volume;
24         }
25         System.out.println("현재 TV 볼륨: " + this.volume);
26     }
27 }
28
```

## 2-2) 인터페이스 구현 – Audio 구현 클래스

RemoteControl.java Television.java Audio.java Searchable.java SmartTelevision.java RemoteControlExample.java

```
1 package week1;
2
3 public class Audio implements RemoteControl{
4
5     //필드
6     private int volume;
7     private boolean mute;
8
9     //turnOn() 추상 메소드의 실제
10    public void turnOn() {
11        System.out.println("Audio를 켭니다.");
12    }
13    //turnOff() 추상 메소드의 실제
14    public void turnOff() {
15        System.out.println("Audio를 끕니다.");
16    }
17    //setVolume() 추상 메소드의 실제
18    public void setVolume(int volume) {
19        if(volume>RemoteControl.MAX_VOLUME) {
20            this.volume = RemoteControl.MAX_VOLUME;
21        }else if(volume<RemoteControl.MIN_VOLUME) {
22            this.volume = RemoteControl.MIN_VOLUME;
23        }else {
24            this.volume = volume;
25        }
26        System.out.println("현재 Audio 볼륨: " + this.volume);
27    }
28
29    //디폴트 메소드 재정의
30    @Override
31    public void setMute(boolean mute) {
32        this.mute = mute;
33        if(mute) {
34            System.out.println("Audio 무음 처리합니다.");
35        }else {
36            System.out.println("Audio 무음 해제합니다.");
37        }
38    }
39 }
```

### 3) 다중 인터페이스 구현을 위한 다른 인터페이스 선언

```
RemoteControl.java Television.java Audio.java *Searchable.java SmartTelevision.java RemoteControlExample.java
1 package week1;
2
3 public interface Searchable {
4
5     void search(String url);
6 }
7
```

## 4) 다중 인터페이스 구현 클래스 - SmartTelevision

```
RemoteControl.java  Television.java  Audio.java  *Searchable.java  SmartTelevision.java  RemoteControlExample.java

1 package week1;
2
3 public class SmartTelevision implements RemoteControl, Searchable{
4     private int volume;
5
6     //RemoteControl의 추상 메소드에 대한 실제 메소드
7     public void turnOn() {
8         System.out.println("TV를 켭니다.");
9     }
10    public void turnOff() {
11        System.out.println("TV를 끕니다.");
12    }
13    public void setVolume(int volume) {
14        if(volume>RemoteControl.MAX_VOLUME) {
15            this.volume = RemoteControl.MAX_VOLUME;
16        }else if(volume<RemoteControl.MIN_VOLUME) {
17            this.volume = RemoteControl.MIN_VOLUME;
18        }else {
19            this.volume = volume;
20        }
21        System.out.println("현재 TV 볼륨: " + this.volume);
22    }
23
24    //Searchable의 추상 메소드에 대한 실제 메소드
25    public void search(String url) {
26        System.out.println(url + "을 검색합니다.");
27    }
28
29 }
```

## 5) 인터페이스 사용

```
RemoteControl.java  Television.java  Audio.java  *Searchable.java  SmartTelevision.java  RemoteControlExample.java ✕
1 package week1;
2
3 public class RemoteControlExample {
4
5     public static void main(String[] args) {
6         RemoteControl rc;
7
8         rc = new Television();
9         rc.turnOn();
10        rc.setMute(true);
11        rc.turnOff();
12
13        rc = new Audio();
14        rc.turnOn();
15        rc.setMute(true);
16        rc.turnOff();
17
18        RemoteControl.changeBattery();
19
20    }
21
22 }
23
```



## \* RemoteControlExample 출력 결과

Problems @ Javadoc Declaration Console

<terminated> RemoteControlExample [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe

TV를 켭니다.

무음 처리합니다.

TV를 끕니다.

Audio를 켭니다.

Audio 무음 처리합니다.

Audio를 끕니다.

건전지를 교환합니다.

## 6) 필드 다형성을 위한 Tire 인터페이스와 HankookTire, KumhoTire 구현 클래스

```
*Tire.java HankookTire.java *KumhoTire.java *Car.java CarExample.java
1 package week1;
2
3 public interface Tire {
4
5     public void roll();
6 }
7
```

```
*Tire.java HankookTire.java *KumhoTire.java *Car.java CarExample.java
1 package week1;
2
3 public class HankookTire implements Tire{
4
5     @Override
6     public void roll() {
7         System.out.println("한국 타이어가 굴러갑니다.");
8     }
9 }
10
```

```
*Tire.java HankookTire.java *KumhoTire.java *Car.java CarExample.java
1 package week1;
2
3 public class KumhoTire implements Tire{
4
5     //Tire 인터페이스 구현
6     @Override
7     public void roll() {
8         System.out.println("금호 타이어가 굴러갑니다.");
9     }
10 }
11
```

## 7) Car 클래스

### - 인터페이스 타입 필드 선언과 초기 구현 객체 대입

```
*Tire.java  HankookTire.java  *KumhoTire.java  *Car.java  CarExample.java
1 package week1;
2
3 public class Car {
4
5     Tire FrontLeftTire = new HankookTire();
6     Tire FrontRightTire = new HankookTire();
7     Tire BackLeftTire = new HankookTire();
8     Tire BackRightTire = new HankookTire();
9
10    void run() {
11        FrontLeftTire.roll();
12        FrontRightTire.roll();
13        BackLeftTire.roll();
14        BackRightTire.roll();
15    }
16 }
17
```

## 8) 필드 다형성 테스트

```
*Tire.java  HankookTire.java  *KumhoTire.java  *Car.java  *CarExample.java ✕  
1 package week1;  
2  
3 public class CarExample {  
4  
5     public static void main(String[] args) {  
6         Car myCar = new Car();  
7  
8         myCar.run();  
9  
10        //타이어 교체  
11        myCar.FrontLeftTire = new KumhoTire();  
12        myCar.FrontRightTire = new KumhoTire();  
13  
14        myCar.run();  
15    }  
16 }  
17
```

## \*\* CarExample 출력 결과

Problems @ Javadoc Declaration Console ✕

<terminated> CarExample [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe

한국 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.  
금호 타이어가 굴러갑니다.  
금호 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.

## 9) 인터페이스 배열로 구현 객체 관리

```
*Tire.java  HankookTire.java  *KumhoTire.java  Car2.java  CarExample2.java
1 package week1;
2
3 public class Car2 {
4
5     Tire[] tires = {
6         new HankookTire(),
7         new HankookTire(),
8         new HankookTire(),
9         new HankookTire()
10    };
11
12    void run() {
13        //tires 배열의 각 인데스에 담긴 객체들을 한 번씩 실행
14        for(Tire tire : tires) {
15            tire.roll();
16        }
17    }
18 }
19
```

## 10) 필드 다형성 테스트

```
*Tire.java    HankookTire.java    *KumhoTire.java    Car2.java    *CarExample2.java ✖
1 package week1;
2
3 public class CarExample2 {
4
5     public static void main(String[] args) {
6         Car2 myCar = new Car2();
7
8         myCar.run();
9
10        //타이어 교체
11        myCar.tires[0] = new KumhoTire();
12        myCar.tires[1] = new KumhoTire();
13
14        myCar.run();
15
16    }
17
18 }
19
```

## \*\*\* CarExample2 출력 결과

 Problems  Javadoc  Declaration  Console 

<terminated> CarExample2 [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe

한국 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.  
금호 타이어가 굴러갑니다.  
금호 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.  
한국 타이어가 굴러갑니다.



## Part2. 중첩 클래스와 중첩 인터페이스

### 1) 중첩 클래스

- 개념 정리
- 인스턴스 멤버 클래스
- 정적 멤버 클래스
- 로컬 클래스
- 중첩 클래스 객체 생성

\* 중첩 클래스 출력 결과

### 2) 중첩 인터페이스

- 구현 클래스
- 버튼 이벤트 처리

\*\* ButtonExample 출력 결과

# 1) 중첩 클래스

- 중첩 클래스는 클래스 내부에 선언되는 위치에 따라서 두 가지로 분류
  1. 멤버 클래스 : 클래스의 멤버로서 선언되는 중첩 클래스
  2. 로컬 클래스 : 메소드 내부에서 선언되는 중첩 클래스

선언 위치에 따른 분류		선언 위치	설명
멤버 클래스	인스턴스 멤버 클래스	<pre>class A {     class B { ... } }</pre>	A 객체를 생성해야만 사용할 수 있는 B 중첩 클래스
	정적 멤버 클래스	<pre>class A {     <b>static</b> class B { ... } }</pre>	A 클래스로 바로 접근할 수 있는 B 중첩 클래스
로컬 클래스		<pre>class A {     void method() {         class B { ... }     } }</pre>	method()가 실행할 때만 사용할 수 있는 B 중첩 클래스

# 1) 중첩 클래스

- 인스턴스 멤버 클래스
- 정적 멤버 클래스
- 로컬 클래스

```
A.java ✕
1 package week1;
2
3 /**바깥 클래스**/
4 class A {
5     A() { System.out.println("A 객체가 생성됨"); }
6
7     /**인스턴스 멤버 클래스**/
8     class B {
9         B() { System.out.println("B 객체가 생성됨"); }
10        int field1;
11        //static int field2;
12        void method1() { }
13        //static void method2() { }
14    }
15
16    /**정적 멤버 클래스**/
17    static class C {
18        C() { System.out.println("C 객체가 생성됨"); }
19        int field1;
20        static int field2;
21        void method1() { }
22        static void method2() { }
23    }
24
```

```
25 void method() {
26     /**로컬 클래스**/
27     class D {
28         D() { System.out.println("D 객체가 생성됨"); }
29         int field1;
30         //static int field2;
31         void method1() { }
32         //static void method2() { }
33     }
34     D d = new D();
35     d.field1 = 3;
36     d.method1();
37 }
38 }
```

# 1) 중첩 클래스 - 객체 생성

실행 결과

Main.java

```
1 package week1;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         A a = new A();
7
8         //인스턴스 멤버 클래스 객체 생성
9         A.B b = a.new B();
10        b.field1 = 3;
11        b.method1();
12
13        //정적 멤버 클래스 객체 생성
14        A.C c = new A.C();
15        c.field1 = 3;
16        c.method1();
17        A.C.field2 = 3;
18        A.C.method2();
19
20        //로컬 클래스 객체 생성을 위한 메소드 호출
21        a.method();
22
23    }
24
25 }
```

Console

P

<terminated> Ma

A 객체가 생성됨

B 객체가 생성됨

C 객체가 생성됨

D 객체가 생성됨

## 2) 중첩 인터페이스

```
Button.java ✕
1 package week1;
2
3 public class Button {
4     OnClickListener listener;
5
6     void setOnClickListener(OnClickListener listener) {
7         this.listener = listener;
8     }
9
10    void touch() {
11        listener.onClick();
12    }
13
14    interface OnClickListener {
15        void onClick();
16    }
17 }
```

## 2) 중첩 인터페이스 - 구현 클래스

```
CallListener.java ✕
1 package week1;
2
3 public class CallListener implements Button.OnClickListener {
4     @Override
5     public void onClick() {
6         System.out.println("전화를 겁니다.");
7     }
8 }
```

```
MessageListener.java ✕
1 package week1;
2
3 public class MessageListener implements Button.OnClickListener {
4     @Override
5     public void onClick() {
6         System.out.println("메시지를 보냅니다.");
7     }
8 }
```

## 2) 중첩 인터페이스 - 버튼 이벤트 처리

ButtonExample.java

```
1 package week1;
2
3 public class ButtonExample {
4
5     public static void main(String[] args) {
6         Button btn = new Button();
7
8         btn.setClickListener(new CallListener());
9         btn.touch();
10
11        btn.setClickListener(new MessageListener());
12        btn.touch();
13    }
14
15 }
```

실행 결과

Console

<terminated> Button

전화를 겁니다.

메시지를 보냅니다.

# 출석 과제 (3/15 월 오후 11:59 마감)

Q. 다음과 같이 Car 클래스 내부에 Tire와 Engine이 멤버 클래스로 선언되어 있습니다.

NestedClassExample에서 [1], [2] 위치에 멤버 클래스의 객체를 생성하는 코드를 작성해서 제출해주세요.

\*Car.java

```
1
2 public class Car {
3     class Tire { }
4     static class Engine { }
5 }
```

\*NestedClassExample.java

```
1
2 public class NestedClassExample {
3     public static void main(String[] args) {
4
5         Car myCar = new Car();
6
7         Car.Tire tire = [ 1 ]
8
9         Car.Engine engine = [ 2 ]
10
11     }
12 }
```