

## 09-2. 익명 객체

혼자 공부하는 자바 (신용권 저)

- 시작하기 전에
- 익명 자식 객체 생성
- 익명 구현 객체 생성
- 익명 객체의 로컬 변수 사용
- 키워드로 끝내는 핵심 포인트
- 확인문제

## 시작하기 전에

[핵심 키워드] : 익명 자식 객체, 익명 구현 객체

[핵심 포인트]

클래스 선언 시 일반적으로 클래스 이름과 동일한 소스 파일 생성하고 클래스를 선언한다. 그런데 클래스 이름이 없는 객체가 있고, 이를 익명 객체라고 한다. 익명 객체에 대해 알아본다.

# 시작하기 전에

## ❖ 익명 (anonymous) 객체

- 이름이 없는 객체
- 어떤 클래스를 상속하거나 인터페이스를 구현하여야 함

[상속]

```
class 클래스이름1 extends 부모클래스 { ... }  
부모클래스 변수 = new 클래스이름1();
```

[구현]

```
class 클래스이름2 implements 인터페이스 { ... }  
인터페이스 변수 = new 클래스이름2();
```

[상속]

```
부모클래스 변수 = new 부모클래스() { ... };
```

[구현]

```
인터페이스 변수 = new 인터페이스() { ... };
```

# 익명 자식 객체 생성

## ❖ 익명 자식 객체 생성

- 일반적인 경우 부모 타입의 필드나 변수 선언하고 자식 객체를 초기값으로 대입하는 경우
  - 부모 클래스 상속하여 자식 클래스 선언
  - new 연산자 이용하여 자식 객체 생성 후 부모 타입의 필드나 변수에 대입

```
class Child extends Parent { } ← 자식 클래스 선언

class A {
    Parent field = new Child(); ← 필드에 자식 객체를 대입
    void method() {
        Parent localVar = new Child(); ← 로컬 변수에 자식 객체를 대입
    }
}
```

# 익명 자식 객체 생성

- 자식 클래스를 재사용하지 않고 특정 위치에서만 사용하려는 경우
  - 익명 자식 객체 생성하여 사용

```
부모클래스 [필드|변수] = new 부모클래스(매개값, ...) {  
    //필드  
    //메소드  
};
```

- 필드 선언할 때 초기값으로 익명 자식 객체 생성하여 대입

```
class A {  
    Parent field = new Parent() {  
        int childField;  
        void childMethod() { }  
        @Override  
        void parentMethod() { }  
    };  
}
```

A 클래스의 필드 선언

Parent의 메소드를 재정의

# 익명 자식 객체 생성

- 메소드 내에서 로컬 변수 선언 시 초기값으로 익명 자식 객체 생성하여 대입

```
class A {  
    void method() {  
        Parent localVar = new Parent() {  
            int childField;  
            void childMethod() { }  
            @Override  
            void parentMethod() { }  
        };  
    }  
}
```

로컬 변수 선언

Parent의 메소드를 재정의

# 익명 자식 객체 생성

- 메소드 매개 변수가 부모 타입일 경우 메소드 호출하는 코드에서 익명 자식 객체 생성하여 매개값으로 대입

```
class A {  
    void method1(Parent parent) { }
```

```
    void method2() {
```

```
        method1(  
            new Parent() {  
                int childField;  
                void childMethod() { }  
                @Override  
                void parentMethod() { }  
            }  
        );  
    }  
}
```

method1() 메소드 호출

method1()의 매개값으로  
익명 자식 객체를 대입



# 익명 자식 객체 생성

- 익명 자식 객체에 새롭게 정의된 필드 및 메소드는 익명 자식 객체 내부에서만 사용되고 외부에서는 접근할 수 없음

```
class A {  
    Parent field = new Parent() {  
        int childField; ←  
        void childMethod() { } ←  
        @Override  
        void parentMethod() { ←  
            childField = 3;  
            childMethod();  
        }  
    };  
  
    void method() {  
        field.childField = 3; ←  
        field.childMethod(); ←  
        field.parentMethod(); ←  
    }  
}
```

# 익명 자식 객체 생성

## ❖ 예시 - 부모 클래스

```
01 package sec02.exam01;  
02  
03 public class Person {  
04     void wake() {  
05         System.out.println("7시에 일어납니다.");  
06     }  
07 }
```

# 익명 자식 객체 생성

## ❖ 예시 - 자식 객체 생성

```
01 package sec02.exam01;
02
03 public class Anonymous {
04     //필드 초기값으로 대입
05     Person field = new Person() {
06         void work() {
07             System.out.println("출근합니다.");
08         }
09         @Override
10         void wake() {
11             System.out.println("6시에 일어납니다.");
12             work();
13         }
14     };
15
16     void method1() {
```

← 필드값으로 익명 객체 대입

# 익명 자식 객체 생성

```
17      //로컬 변수값으로 대입
18      Person localVar = new Person() {
19          void walk() {
20              System.out.println("산책합니다.");
21          }
22          @Override
23          void wake() {
24              System.out.println("7시에 일어납니다.");
25              walk();
26          }
27      };
28      //로컬 변수 사용
29      localVar.wake();
30  }
31
32  void method2(Person person) {
33      person.wake();
34  }
35  }
```

← 로컬 변수값으로 익명 객체 대입

# 익명 자식 객체 생성

```
01 package sec02.exam01;
02
03 public class AnonymousExample {
04     public static void main(String[] args) {
05         Anonymous anony = new Anonymous();
06         //익명 객체 필드 사용
07         anony.field.wake();
08         //익명 객체 로컬 변수 사용
09         anony.method1();
10         //익명 객체 매개값 사용
11         anony.method2(
12             new Person() {
13                 void study() {
14                     System.out.println("공부합니다.");
15                 }
16                 @Override
17                 void wake() {
18                     System.out.println("8시에 일어납니다.");
19                     study();
20                 }
21             }
22         );
23     }
24 }
```

← 매개값으로 익명 객체 대입

**실행결과**

6시에 일어납니다.  
출근합니다.  
7시에 일어납니다.  
산책합니다.  
8시에 일어납니다.  
공부합니다.

# 익명 구현 객체 생성

- ❖ 인터페이스 타입의 필드 혹은 변수 선언 후 구현 객체를 초기값으로 대입하는 경우

```
class TV implements RemoteControl { }
```

```
class A {  
    RemoteControl field = new TV(); ← 필드에 구현 객체를 대입  
    void method() {  
        RemoteControl localVar = new TV(); ← 로컬 변수에 구현 객체를 대입  
    }  
}
```

- 구현 클래스가 재사용되지 않고 특정 위치에서만 사용되는 경우 – 익명 구현 객체 생성

```
인터페이스 [필드|변수] = new 인터페이스() {  
    //인터페이스에 선언된 추상 메소드의 실제 메소드 선언  
    //필드  
    //메소드  
} ;
```

# 익명 구현 객체 생성

- 필드 선언 시 초기값으로 익명 구현 객체 생성하여 대입하는 경우

```
class A {  
    RemoteControl field = new RemoteControl() {  
        @Override  
        void turnOn() { }  
    };  
}
```

클래스 A의 필드 선언

RemoteControl 인터페이스의 추상 메소드에 대한 실제 메소드

- 메소드 내에서 로컬 변수 선언 시 초기값으로 익명 구현 객체 생성하여 대입

```
void method() {  
    RemoteControl localVar = new RemoteControl() {  
        @Override  
        void turnOn() { }  
    };  
}
```

로컬 변수 선언

RemoteControl 인터페이스의 추상 메소드에 대한 실제 메소드

# 익명 구현 객체 생성

- 메소드의 매개 변수가 인터페이스 타입일 때 메소드 호출하는 코드에서 익명 구현 객체 생성하여 매개값으로 대입하는 경우

```
class A {  
    void method1(RemoteControl rc) { }  
  
    void method2() {  
        method1(  
            new RemoteControl() {  
                @Override  
                void turnOn() { }  
            }  
        );  
    }  
}
```

method1() 메소드 호출

method1()의 매개값으로  
익명 구현 객체를 대입



# 익명 구현 객체 생성

## ❖ 예시 – 인터페이스

```
01 package sec02.exam02;
02
03 public interface RemoteControl {
04     public void turnOn();
05     public void turnOff();
06 }
```

## ❖ 예시 – 익명 구현 객체 생성

```
01 package sec02.exam02;
02
03 public class Anonymous {
04     //필드 초기값으로 대입
05     RemoteControl field = new RemoteControl() {
06         @Override
07         public void turnOn() {
08             System.out.println("TV를 켭니다.");
09         }
10         @Override
11         public void turnOff() {
12             System.out.println("TV를 끕니다.");
```

← 필드 선언과  
초기값 대입

# 익명 구현 객체 생성

```
13     }  
14 };  
15  
16 void method1() {  
17     //로컬 변수값으로 대입  
18     RemoteControl localVar = new RemoteControl() {  
19         @Override  
20         public void turnOn() {  
21             System.out.println("Audio를 켭니다.");  
22         }  
23         @Override  
24         public void turnOff() {  
25             System.out.println("Audio를 끕니다.");  
26         }  
27     };  
28     //로컬 변수 사용  
29     localVar.turnOn();  
30 }  
31  
32 void method2(RemoteControl rc) {  
33     rc.turnOn();  
34 }  
35 }
```

← 로컬 변수 선언과  
초기값 대입

# 익명 구현 객체 생성

```
01 package sec02.exam02;
02
03 public class AnonymousExample {
04     public static void main(String[] args) {
05         Anonymous anony = new Anonymous();
06         //익명 객체 필드 사용
07         anony.field.turnOn();
08         //익명 객체 로컬 변수 사용
09         anony.method1();
10         //익명 객체 매개값 사용
11         anony.method2(
12             new RemoteControl() {
13                 @Override
14                 public void turnOn() {
15                     System.out.println("SmartTV를 켭니다.");
16                 }
17                 @Override
18                 public void turnOff() {
19                     System.out.println("SmartTV를 끕니다.");
20                 }
21             }
22         );
23     }
24 }
```

← 매개값

**실행결과**

TV를 켭니다.  
Audio를 켭니다.  
SmartTV를 켭니다.

# 익명 구현 객체 생성

## ❖ 예시 - UI 클래스

```
01 package sec02.exam03;
02
03 public class Button {
04     OnClickListener listener; ← 인터페이스 타입 필드
05
06     void setOnClickListener(OnClickListener listener) {
07         this.listener = listener; ← 매개 변수의 다형성
08     }
09
10     void touch() {
11         listener.onClick(); ← 구현 객체의 onClick()
12         메소드 호출
13     }
14
15     static interface OnClickListener {
16         void onClick(); ← 중첩 인터페이스
17     }
17 }
```

# 익명 구현 객체 생성

- Window 클래스를 2개의 Button 객체 가진 창이라 가정
- 첫 번째 button1 클릭 이벤트 처리는 필드로 선언한 익명 구현 객체가 담당
- 두 번째 button2 클릭 이벤트 처리는 setOnClickListener() 호출할 때 매개값으로 준 익명 구현 객체가 담당

```
01 package sec02.exam03;
02
03 public class Window {
04     Button button1 = new Button();
05     Button button2 = new Button();
06
07     //필드 초기값으로 대입
08     Button.OnClickListener listener = new Button.OnClickListener() {
09         @Override
10         public void onClick() {
11             System.out.println("전화를 겁니다.");
12         }
13     };
14
15     Window() {
16         button1.setOnClickListener( listener ); ← 매개값으로 필드 대입
17         button2.setOnClickListener(new Button.OnClickListener() {
18             @Override
19             public void onClick() {
20                 System.out.println("메시지를 보냅니다.");
21             }
22         });
23     }
24 }
```

← 필드값으로 익명 객체 대입

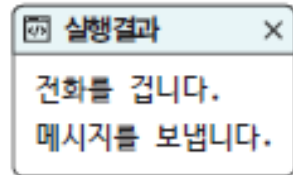
← 매개값으로 익명 객체 대입

# 익명 구현 객체 생성

## ❖ 예시 - 실행 클래스

```
01 package sec02.exam03;  
02  
03 public class Main {  
04     public static void main(String[] args) {  
05         Window w = new Window();  
06         w.button1.touch();  
07         w.button2.touch();  
08     }  
09 }
```

← 버튼 클릭



# 익명 객체의 로컬 변수 사용

- ❖ 메소드의 매개 변수나 로컬 변수를 익명 객체 내부에서 사용할 때의 제한
  - 메소드가 종료되어도 익명 객체가 계속 실행 상태로 존재할 수 있음
  - 메소드의 매개 및 로컬 변수를 익명 객체 내부에서 사용할 경우에는 지속 사용 불가
  - 컴파일 시 익명 객체에서 사용하는 매개 변수나 로컬 변수의 값을 익명 객체 내부에 복사해두고 사용
    - 매개 및 로컬 변수가 수정되어 값 변경되면 매개 및 로컬 변수를 final로 선언할 것을 요구

## ❖ 예시 - 인터페이스

---

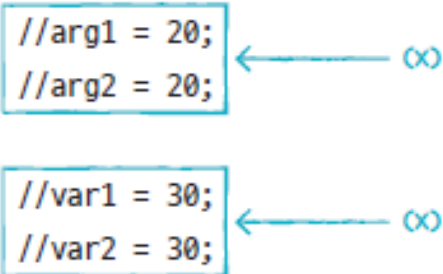
```
01 package sec02.exam04;
02
03 public interface Calculatable {
04     public int sum();
05 }
```

---

# 익명 객체의 로컬 변수 사용

## ❖ 예시 – 익명 객체의 로컬 변수 사용

```
01 package sec02.exam04;
02
03 public class Anonymous {
04     private int field;
05
06     public void method(final int arg1, int arg2) {
07         final int var1 = 0;
08         int var2 = 0;
09
10         field = 10;
11
12         //arg1 = 20;
13         //arg2 = 20;
14
15         //var1 = 30;
16         //var2 = 30;
17
```



The diagram illustrates that the commented-out lines for `arg1`, `arg2`, `var1`, and `var2` are not executed, as indicated by the arrows pointing to the infinity symbol ( $\infty$ ).

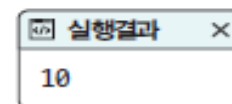


## 익명 객체의 로컬 변수 사용

```
18     Calculatable calc = new Calculatable() {
19         @Override
20         public int sum() {
21             int result = field + arg1 + arg2 + var1 + var2;
22             return result;
23         }
24     };
25
26     System.out.println(calc.sum());
27 }
28 }
```

### ❖ 예시 – 익명 객체의 로컬 변수 사용

```
01     package sec02.exam04;
02
03     public class AnonymousExample {
04         public static void main(String[] args) {
05             Anonymous anony = new Anonymous();
06             anony.method(0, 0);
07         }
08     }
```



실행결과  
10

## 키워드로 끝내는 핵심 포인트

- **익명 자식 객체**: 자식 클래스가 재사용되지 않고 오로지 특정 위치에서 사용되는 경우라면 익명 자식 객체 생성하여 사용하는 것이 편리함

```
부모클래스 [필드|변수] = new 부모클래스(매개값, ...) {  
    //필드  
    //메소드  
};
```

- **익명 구현 객체**: 구현 객체 클래스가 재사용되지 않고 오로지 특정 위치에서 사용되는 경우라면 익명 구현 객체 생성하여 사용하는 것이 편리함

```
인터페이스 [필드|변수] = new 인터페이스() {  
    //인터페이스에 선언된 추상 메소드의 실제 메소드 선언  
    //필드  
    //메소드  
};
```

## 확인문제

- ❖ AnonymousExample 클래스의 실행결과를 보고 Worker 클래스의 익명 자식 객체를 이용해서 필드, 로컬 변수의 초기값과 메소드의 매개값을 대입해보세요

인터페이스

소스 코드 Vehicle.java

```
01 package sec02.verify.exam01;
02
03 public class Worker {
04     public void start() {
05         System.out.println("쉬고 있습니다.");
06     }
07 }
```

익명 구현 클래스와 객체 생성

[소스 코드](#) Anonymous.java


```
01 package sec02.verify.exam01;
02
03 public class Anonymous {
04     Worker field = 
05
06
07     void method1() {
08         Worker localVar = 
09
10
11         localVar.start();
12     }
13
14     void method2(Worker worker) {
15         worker.start();
16     }
17 }
```

# 확인문제

익명 구현 클래스와 객체 생성

[소스 코드](#) AnonymousExample.java

```
01 package sec02.verify.exam01;
02
03 public class AnonymousExample {
04     public static void main(String[] args) {
05         Anonymous anony = new Anonymous();
06         anony.field.start();
07         anony.method1();
08         anony.method2(
09             
10
11     );
12 }
13 }
```

 실행결과 ×

디자인을 합니다.  
개발을 합니다.  
테스트를 합니다.

## 확인문제

- ❖ CheckBox 클래스 내용을 보면 중첩 인터페이스 타입으로 필드를 선언하고 Setter 메소드로 외부에서 구현 객체를 받아 필드에 대입합니다. 선택 이벤트가 발생했을 때 인터페이스를 통해 구현 객체의 메소드를 호출합니다.

```
package sec02.verify.exam03;

public class CheckBox {
    OnSelectListener listener;

    void setOnSelectListener(OnSelectListener listener) {
        this.listener = listener;
    }

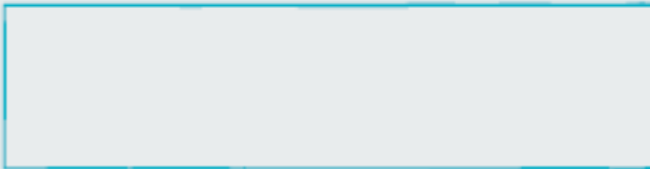
    void select() {
        listener.onSelect();
    }


    static interface OnSelectListener {
        void onSelect();
    }
}
```

# 확인문제

- 다음 CheckBoxExample 클래스를 실행했을 때 다음과 같은 실행결과가 출력되도록 익명 구현 객체를 작성해보세요

```
package sec02.verify.exam03;

public class CheckBoxExample {
    public static void main(String[] args) {
        CheckBox checkBox = new CheckBox();
        checkBox.setOnSelectListener(
            
        );
        checkBox.select();
    }
}
```

 실행결과 ×

배경을 변경합니다.

# Thank You !

혼자 공부하는 자바 (신용권 저)