

# JAVA Inner Class

프로그램은 사람이 이해하는 코드를 작성.  
느려도 꾸준하면 경기에서 이긴다.

# Content

---

## 9. Inner Class

1. Inner Class
2. 익명(Anonymous) Inner Class
3. Inner Interface

## “ Class 내부에 Class가 있는 Class”

### 종류

- Instance Member Inner Class, Static Member Inner Class, Local Inner Class가 있다.
- Class Member 처럼 사용 하는 Instance Member Inner Class, Static Member Inner Class
- Method내부에서 한정적으로 사용 하는 Local Inner Class

```
abstract class AnonymousAbstract {  
    abstract void doWork();  
    abstract void doWork(String str);  
}
```

익명(Anonymous) Class

```
AnonymousAbstract workerMember = new AnonymousAbstract() {  
    String getWorkerName() {  
        return workerName;  
    }  
    @Override  
    void doWork() {  
        System.out.println("작업자 : " + workerName);  
    }  
    @Override  
    void doWork(String str) {  
        System.out.println("작업자 : " + str);  
    }  
};
```

Inner Interface : class 를 interface로 만든 것

Instance Member Inner Class

Static Member Inner Class

Local Inner Class

```
public class Class {  
  
    // instance Memeber Class  
    class inClass {  
  
    }  
  
    // static Memeber Class  
    static class StaticInnerClass {  
  
    }  
  
    // 생성자  
    Class() {}  
  
    void method() {  
        class localInnerClass {  
  
        }  
    }  
}
```

# 2. Instance Member Inner Class

## 9. Inner Class 9-1. Inner Class

### Instance Member Inner Class

- 객체 내부에 있는 클래스로 Instance Member의 특성을 가지고 있다.
- 모든 접근 지정자를 사용 할 수 있으며 실제로는 Outer Class 이다. ( 클래스\$이너클래스 )

#### 01. 인스턴스화

- 클래스 생성
- 클래스.이너클래스 참조변수 = 클래스.new 이너클래스() 으로 생성

```
public class InnerClassMain {  
    public static void main(String... args) {  
        Human human = new Human();  
        human.eat();  
        Human.Student student = human.new Student("SB-000-0001", "한국학교");  
        student.eat();  
        student.viewStudent();  
    }  
}
```

#### 02. 이너 클래스 에서 외부 클래스 사용

- 외부클래스.this. 으로 외부 클래스의 멤버, 메서드 사용  
- 하나의 파일 (.java) 내부 이너클래스에서 외부 클래스를 호출 하는 경우 new 로 생성 할 필요 가 없다.

#### 03. 이너 클래스 에서 이너 클래스 사용

- 동일 파일이 아니거나 동일 파일 내의 다른 이너 클래스를 사용 할 경우는 일반적인 클래스 생성 과 동일 하게 선언을 하여 사용 해야 한다.

```
Student(String studentId, String school) {  
    this.studentId = studentId;  
    Human.this.school = school;  
    CollegeStudent collegeStudent = new CollegeStudent();  
    collegeStudent.drinking();  
}
```

```
class CollegeStudent {  
    void drinking() {  
        System.out.println("술을 마신다.");  
    }  
}
```

```
public class Human {  
    String name;  
    String school;  
  
    void eat() {  
        System.out.println("사람이 밥을 먹는다.");  
    }  
  
    class Student {  
        String studentId;  
  
        Student(String studentId, String school) {  
            this.studentId = studentId;  
            Human.this.school = school;  
        }  
  
        void eat() {  
            System.out.println("* 학생 이너 클래스 *");  
            Human.this.eat();  
            System.out.println("학생이 밥을 먹는다.");  
        }  
  
        void viewStudent() {  
            System.out.println("학교 : " + Human.this.school);  
            System.out.println("학번 : " + this.studentId);  
        }  
    }  
}
```

Instance Member Inner Class

외부클래스명.this.멤버

외부클래스명.this.메서드

# 2. Static Member Inner Class

## 9. Inner Class 9-1. Inner Class

### Static Member Inner Class

- 객체 내부에 있는 클래스로 Static Member의 특성을 가지고 있다.
- 객체를 생성 하지 않고 바로 사용 한다.
- 정적 이너 클래스는 아우터 클래스의 정적 멤버, 메서드만 사용 가능 하다.

#### 01. static 메서드, 멤버 사용

- ① 클래스 생성 없이 직접 사용  
: 클래스.메서드

```
StaticHuman.eat();
```

#### 02. 인스턴스화

- ① 클래스 생성  
② 클래스.이너클래스 참조변수  
= new 클래스.이너클래스()  
으로 생성

```
StaticHuman staticHuman = new StaticHuman();  
System.out.println("이름 : " + staticHuman.name);  
StaticHuman.StaticStudent staticStudent =  
    new StaticHuman.StaticStudent("SB-000-0001", "한국학교");  
staticStudent.eat();  
staticStudent.viewStudent();
```

#### 03. 이너 클래스 에서 이너클래스 및 외부 클래스 사용

- ① static inner class는 static 클래스 멤버, 메서드만 호출 할 수 있다.

```
StaticStudent(String studentId, String school) {  
    this.studentId = studentId;  
    // 오류 : StaticHuman.this.name = "홍길동";  
    StaticHuman.school = school;  
    CollegeStudent collegeStudent = new CollegeStudent();  
    collegeStudent.drinking();  
}
```

```
static class CollegeStudent {  
    void drinking() {  
        System.out.println("술을 마신다.");  
    };  
}
```

외부클래스명.static멤버

외부클래스명.static 메서드

```
public class StaticHuman {  
    String name = "홍길동";  
    static String school;  
  
    static void eat() {  
        System.out.println("사람이 밥을 먹는다.");  
    }  
  
    static class StaticStudent {  
        String studentId;  
  
        StaticStudent(String studentId, String school) {  
            this.studentId = studentId;  
            // 오류 : StaticHuman.this.name = "홍길동";  
            StaticHuman.school = school;  
        }  
  
        void eat() {  
            System.out.println("* 학생 이너 클래스 *");  
            StaticHuman.eat();  
            System.out.println("학생이 밥을 먹는다.");  
        }  
  
        void viewStudent() {  
            System.out.println("학교 : " + StaticHuman.school);  
            System.out.println("학번 : " + this.studentId);  
        }  
    }  
}
```

static Member Inner Class

# 3. Local Member Inner Class

## 9. Inner Class 9-1. Inner Class

### Local Member Inner Class

- 메서드 내부에 선언 하는 클래스
- 메서드가 호출 될 때만 메모리에 생성되므로 static class로 사용 할 수 없다.

#### 01. Local Inner Class 생성

- ① Method 내부에 생성

```
LocalHuman localHuman = new LocalHuman();  
localHuman.student();
```

#### 02. Local Inner Class 인스턴스화

- ① 외부 클래스 사용시 new 연산자로 생성 하여 사용
- ② Method 내부에 객체 생성 하여 사용

```
public class LocalHuman {  
    String name ;  
    String school;  
  
    void eat() {  
        System.out.println("사람이 밥을 먹는다.");  
    }  
  
    void student() {  
        class Student {  
            String studentId = "SB-000-0000" ;  
            LocalHuman localHuman = new LocalHuman();  
  
            Student(String studentId, String school) {  
                this.studentId = studentId;  
                localHuman.school = school;  
            }  
  
            void viewStudent() {  
                System.out.println("학교 : " + localHuman.school);  
                System.out.println("학번 : " + this.studentId);  
            }  
  
            void setStudentId(String studentId) {  
                this.studentId = studentId;  
            }  
        }  
  
        Student student = new Student("SB-000-0001", "한국학교");  
        student.localHuman.eat();  
        student.viewStudent();  
        student.setStudentId("SB-000-0002");  
        student.viewStudent();  
    }  
}
```

Local Inner Class

# 1. 익명(Anonymous) Class

“ 이름을 알 수 없는 객체로 한번만 사용하고 버려지는 객체 ”

## 사용하는 이유

- 프로그램에서 일시적으로 한번만 사용되고 버려지는 객체를 매번 객체를 만들어야 하나?
- 확장성을 고려해서 객체를 생성해야 하는데 ... 수정이 편 할 까?
- 사용 처: 인스턴스 변수, 인스턴스 메서드, 인스턴스 메소드의 매개변수

## 구현 하는 방법

- 클래스 생성
- 인터페이스의 구현

### 01. 클래스 생성

#### ➤ 1. 추상 클래스 생성

예제 : AnonymousAbstract.java

```
abstract class AnonymousAbstract {  
    abstract void doWork();  
    abstract void doWork(String str);  
}
```

#### ➤ 2. 추상 클래스 구현 Class 생성

## 익명(Anonymous) Class

- 클래스를 정의하지 않고 필요할 때 이름없이 즉시 선언하고 인스턴스화 해서 사용
- 객체 안에 만드는 로컬 클래스와 동일 하다
- new 수식이 올 수 있는 곳 어디든지 사용 가능하나 생성자는 정의 할 수 없음
- 익명 클래스내부에서 외부의 메소드 내 변수를 참조할 때는 메소드의 지역 변수 중 final로 선언된 변수만 참조 가능
  - 변수는 Stack에 있고 객체는 Heap에 있음, 즉 Method 실행 이 끝나고 Stack는 사라지지만 Heap에 있는 Method는 사라지지 않기 때문

예제 : AnonymousAbstractClass.java - 인스턴스 변수

```
// 인스턴스 변수  
AnonymousAbstract workerMember = new AnonymousAbstract() {  
    String getWorkerName() {  
        return workerName;  
    }  
    @Override  
    void doWork() {  
        System.out.println("작업자 : " + workerName);  
    }  
    @Override  
    void doWork(String str) {  
        System.out.println("작업자 : " + str);  
    }  
};
```

# 1. 익명(Anonymous) Class

## 9. Inner Class

### 9-2. 익명(Anonymous) Inner Class

#### ➤ 2 추상 클래스 구현 Class 생성

예제 : AnonymousAbstractClass.java - 인스턴스 Method

```
// 인스턴스 Method
void workerMethod(String workerNm) {
    AnonymousAbstract worker = new AnonymousAbstract() {
        @Override
        void doWork() {
            System.out.println("기본 작업자 : " + workerName + ", 작업자 : " + workerNm);
        }
        @Override
        void doWork(String str) {
            System.out.println("작업자 : " + str);
        }
    };
    worker.doWork();
    worker.doWork("홍길동");
}
```

기본 작업자 : 기본작업자, 작업자 : 인스턴스의 지역변수  
작업자 : 홍길동

#### ➤ 2 추상 클래스 구현 Class 생성

예제 : AnonymousAbstractClass.java - 인스턴스 메서드의 파라미터

```
// 인스턴스 메서드의 파라미터
void workerMethod(AnonymousAbstract worker) {
    worker.doWork();
    worker.doWork("익명객체 파라미터");
}
```

기본 작업자 없습니다.  
작업자 : 익명객체 파라미터

#### ➤ 3. 실행

예제 : AnonymousMain.java

```
public static void main(String... args) {
    AnonymousAbstractClass anonymousClass = new AnonymousAbstractClass();
    anonymousClass.workerMember.doWork();
    anonymousClass.workerMember.doWork("인스턴변수");
    // anonymousClass.workerMember.getWorkerName() 익명 함수 참조 불가
    anonymousClass.workerMethod("인스턴스의 지역변수");

    anonymousClass.workerMethod(new AnonymousAbstract() {
        @Override
        void doWork() {
            // private String workerName 접근 못함
            System.out.println("기본 작업자 없습니다.");
        }
        @Override
        void doWork(String str) {
            System.out.println("작업자 : " + str);
        }
    });

    AnonymousChild anonymousChild = new AnonymousChild();
    String str = anonymousChild.action.action("홍길동");
    System.out.println(str);

    anonymousChild.actionMethod("인스턴스메서드");
}
```

작업자 : 기본작업자  
작업자 : 인스턴변수

파라미터로 객체 생성 하여 파라미터로 전달

예제 : AnonymousChild .java 참조

작업자 : 홍길동  
작업자 : 인스턴스메서드



# 1. 익명(Anonymous) Class

## 9. Inner Class

### 9-2. 익명(Anonymous) Inner Class

#### 02. 인터페이스의 구현

##### ➤ 1. 인터페이스 구현

예제 : AnonymousInterface.java

```
package com.hyomee.lambda;  
  
public interface AnonymousInterface {  
    public String action(String str);  
}
```

##### ➤ 2 인터페이스 구현 체

예제 : AnonymousInterfaceClass.java

```
package com.hyomee.lambda;  
  
public class AnonymousInterfaceClass implements AnonymousInterface {  
  
    @Override  
    public String action(String str) {  
        return str;  
    }  
}
```

##### ➤ 3. 실행

```
public static void main(String... args) {  
    AnonymousInterfaceClass ac01 = new AnonymousInterfaceClass();  
    System.out.println("객체 지향으로 함수형 : " + ac01.action("기본"));
```

객체 지향으로 함수형 : 기본

```
    AnonymousInterface ac02 = new AnonymousInterface() {  
        @Override  
        public String action(String str) {  
            return str;  
        }  
    };  
};
```

익명 객체 생성

```
    System.out.println("익명 미너 클래스 : " + ac02.action("익명"));
```

익명 이너 클래스 : 익명

```
    AnonymousInterface ac03 = (String str) -> { return str; };  
    System.out.println("람다 : " + ac02.action("람다"));  
}
```

람다 생성

람다 : 람다

## “ 정적(static) Inner Interface만 존재 ”

### Inner Interface

- 정적 이너 클래스만 존재
- static 생략 시 자동으로 추가 됨

#### 01. Interface Inner Class 생성

- ① Method 내부에 생성  
- static 생략 하면 자동으로 추가 됨

#### 02. 인스턴스화

- ① 클래스 생성
- ② 클래스.이너클래스 참조변수  
= 클래스.new 이너클래스()  
으로 생성

```
public class InterfaceInnerClassMain {  
    public static void main(String... args) {  
        InnerfaceHuman innerfaceHuman = new InnerfaceHuman();  
        InnerfaceHuman.student student =  
            innerfaceHuman.new StudentImpl("SB-000-0001", "한국 학교");  
        student.viewStudent();  
    }  
}
```

#### 03. 이너 클래스 에서 외부 클래스 사용

- ① 외부클래스.this. 으로 외부 클래스의 멤버, 메서드 사용  
- 하나의 파일 (.java) 내부 이너클래스에서 외부 클래스를 호출 하는 경우  
new 로 생성 할 필요 가 없다.

```
public class InnerfaceHuman {  
    String name ;  
    String school;   
  
    void eat() {  
        System.out.println("사람이 밥을 먹는다.");  
    }  
  
    static interface student {  
        void viewStudent();  
    }  
  
    class StudentImpl implements InnerfaceHuman.student {  
        String studentId = "SB-000-0000" ;  
  
        StudentImpl(String studentId, String school) {  
            this.studentId = studentId;  
            InnerfaceHuman.this.school = school;  
        }  
  
        @Override  
        public void viewStudent() {  
            System.out.println("학교 : " + InnerfaceHuman.this.school);  
            System.out.println("학번 : " + this.studentId);  
        }  
    }  
}
```

Interface Inner Class