코틀린을 활용한 안드로이드 앱 개발

코틀린 객체지향 프로그래밍 이해
- Class, Constructor

한국기술교육대학교
온라인평생교육원

학습내용

1. 클래스 선언법 이해
2. 주 생성자와 보조 생성자의 차이

학습목표

1. 코틀린의 클래스 선언법을 이해할 수 있다.
2. 주 생성자와 보조 생성자의 차이를 이해할 수 있다.

1. 클래스 선언법 이해

1 클래스 선언

- 클래스(class) 예약어로 선언
- 클래스 몸체가 없다면 { }는 생략 가능

```
class MyClass { }

class MyClass
```

- 클래스는 멤버 변수, 함수, 생성자, 다른 클래스로 구성

```
class MyClass {
    val myVariable=10
    constructor(){ }
    fun myFun(){ }
    class Inner {
    }
}
```

2 객체 생성

- 클래스의 객체 생성은 new 연산자를 사용하지 않으며 생성자 호출로 생성 됨

```
class MyClass {
    var name: String="world"

    fun sayHello(){
        println("hello $name")
    }
}

fun main(args: Array<String>) {
    val obj1=MyClass()
    val obj2=MyClass()
}
```

2. 주 생성자와 보조 생성자의 차이

1 주 생성자

- 매개변수를 가지는 주 생성자
- 주 생성자의 constructor 예약어는 생략이 가능

```
class User1 constructor(name: String, age: Int){ }
class User2(name: String, age: Int){ }

val user1=User1()//error
val user2=User1("kkang", 33)
val user3=User2("kim", 28)
```

- 생성자 매개변수 기본값 명시

```
class User3(name: String, age: Int = 0) { }

val user4=User3("kkang", 33)
val user5=User3("kkang")
```

생성자 초기화 블록

- init 예약어로 명시

```
class User4(name: String, age: Int) { }{ //error

}
```

```
class User4(name: String, age: Int) {
    init {
        println("i am init...")
    }
}
.val user6=User4("kkang", 33)
```

2. 주 생성자와 보조 생성자의 차이

1 주 생성자

생성자 매개변수 값 이용

- 클래스의 초기화 블록, 클래스 프로퍼티에서는 접근이 되지만 클래스에 정의된 함수에서는 사용이 불가능

```
class User5(name: String, age: Int){
    init {
        println("i am init... constructor argument : $name .. $age")
    }
    val upperName=name.toUpperCase()
    fun sayHello(){
        println("hello $name")//error
    }
}
```

- 생성자 내에서 val, var 을 이용해 매개변수를 선언

```
class User6(val name: String, val age: Int){
    val myName=name
    init {
        println("i am init... constructor argument : $name .. ${age}")
    }
    fun sayHello() {
        println("hello $name")
    }
}
```

2. 주 생성자와 보조 생성자의 차이

2 보조 생성자

- 주 생성자와 보조 생성자로 구분
- 주 생성자는 클래스 선언 영역에 작성하는 생성자이며 보조 생성자는 클래스 바디 영역에 constructor 예약어로 선언하는 생성자
- 클래스 선언 시 최소한 하나 이상의 생성자는 정의되어 있어야 함

```
//컴파일러에 의해 매개변수 없는 주 생성자 자동 추가
class User1 { }
//주생성자만 선언
class User2(name: String) { }
//보조 생성자만 선언
class User3 {
    constructor(name: String){ }
}
fun main(args: Array<String>) {
    val user1=User1()
    val user2=User2("kkang")
//    val user3=User3()//error
    val user4=User3("kkang")
}
```

2. 주 생성자와 보조 생성자의 차이

2 보조 생성자

- 생성자 오버로딩으로 보조 생성자는 하나의 클래스에 여러 개 선언이 가능

```
class User4 {
    constructor(){}
    constructor(name: String){}
    constructor(name: String, age: Int){}
}//.................
val user5=User4()
val user6=User4("kkang")
val user7=User4("kkang", 10)
```

- 보조생성자의 매개변수는 var, val로 선언 할 수 없음

```
class User6 {
    constructor(val name: String){//error
        println("i am constructor....$name")
    }
}
```

2. 주 생성자와 보조 생성자의 차이

3 This()

생성자 연결

- 주 생성자가 선언되어 있다면 보조 생성자는 무조건 주 생성자를 같이 호출

```
class User1(name: String){

    constructor(name: String, age: Int){//error

    }
}
```

```
//주 생성자와 보조생성자 같이 선언
class User1(name: String){
    init {
        println("init block... $name")
    }
    constructor(name: String, age: Int): this(name){
        println("constructor ... $name ... $age")
    }
}
```

정리하기

1. 클래스 선언법 이해

Class

- class 예약어로 선언
- 클래스는 멤버 변수, 함수, 생성자, 다른 클래스로

2. 주 생성자와 보조 생성자의 차이

Constructor

- 주 생성자는 클래스 선언부분에 작성
- 보조 생성자는 클래스 바디 영역에 constructor 예약어로 선언하는 생성자