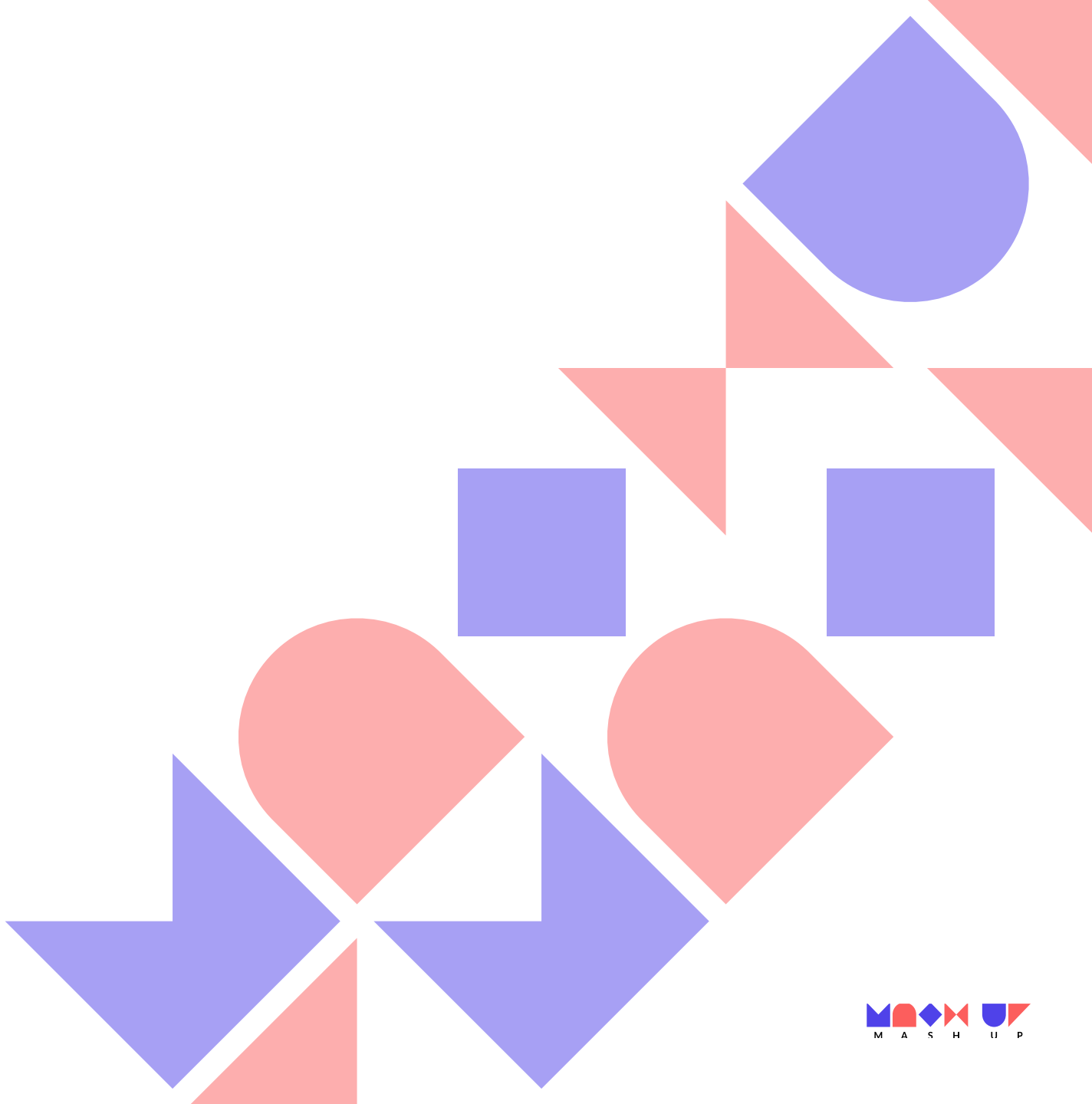
2020년 12월 9일

CHAP 3.
함수 정의와 호출

안드로이드 10기
양민욱



여러가지 구조로 자료를 저장/ 관리하는 **클래스**

```
val hashSet = hashSetOf(1, 7, 3)

val arrayList = arrayListOf(1, 7, 3)

val hashMap = hashMapOf(1 to "one", 7 to "seven",
                        53 to "fifty-three")
```

Kotlin의 **개발 방향성**

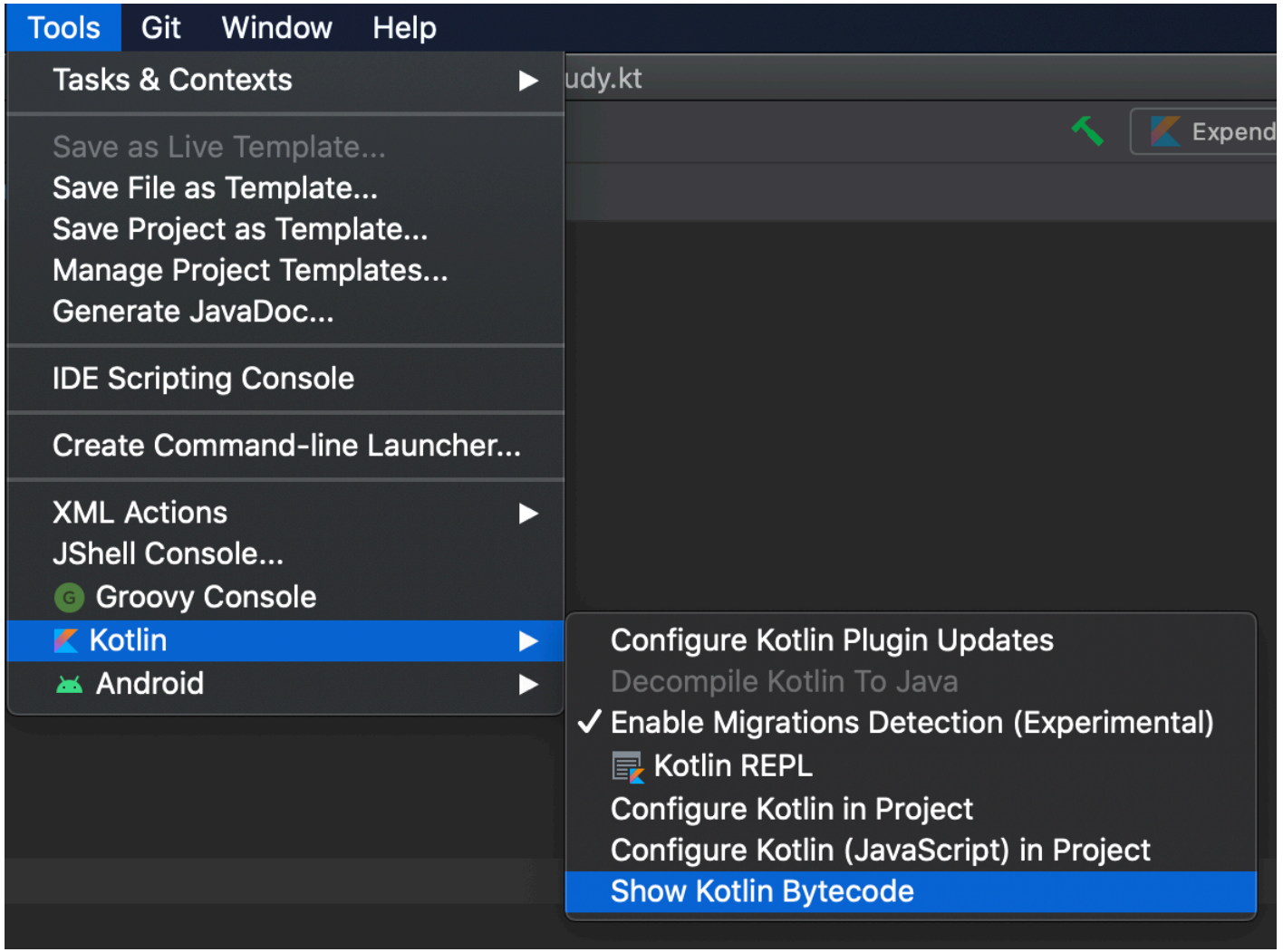
```
val set1: HashSet<Int> = HashSet();

val set2 = emptySet<Int>()
val set3 = hashSetOf(1, 7, 3)
```

확장성

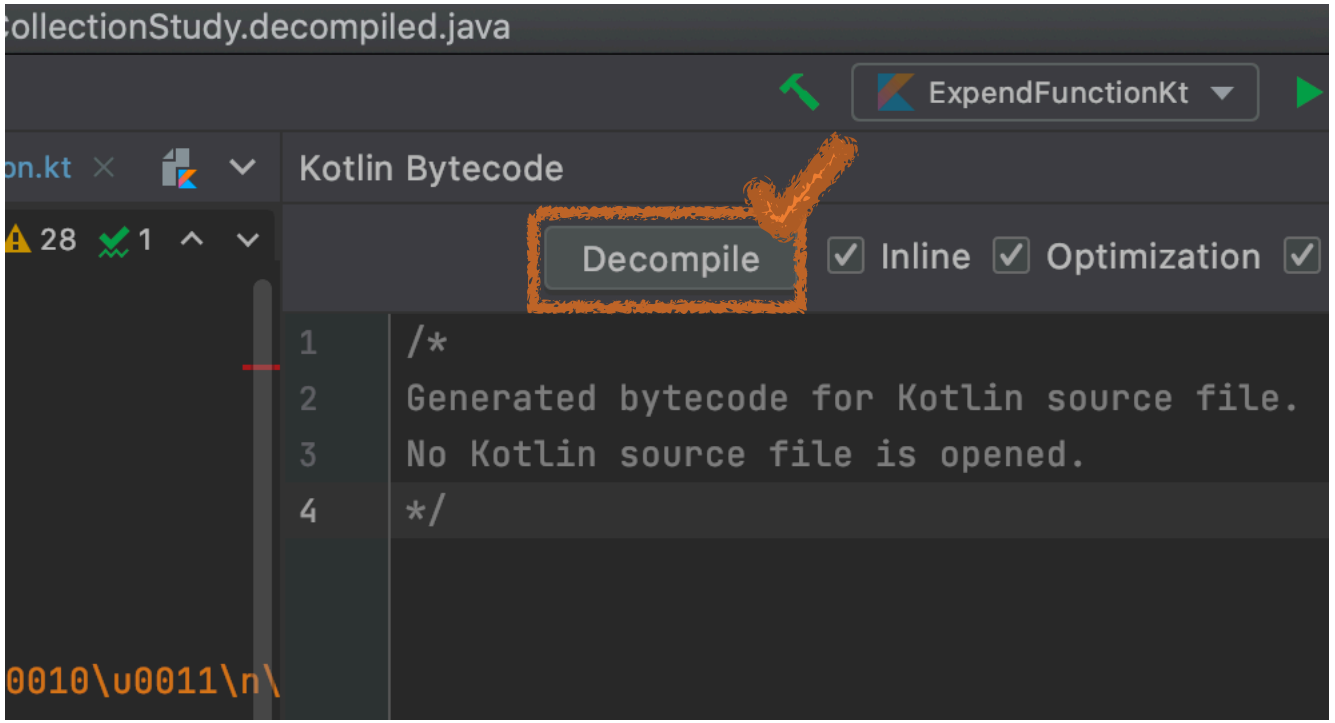
```
val hashSet = hashSetOf(1, 7, 3)
println(hashSet.javaclass)

>> class java.util.HashSet
```

호환성

Kotlin 코드를 자바로



Kotlin 코드를 자바로



어느새 익숙한..

TopFunction.kt

```
var studyDay = 2

fun main(args: Array<String>) {

  println("Hello Mash-Up")
}
```

어느새 익숙한..

```java
public final class TopFunctionKt {

   private static int studyDay;

   public static final int getStudyDay() {
      return studyDay;
   }

   public static final void setStudyDay(int var0) {
      studyDay = var0;
   }

   public static final void main() {
   }

   // $FF: synthetic method
   public static void main(String[] var0) {
      main();
   }
}
```

디폴트 파라미터

arrayListOf(1, 7, 3)의 **toString()** => [1, 7, 4]

```
fun <T> joinToString(
    collection: Collection<T>,
    separator: String,
    prefix: String,
    postfix: String,
): String {
    val result = StringBuilder(prefix)

    for ((index, element) in collection.withIndex()) {
        if (index > 0) result.append(separator)
        result.append(element)
    }

    result.append(postfix)
    return result.toString()
}
```

디폴트 파라미터

```
println(joinToString(collection = list, separator = " ",
                prefix = " ", postfix = "."))
>> 1 2 3.

println(joinToString(collection = list, separator = " separator ",
                prefix = " prefix ", postfix = " postfix "))

>> prefix 1 separator 2 separator 3 postfix
```

디폴트 파라미터

```
fun <T> joinToString(
    collection: Collection<T>,
    separator: String = ", ",
    prefix: String = "",
    postfix: String = "",
): String {
    val result = StringBuilder(prefix)

    for ((index, element) in collection.withIndex()) {
        if (index > 0) result.append(separator)
        result.append(element)
    }

    result.append(postfix)
    return result.toString()
}
```

디폴트 파라미터

```
println(joinToString(list))
>> 1, 2, 3

println(joinToString(list, prefix = "<", postfix = ">"))
>> <1, 2, 3>
```

Java 에서 접근은?

```
   // $FF: synthetic method
   public static String joinToString$default(Collection var0, String var1, String var2, String var3,
int var4, Object var5) {
      if ((var4 & 2) != 0) {
         var1 = ", ";
      }

      if ((var4 & 4) != 0) {
         var2 = "";
      }

      if ((var4 & 8) != 0) {
         var3 = "";
      }

      return joinToString(var0, var1, var2, var3);
   }

   joinToString$default((Collection)list, (String)null, "<", ">", 2, (Object)null);
```

디폴트 파라미터

```kotlin
@JvmOverloads
fun <T> joinToString(
    collection: Collection<T>,
    separator: String = ", ",
    prefix: String = "",
    postfix: String = "",
): String {
 ... 생략
}
```

```
public static final String joinToString(@NotNull Collection collection)

public static final String joinToString(@NotNull Collection collection, @NotNull String separator)

public static final String joinToString(@NotNull Collection collection, @NotNull String separator,
@NotNull String prefix)

public static final String expendJoinToString(@NotNull Collection $this$expendJoinToString, @NotNull
String separator, @NotNull String prefix, @NotNull String postfix)
```

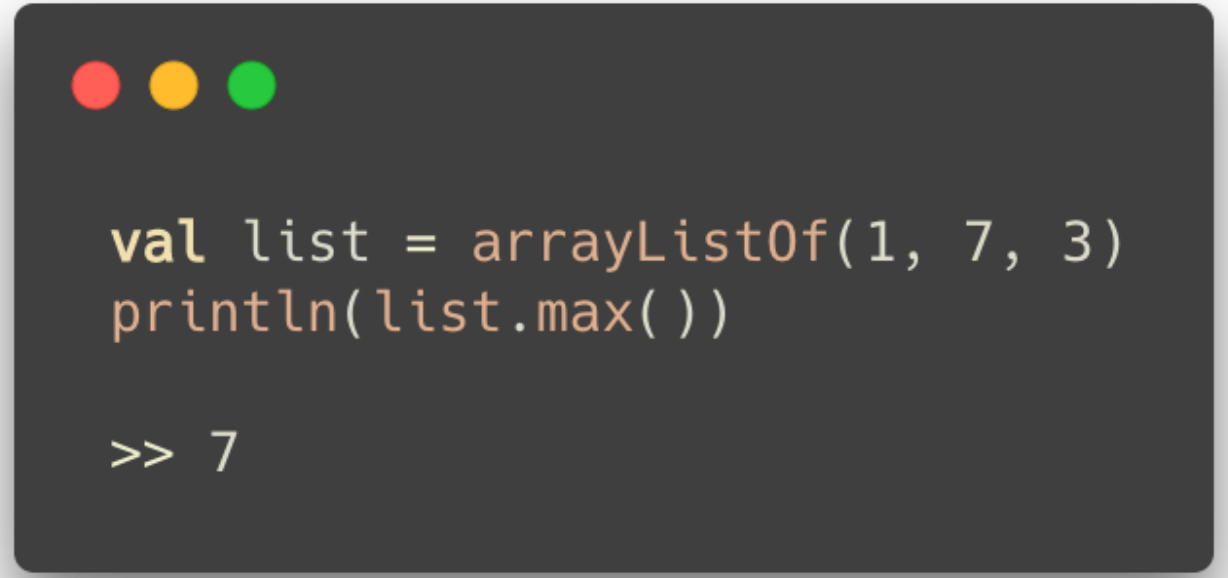
어디에 적용할 수 있을까요?

실제 적용 사례 - CircleTimer CustomView

```java
public CircleTimer(Context context) {
      this(context, null, 0);
  }


  public CircleTimer(Context context, AttributeSet attrs) {
      this(context, attrs, 0);
  }

  public CircleTimer(Context context, AttributeSet attrs, int defStyle) {
      super(context, attrs, defStyle);
      init(attrs, defStyle);
  }
```

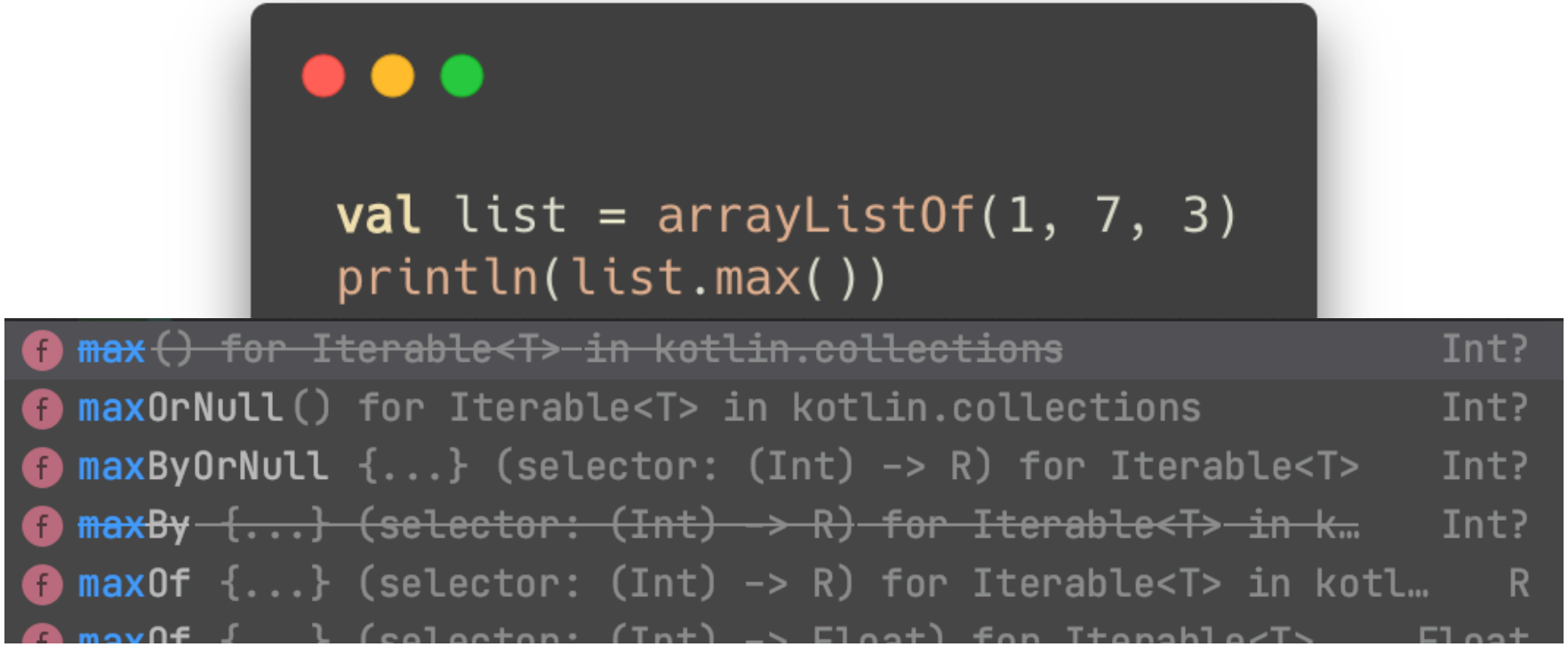
실제 적용 사례

```
class CircleTimer @JvmOverloads constructor(
    context: Context,
    attrs: AttributeSet? = null,
    defStyleAttr: Int = 0
) : View(context, attrs, defStyleAttr) {
```



```
val list = arrayListOf(1, 7, 3)
println(list.max())

>> 7
```

여기서 잠깐?..


val list = arrayListOf(1, 7, 3)
println(list.max())
max() for Iterable<T> in kotlin.collections Int?
maxOrNull() for Iterable<T> in kotlin.collections Int?
maxByOrNull {...} (selector: (Int) -> R) for Iterable<T> Int?
maxBy {...} (selector: (Int) -> R) for Iterable<T> in k… Int?
maxOf {...} (selector: (Int) -> R) for Iterable<T> in kotl… R

Kotlin 1.4

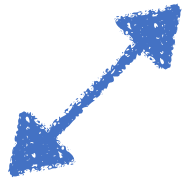
- 표준 라이브러리 일관성 개선

함수	동작
String.toInt()	문자열을 정수로 변환합니다. 문자열이 숫자가 아닌 경우 NumberFormatException 예외가 발생합니다.
String.toIntOrNull()	문자열을 정수로 변환합니다. 문자열이 숫자가 아닌 경우 null을 반환합니다.

확장 함수

```kotlin
@Deprecated("Use maxOrNull instead.", ReplaceWith("maxOrNull()"))
@DeprecatedSinceKotlin(warningSince = "1.4")
public fun <T : Comparable<T>> Iterable<T>.max(): T? {
    return maxOrNull()
}

println(list.maxOrNull())
```

수신 객체 타입



```
fun String.lastChar() : Char = this.get(this.length - 1)
```

수신 객체

```kotlin
fun <T> Collection<T>.expendJoinToString(
    separator: String = ", ",
    prefix: String = "",
    postfix: String = "",
): String {
    val result = StringBuilder(prefix)

    for ((index, element) in this.withIndex()) {
        if (index > 0) result.append(separator)
        result.append(element)
    }

    result.append(postfix)
    return result.toString()
}
```

클래스의 확장?

```
@NotNull
   public static final String expendJoinToString(@NotNull Collection $this$expendJoinToString, @NotNull
String separator, @NotNull String prefix, @NotNull String postfix) {

   }
```

클래스의 확장은 아니며, 매개 변수로 수신 객체를 접근하는 방식

-> 수신 객체의 private, protected 멤버 접근 불가

어디에 적용할 수 있을까요?

실제 적용 사례

```
fun Activity.showToast(toastText: String, timeLength: Int) {
    Toast.makeText(this, toastText, timeLength).show()
}

//in Activity
this.showToast("간결!", Toast.LENGTH_SHORT)
```

kotlin의 to 그리고 until은 Keyword가 아니다.

```kotlin
val map = hashMapOf(1 to "one", 7 to "seven",
        53 to "fifty-three")

for (index in 1 until 10) {
        println(index)
}
```

중위 표기법 inflx

- 함수가 인자가 **하나**
- 멤버 함수이거나 확장 함수
- **가변 인자**이거나 **디폴트 파라미터**면 안된다.

```kotlin
public infix fun <A, B> A.to(that: B): Pair<A, B> = Pair(this, that)

public infix fun Int.until(to: Int): IntRange {
    if (to <= Int.MIN_VALUE) return IntRange.EMPTY
    return this .. (to - 1).toInt()
}
```

오늘의 내용은

- **Kotlin** Collection은 Java Collection 객체를 그대로 사용하여 Java와의 **호환성**을 더불어 maxOrNull 등 **다양한 표준 라이브러리**를 추가로 제공해준다.

- **Kotlin**은 메소드와 프로퍼티가 **최상위**로 존재할 수 있다. 따라서 클래스 내에서만 선언되지 않아도 된다.

- **디폴트 파라미터**로 자바의 불필요한 **오버로딩**를 제거해보자

- **확장 함수**는 기존 클래스에 정의된 함수처럼 새로운 기능을 추가할 수 있다.

- 함수 인자가 **하나**일 경우 **중위 표기법**으로 가독성 좋게 표현할 수 있다.

THANKS FOR WATCHING

감사합니다