T315067 服部 颯太

Kotlinレクチャー

目次

- Kotlinの概要
- Kotlinの基本的文法
- Kotlinの特徴的なところ
- KotlinをつかってAndroidアプリ開発

今回の資料はすべてここに置いてます

https://github.com/hato52/KotlinLecture

Kotlinとは?

- JetBrains社が2011年に開発したJVM言語
- 2017年にGoogleがAndroid開発言語として正式サポート
- 静的型付けのオブジェクト指向言語

Kotlinとは?

- Javaをより書きやすくすることを目標に開発

- Javaと完全に相互運用可能

↓

  - JavaのライブラリをKotlinで使用可能
  - Javaで開発されたプロジェクトに追加で混ぜ込むことも可能

Kotlinの基本的文法

ここのページでKotlinを動かすことができます

https://try.kotlinlang.org

Hello World!

Kotlin

```kotlin
fun main(args: Array<String>) {
    println("Hello World!")
}
```

class必要なし
セミコロン必要なし
publicもstaticもvoidも必要なし
printlnはクラスに属さない関数

Java

```java
public class Sample {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

変数宣言

Kotlin

```kotlin
var num: Int = 100
val word: String = "abc“

//これでもOK
var num = 100
val word = "abc"
```

var = 変数
val = 定数
後ろの型指定は省略可

Java

```java
int num = 100;
final String word = "abc";
```

if式

Kotlin

```kotlin
val word = if (num == 0) {
    "abc"
}
```

Kotlinのifは文ではなく式
なので値を返せる

Java

```java
final String word;
if (num == 0) {
    word = "abc";
}
```

for文

Kotlin

```kotlin
for (i in 0..100) {
    println(i)
}
```

Javaでいう拡張for文のみ存在
PHPのforeachやPythonのfor文のように使える

Java

```java
for (int i=0; i<100; i++) {
    System.out.println(i);
}
```

when式

Kotlin

```kotlin
val word = when (num) {
    0 -> "abc"
    1,2 -> "def"
    "number" -> "xyz"
}
```

ラムダで書ける
break必要なし
比較対象は違う型でもOK

Java

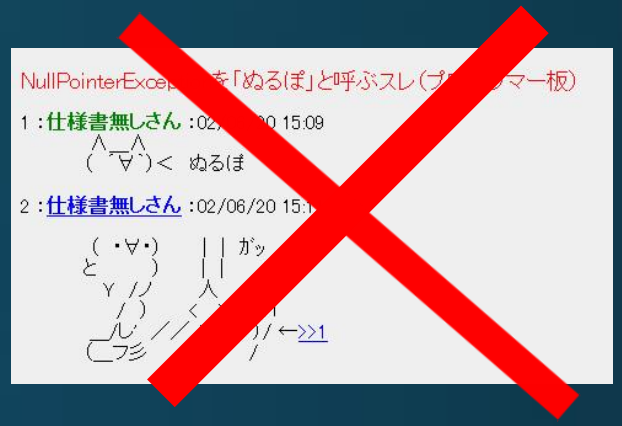
```java
final String word;
switch (num) {
    case 0:
        word = "abc";
        break;
    case 1:
    case 2:
        word = "def";
        break;
}
```

Kotlinの特徴的なところ

- null安全
- プロパティ
- コレクション
- companion object

null安全

- ぬるぽ撲滅用の機能



- 許可されていないnullはコンパイルエラーになる

```
var word: String = null //この段階でエラー
println(word)
```

null安全

- ?をつけるとnullable(nullを許可)

- nullチェックがないとコンパイルエラー

```
var word: String? = null //nullが入ることを許可

if (s != null) { //nullチェックをしないとエラーになる
    word.length
}
```

null安全

- !!をつけると強制的にnon-nullに変換

```
var word: String? = "hoge" //この時点でwordはString?型

word!!.length //強制的にString型へと変換
```

プロパティ

- 内部でgetterとsetterを自動生成する
  - 簡潔な記述でカプセル化可能

```
class Employee {
    var name: String? = null
    var salary: Int? = null  //getterとsetterは必要なし
}

fun main(args: Array<String>) {
    val employee = Employee()
    employee.name = "ラ王"
    employee.salary = 10000000

    println(employee.name) //通常のフィールドのように使える
    println(employee.salary.toString())
}
```

コレクション

- ListとかMapとか

- 読み取り専用と書き込み可能の2種類ある

```
//リスト
val numList1 = listOf(1,2,3) //変更不可
val numList2 = mutableListOf(1,2,3) //変更可能

numList1[0] = 10 //変更不可なのでエラーになる
numList2[1] = 10 //これは通る

//マップ
val fruit = mapOf("apple" to 1, "banana" to 2, "orange" to 3)
println(fruit["apple"])
```

companion object

- KotlinにはJavaでいうstatic宣言が存在しない
- companion object内で定義すればstaticになる

```
companion object {
    val str: String = "hoge"

    fun fuga() {...}
}
```

ここからAndroidアプリ作成

- 簡単なメモ帳アプリの作成