Kotlinの概要

**・Kotlinってなんだろう？**

2011年7月、カリフォルニアで開かれたJVM Language Summitというイベント中で初めて登場した。

↓

発表したのは、『IntelliJ IDEA』などのIDE（Integrated Development Environment:統合開発環境）製品の開発元であったJetBrain社

・なぜ今Kotlinを使うのか？

現在は、プログラミング言語がたくさん！

その中でもJavaは有力な選択肢となっている。

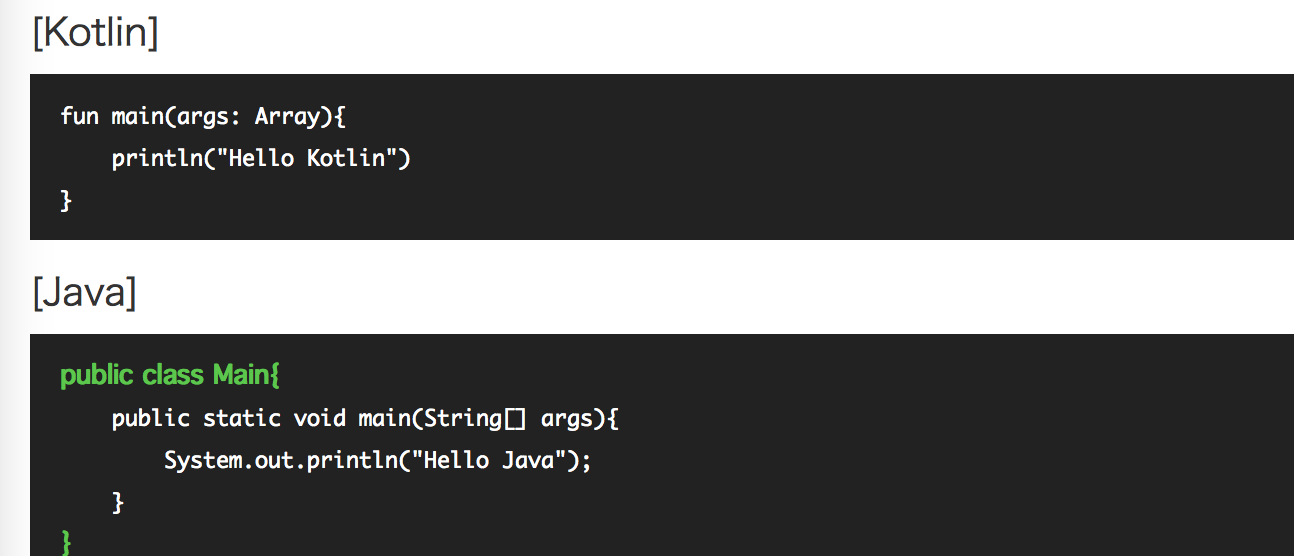
Javaの問題は、記述の単調さ、型安全性の問題、後方互換の維持などがある。

Kotlinでは多くの概念はJavaと同じ。

**Javaとどう違うのか？**

Mainメソッドはクラスの外

Javaではmainメソッドはpublicなクラスのstaticメンバとして書く必要があったが、Kotlinでは、いきなりメインメソッドを書くことができる。



**セミコロンレス**

Javaでは文末のセミコロンが抜けるといちいち怒られるが、Kotlinでは改行するだけで文末とみなされる。一行に複数の文を書きたい場合はセミコロンがいる。

**基本データ型（プリミティブ型）は無し**

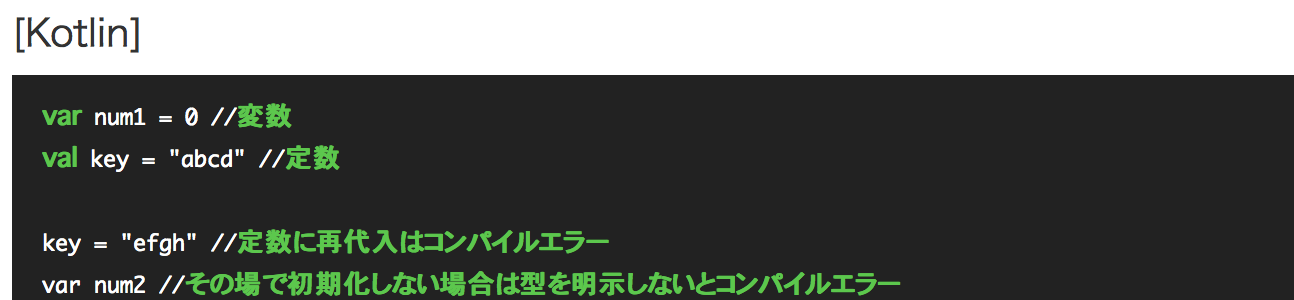
**Javaではintやfloatやcharなどの基本データ型が用意されているがKotlinではない。その代わりに扱えるクラス（オブジェクト）が用意されている。**

****

**型の宣言不要**

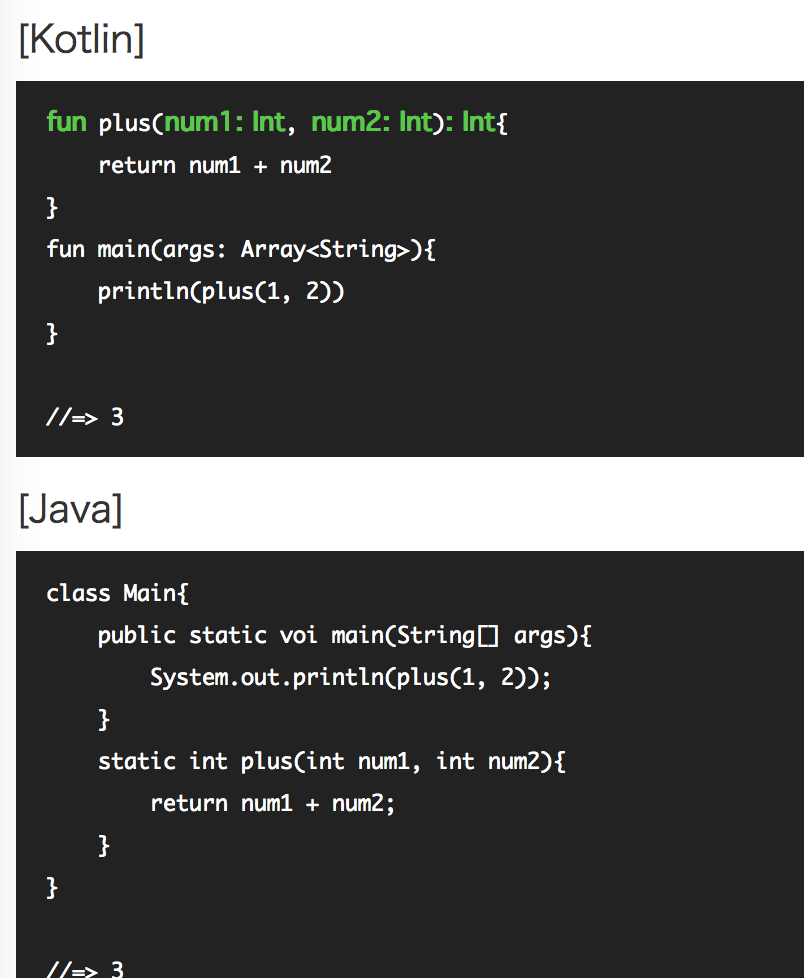
Kotlinは型推論が備わっており、変数の宣言時に初期化（値の代入）を行う場合は型を書く必要はない。

その代わり、変数の宣言時にはvarかvalをつける必要がある。Varは普通の変数、valは定数となり再代入不可になる。（javaのfinalと同じ）

****

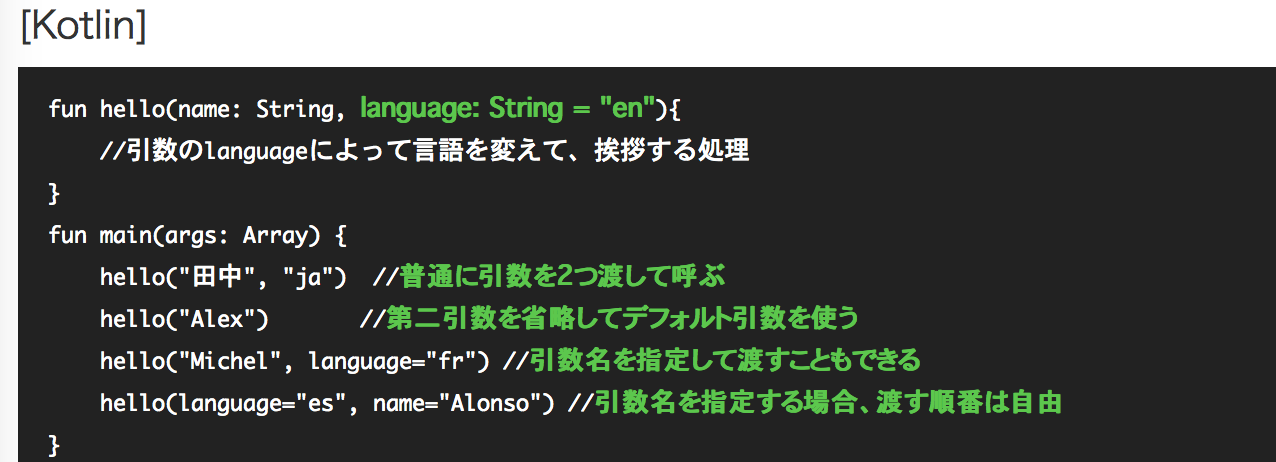
**関数（メソッド）の定義の仕方**

関数クラスのメンバである必要はなく、トップレベルで宣言することができる。関数に宣言する際にはfunを頭につける。仮引数は型を明示する必要がある。戻り値がある場合は（）の後ろにコロンをつけて書き足す。戻り値がない場合は何も書かない。



**引数のデフォルト値を設定できる**

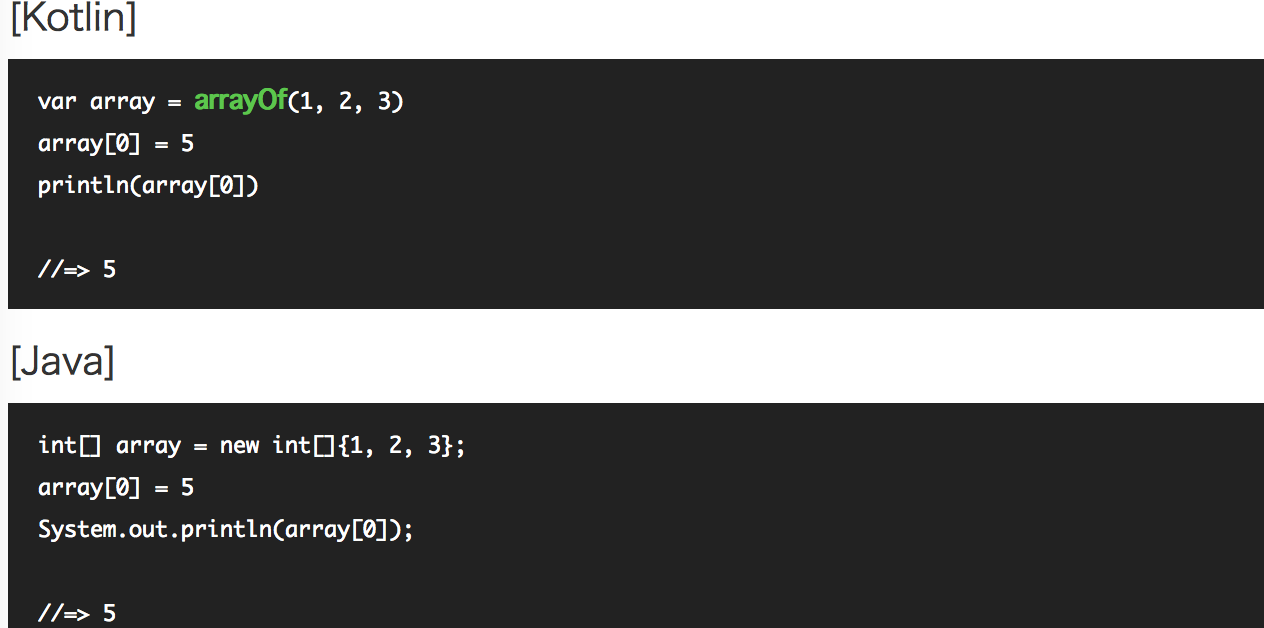
Kotlinでは引数のデフォルト値を設定することができる。



**配列はArrayオブジェクト**

Kotlinではいわゆる配列はない。その代わりにArrayクラスを使う。

Arrayオブジェクトを作る際には、arrayOfメソッドを使う。要素にアクセスする際には普通に[]でインデックスを指定することができる。



**コネクションは読み込み専用＆書き込み可**

KotlinのコネクションはList,Map,Setの３つクラスが用意される。それぞれ基本的には読み取り専用になっていて、要素を一度入れたら書き換えることができない。（値を変更するメソッドが用意されない）書き込み（要素の変更）もしたい場合はそれぞれ**MutableList**、**MutableMap**、**MutableSet**クラスを使う必要があります。

**Newは無し、コンストラクタはinit**

インスタンス化する際のnewは要らない。コンストラクタは、init{}で定義する。コンストラクタに渡したい引数はクラス名の後ろに（）をつけてそこにかく。



**プロパティは自動でセット可能**

コンストラクタに渡した値をそのままプロパティに代入したいようなパターンが多くあると思う。上の書いたプログラムもそうである。コンストラクタが受け取るべき仮引数を定義する際に、varかvalを書いて宣言することで、その変数をそのクラスのプロパティ（インスタンスフィールド）として宣言したことになり、しかも、そのプロパティへ代入する処理をコンストラクタに書かなくても良い。



**文字列の中に変数を入れる**

ダブルクォーテーションで囲った文字列リテラルの中に変数を放り込むことができる。変数名の頭に＄をつけて放り込むだけでできる。



ただし、この場合、変数名の後ろに文字が続く場合境目をうまく区切ってくれないことがある。



＄{}を使えば明示的に変数部分を指定できる。



・**JVMについて**

Javaが動作する仮想マシンJVM（**Java Virtual Machine**）他に「Java仮想マシン

」、「JavaVM」ということもある。

JVMはJavaを動かすために必要なソフトウェアであって、高性能で高信頼性を備えていて、非常に魅力的である。Javaのための資産として、ライブラリーやフレームワーク、ツールなどが豊富！Kotlinプログラムは、JVMやJava用のライブラリ、フレームワークといった資産をそのまま使える。

さらに「KotlinとJavaの相互運用性は100％」100％ということはJavaで記述されたプログラムを、Kotlinで記述したプログラムから使用することと、その逆が可能である。

*KotlinはJavaよりもシンプルかつ安全に設計されている。*

**なぜ安全なのか？**

・Kotlinには、よくあるプログラミングミスを未然に防ぐための仕組みが備わっている。

・型やnullの扱いが厳格である。キャストやnullのデリファレンスによる実行事例外が起こることはまれ。特にnullにまつわる安全確保の仕組みを「Null安全」という。

Null安全とは、JavaがNullPointerExceptionが出た時に、kotlinの時に実行エラーになるかもしれないコードをkotlinコンパイラが検出して、コンパイル時に教えてくれる。エラーの検出は遅れれば遅れるほど対処が困難になる。実行エラーはテストで検出できなければ、潜在バグとしてプロダクトを埋め込まれてリリースされてしまうかもしれない。コンパイルエラーであれば解消されないままリリースされるということはない。

このNull安全は実際にプログラムを書く時に理解できると思う。

参考文献：JavaプログラマがKotlinでつまづきがちなところ　<https://qiita.com/koher/items/d9411a00986f14683a3f>

**静的型付けについて**

KotlinソースコードはJavaバイトコードに変換（コンパイル）されます。コンパイラは、コンパイル時にソースコートの誤りを発見すると、Javaバイトコードを生成しない。そのため、プログラマはバグを早い段階で発見でき、安全あプログラムを作ることができる。

**オブジェクト指向について**

Kotlinのクラスベースはオブジェクト指向言語である。Javaのように、定義されたクラスからインスタンスを生成することができる。Javaとは違って、Kotlinにはプリミティブ型（intやcharなど）はなく、すべてがオブジェクトであり、一貫した扱い方が可能！またプロパティやオブジェクト宣言、拡張関数など、Javaにはない便利な機能がある。これは実際にプログラムを書いて体験してみる。

**オブジェクト指向とは**

・モノとして考える。

・現実世界と一緒

・オブジェクト指向は「概念」だ！概念と言う言葉自体難しいが物事や対象を丸ごとひっくる見たときの大まかな理解のこと

・オブジェクト指向は対象を操作するイメージ

・オブジェクト指向プログラミング（Object Oriented Programming: OOP）とは、プログラムを手順ではなくて、モノの作成と操作として考える。オブジェクトとは「モノ」と言う意味

テレビはリモコンで操作するというイメージ

・大変な作業をなくすことができる。

・大人数で開発するときに便利である。モノを用意して、それを他の人が触れないようにしておけば、他の人がプログラムを壊してしまう心配がなくなる。

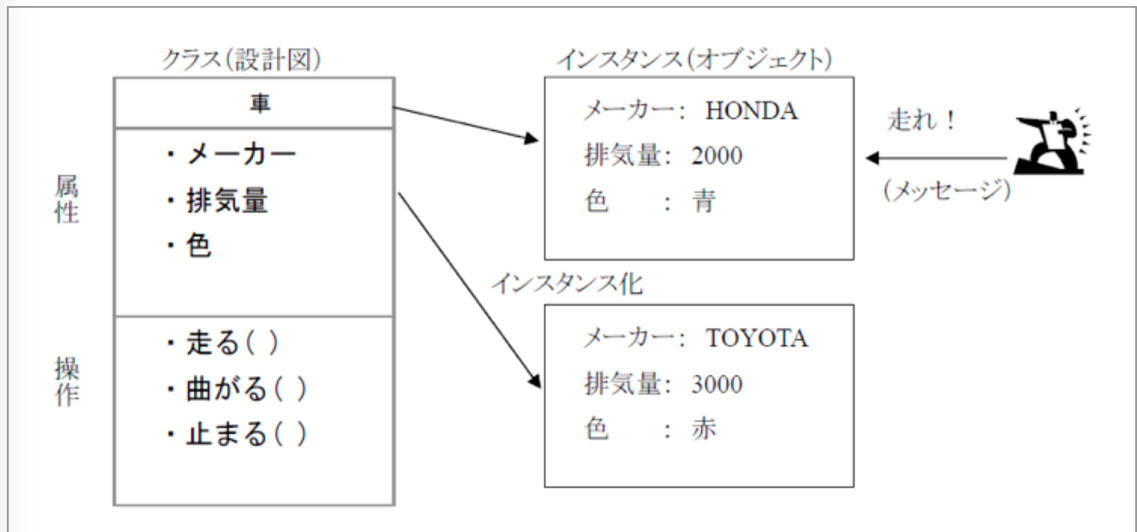
・同じようなものを作りやすい。

**オブジェクト指向の基本用語**

**オブジェクト（Object）**

**オブジェクトは、オブジェクト指向の根本**

**オブジェクトとは、「対象」「物」という意味で、プログラミングにおいてはデータと処理の集まりを意味している。オブジェクト指向で現実のものを例えるとPCやスマホもオブジェクトである。**

****

**クラス（Class）**

**クラスとはオブジェクトの設計書のようなもの。オブジェクトの中のプロパティやメソッドをひとまとめにしたもの。**

**プロパティ（Property）**

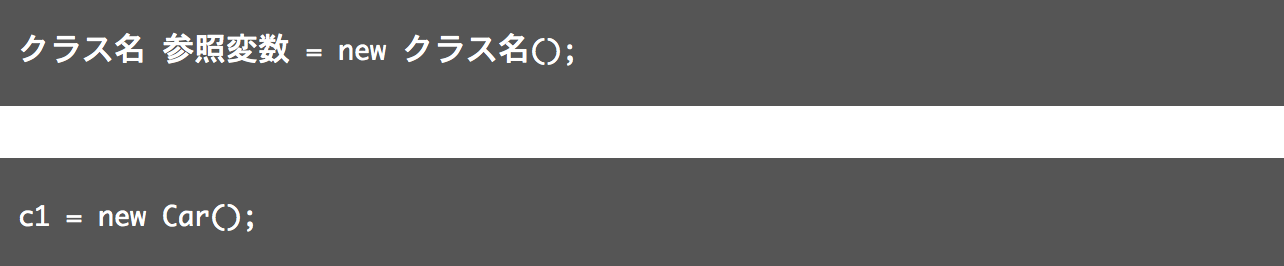
**オブジェクトが持っているデータのことをプロパティ（属性）という。車の例えだと、車というオブジェクトは「メーカー」「排気量」「色」といったプロパティを持っていると言える。**

**メソッド（method）**

**メソッド（操作）とは、オブジェクトが持っている処理のことで、9R魔の例だと「走る」「曲がる」など、オブジェクトから何かしらのアクションを起こす処理のこと。**

**インスタンス化（instance）**

**インスタンスとは「実体」「事例」という意味で、プログラムでオブジェクトを実際に生み出されるものである。設計図からオブジェクトを作ることをインスタンス化と呼ぶ。上がJava、下がKotlin**

****

****

**カプセル化**

**オブジェクトが持つデータや処理のうち、別のオブジェクトから直接利用される必要のないものをいう。利用する場合は外部から操作するために作られた処理を設けることをいう。**

**継承**

**特定のオブジェクトの機能を引き継いで使うことを継承という。例、車からトラックをいう。**

**ポリモーフィズム  
クラスによって同一のメソッドで異なる処理が行えるという性質。**

コンストラクタ

**コンストラクタとは、クラスからオブジェクトを作成した際に、自動的に実行されるメソッドのことで、メンバ変数の初期化などの主に行う。**

**オブジェクト指向の学び方**

**既存のコードを改善しながらオブジェクト指向設計を学ぶ**

**やや極端なコーディング規則を使って、オブジェクト指向らしい設計を体で覚える。**

**どうすればいいのか？**

**実際のコードで設計の違いを知る。・・・やりにくいコードを小さなメソッドや小さなクラスにまとめてみる。重複したコードをメソッドに抽出する。重複した箇所を、抽出したメソッドを呼び出すように書き換える。抽出したメソッドをどちらかのクラスだけに置き、他のクラスからそのメソッドを呼び出す。**

**抽出したメソッドを置くために新しいクラスを作成する。**

**関数型プログラミングについて**

Kotlinには第1級オブジェクトとしての関数がある。つまり、関数を数値や文字列など他の値のように関数の引数として渡したり、戻り値として受け取ったりすることが可能である。これによって、より粒度の小さい単位で再利用が可能になり、抽象的なプログラミングが可能になる。

このことは、Kotlinの簡潔さを実現している仕組みの一つ。これは関数型プログラミングの一要素に過ぎないから、Kotlinは関数型言語ではない。

**実際にどこで使われているか？**

国内だとサイバーエージェント社の映像視聴Androidアプリ「FRESH!by AbemaTV」やSansan社の名刺アプリ「Eight」のAndroid版などはKotlinで開発されている。