****

*Olá, tudo bem com vocês?*

*Espero que estejam bem, muitos aqui já devem me conhecer, mas para aqueles que não me conhecem, eu sou a Ana!*

*Com um imenso prazer, eu compartilharei TODOS os meus resumos, anotações e dicas baseados nas aulas do Bootcamp everis Kotlin Developer / Docs / Comunidades / Artigos / Vídeos / Plataformas de estudos / outros, para agregar o conhecimento de todos, inclusive o meu com o Feedback de vocês!*

*Cabe ressaltar que, assim como muitos, iniciei meus estudos de programação este ano e sei o quão é difícil se orientar nesse novo “mundão” de Dev’s. Pensando nisso, quero ajudá-los a se direcionarem no Bootcamp e nesta incrível e desafiadora jornada!*

***Obs.: Disponibilizarei os resumos em doses fracionadas.***

*Espero que vocês gostem!*

*Ass.: Ana Guerra*

**Introdução ao tema:**

É extremamente necessário conhecer a área da tecnologia que você deseja se aprofundar. Neste caso, é importante que aos poucos você aprimore o seu conhecimento e este documento diz respeito a base principal para iniciar a sua jornada.

**Como baixar o que você precisa:**

Inicialmente, caso se interesse por fazer Download da ferramenta (IDE) utilizada pela instrutora neste módulo, siga o seguinte vídeo passo a passo:

Intellij IDEA e Java: <https://www.youtube.com/watch?v=Ag0uQDMTxMI>

**Como participar da aula sem baixar essa ferramenta?**

Agora não é necessário você ter o Intellij IDEA em sua máquina. Entretanto, ao decorrer do Bootcamp será extremamente necessário fazer o download do tópico anterior. Por enquanto, use o Kotlin por aqui:

Kotlin online: <https://play.kotlinlang.org/>

**Por que baixar uma IDE e o que é isso?**

Para ter muito mais conhecimento do tema e do módulo do mesmo Bootcamp, “ Lógica Essencial da Programação”, acesse o meu artigo na plataforma da Digital Innovation One:

<https://web.digitalinnovation.one/articles/vem-ver-a-base-da-programacao?back=%2Farticles&page=1&order=oldest>

**O que você aprenderá no módulo *Introdução ao Kotlin* ?**

1. Principais conceitos sobre a linguagem Kotlin e Desenvolvimento Android;

. O que aprenderemos nesta aula?

. Sistemas Operacionais (S.O.) que “rodam” no Android Studio;

. Por que começar?

. Curiosidades;

. Desenvolvimento Android;

. Introdução ao Kotlin;

. Estrutura da Linguagem Kotlin;

. Aplicabilidade;

. Vantagens do Kotlin: “Menos é mais”;

. Mapa de movimentação do Kotlin;

. O que aconteceu com o Java?

2. A sintaxe básica do Kotlin: “Olá, Mundo!”

. Tipos de dados presentes no Kotlin;

. Tradução para português dos dados presentes no Kotlin;

. Tipos de variáveis (Números);

. Tipos de variáveis (Outros);

. Entenda Arrays;

. Como criar um Array?

3. Como declarar uma variável?

4. Valor nulo (null) e Operadores Aritméticos;

. Nullability;

. Operadores Aritméticos;

. Operadores Comparativos;

. Operadores Lógicos;

. Operadores in e range;

. Manipulação de Strings;

. Empty e Blank.

5. Introdução a funções;

. Introdução;

. Função de Ordem Superior;

. Função Single-Line;

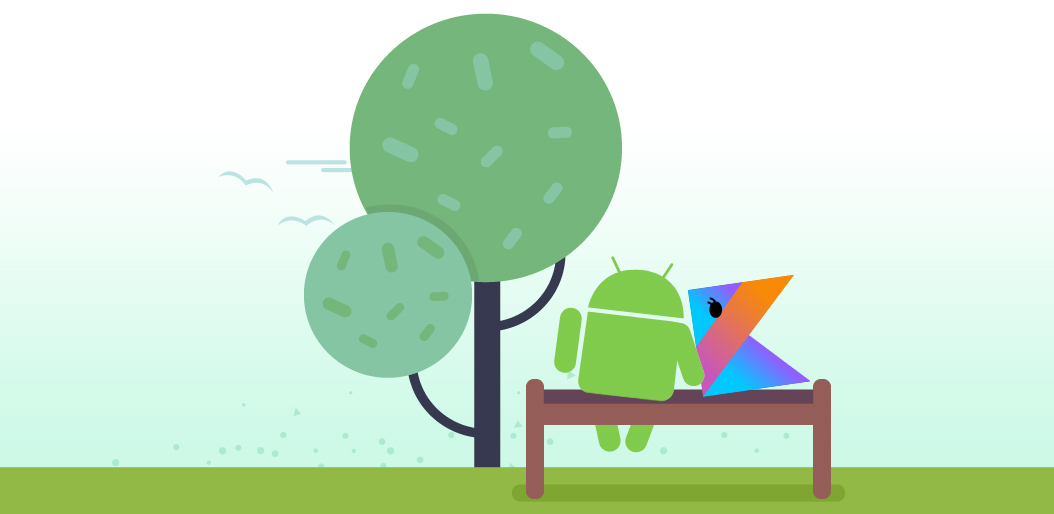
. Funções e Extensões.

6. Estruturas de Controle ou Controle de Fluxo;

. Tipos que devemos conhecer;

. Link de um vídeo no Youtube com a melhor explicação do tema;

. Link do meu repositório do Github para melhor compreensão.

****

**1. Principais conceitos sobre a linguagem Kotlin e Desenvolvimento Android;**

- O que aprenderemos nesta aula?

. Para este tópico é importante seguir a aula da Instrutora, pela DIO;

. Apresentar a linguagem Kotlin e as suas habilidades;

. Apresentar os fundamentos básicos do Desenvolvimento Android;

. Apresentar a sintaxe básica da linguagem Kotlin.

- Sistemas Operacionais (S.O.) que “rodam” no Android Studio e citados em aula;

. Mac OS;

. Windows;

. Linux;

. Chrome OS.

- Por que começar?

. O Android é um Sistema Operacional (S.O.) mantido pelo Google, desde 2005;

. O Google se interessou pelo Android por conta de sua personalização;

. O Android contém mais de 2,5 bilhões de usuários ativos;

. O Android está presente em mais de 24 mil modelos de dispositivos eletrônicos diferentes;

. Última versão disponível é o Android 11, em setembro de 2020;

. A versão Android 12 já foi anunciada, mas não foi liberada para consumo.

- Curiosidades;

. O Android 3.0 (Honeycomb) foi a única versão do Android para Tablets;

. O visual característico do Android surgiu na versão Ice Cream Sandwich;

. Atualmente, mais de 2 milhões de aplicativos são publicados no Google Play;

. No Brasil, 85% dos usuários afirmam que o primeiro Smartphone foi um Android;

- Desenvolvimento Android;

1. Híbrido (Ex.: Kotlin):

. Escreve um código com duas aplicações em outros Sistemas Operacionais;

. Código generalizado e limpo, com acesso em todas as plataformas;

. Difícil desenvolvimento em serviços Background, algumas personalizações e sensores;

. Mais barato e mais rápido;

. Linguagem alinhada ao Framework.

1. Nativo (Ex.: Java):

. Escreve um código que gera apenas uma aplicação específica;

. Possui baixo tempo de resposta.

***Obs.:***

* ***Os TOP 100 aplicativos baixados para os Smartphones foram desenvolvidos parcialmente, ou totalmente, na linguagem Kotlin;***
* ***Aprender a ler a Documentação do Android, segue o Link:***

[***https://developer.android.com/docs***](https://developer.android.com/docs)



- Introdução ao Kotlin;

. A sigla API: Application Programing Interface;

. Em 2011, a JetBrains criou o Kotlin;

. O Google é um dos maiores colaboradores do Kotlin;

. Google + Kotlin = Kotlin first;

. Kotlin é gratuito e possui o seu código disponível para análise (aberto);

. O nome Kotlin é uma homenagem a uma ilha Russa, chamada Kotlin.

- Estrutura da Linguagem Kotlin;

. Diferente da linguagem de programação Java, o Kotlin não utiliza o ponto e vírgula para separar parâmetros, por exemplo;

. Não possui operador ternário tradicional. Ou seja, o controle de fluxo if que você verá se repete apenas 2 vezes, no máximo;

. O Kotlin é famoso por seguir o “Type Safe e Null Safe” (Código Limpo e seguro).

- Aplicabilidade;

. O Desenvolvimento Android iniciou na linguagem de programação Java (nativo) e se adaptou para a linguagem de programação Kotlin (Híbrido);

. O Android é acessível, Mobile Multiplataforma (KMM). Ou seja, compatível com diversos dispositivos eletrônicos;

. O Kotlin pode ser utilizado em:

1. Web Frontend: Kotlin e Java Script;
2. Frameworks: conjunto de códigos de fácil acesso. Em resumo, framework é um conjunto de ferramentas utilizadas para facilitar a vida dos programadores;
3. Web Backend: target Note-JS, Kotlin e Java Script.

- Vantagens do Kotlin: “Menos é mais”;

. Segurança e estrutura bem elaboradas, menos códigos, mais otimizado e grande relevância no mercado de trabalho;

. Interoperabilidade com outras linguagens. Ou seja, é um aliado de outras linguagens de programação;

. “Smart Casts e Null Safety”: Nulo de erros pela correção automática antes da execução do programa;

. O Android é Kotlin First (Lembra do Google?).

- Mapa de movimentação do Kotlin;

2016: Primeira versão do Kotlin é liberada à comunidade Dev;

2017: Google IO anuncia suporte ao Kotlin, para desenvolvimento Android;

2018: Segunda linguagem preferida pelos desenvolvedores;

2019: Google anuncia que o Kotlin é a linguagem principal para o desenvolvimento Android (Java -> Kotlin).

- O que aconteceu com o Java?

Apesar do Java não ser tanto utilizado quanto o Kotlin, essa linguagem ainda tem muita presença no mercado de tecnologia. É importante lembrar que o Kotlin não tem como objetivo substituir qualquer linguagem de programação, mas sim se unir e trabalhar ao lado dela.

**2. A sintaxe básica do Kotlin: “Olá, Mundo!”**

- Tipos de dados presentes no Kotlin;

. Em kotlin os tipos de dados são semelhantes às outras linguagens;

. É importante ter feito o módulo “Estrutura de dados” para compreender melhor o que será dito a partir deste documento;

. Para agregar o seu conhecimento, antes de continuar, leia esse artigo sobre alguns conceitos básicos que você precisa conhecer:

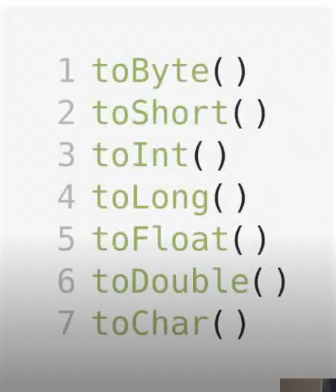
<https://web.digitalinnovation.one/articles/vem-ver-a-base-da-programacao?back=%2Farticles&page=1&order=oldest>

. Os dados são constantes que permitem o acesso aos valores máximos que podem ser colocados no interior;



. A tabela abaixo e acima “servem” para a conexão e a conversão de dados;

. Note que o Intellij IDEA e o Android Studio apontarão os erros;



***Obs.:***

* ***Não fora acrescentado, neste momento, o .random() que serve para a seleção aleatória de números e é importante ter conhecimento dele agora;***
* ***No mesmo caso o .capitalize() que coloca uma String com a inicial maiúscula;***

- Tradução para português dos dados presentes no Kotlin;

. Int. = Número Inteiro (número escrito sem fração);

. Long = Mede distância;

. Float = Flutuar;

. Double = Em dobro;

. Array : Variável (Matriz e vetor);

. Boolean: Possui dois valores (False or True);

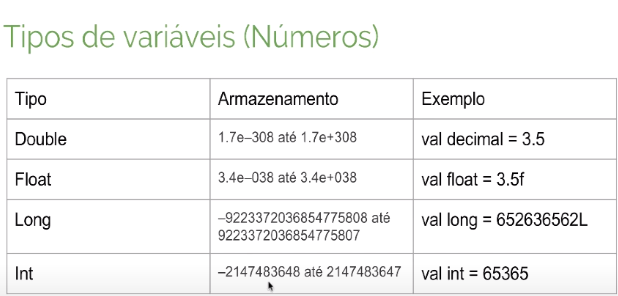
. Char: Caracteres;

. Byte:

. Short: Baixo;

. Null!: Nulo.

- Tipos de variáveis (Números);



. As variáveis Double e Float aproximam os números decimais, a mais usada é o float;

. As variáveis Long e Int servem para dimensionar os números inteiros;

. A estrutura de uma variável é:

*var, val ou const val (CamelCase ou SNAKE\_CASE): Tipo = valor ela*

Ex.:

*var double = 3.5123*

*var double: Double = 3.5123*

*var numero: Double = 3.5123*

*var float = 5.3*

*var float: Float = 5.3f*

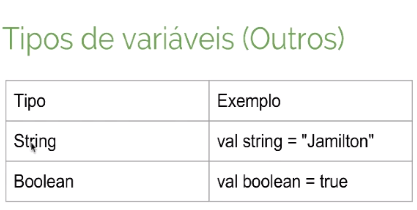
*var numero: Long = 10L*

*var numero: Int = 654*

*var boolean = false*

*var boolean = true*

- Tipos de variáveis (Outros);



. A variável String é usada para nomes;

. A variável Boolean é usada para resultados verdadeiros ou falsos.

- Entenda Arrays;



. Variável armazena apenas um único valor dentro dela;

. Arrays armazenam diversos valores dentro delas;

. Lembre-se, é importante ter conhecimento da programação básica, que já citei um artigo muito importante neste documento, leia:

<https://web.digitalinnovation.one/articles/vem-ver-a-base-da-programacao?back=%2Farticles&page=1&order=oldest>

. Array sempre inicia a contagem dos elementos do zero;

. Entre os colchetes, como mostra a imagem acima, está localizado o índice que indica a posição do elemento dentro da tabela;

. Lembrar que Matriz possui linhas E colunas;

. Lembrar que Vetor possui linhas OU colunas.

- Como criar um Array?

. A seguir citarei algumas imagens de exemplos práticos, em Kotlin;

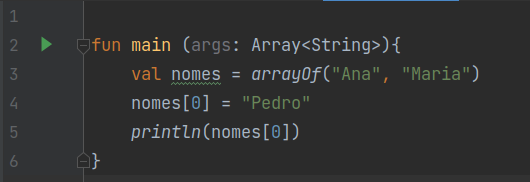
. Preste bastante atenção e tente interpretar cada linha de código para evoluir o seu conhecimento sobre o tema. Posterior a isso, pratique no Kotlin Playground ou Intellij IDEA;

. Lembra que fora dito no início do documento que chegaria uma hora que o Kotlin Playground se tornaria inviável para prosseguir? Então, essa hora está começando e indico a você praticar o que foi lido na IDE.

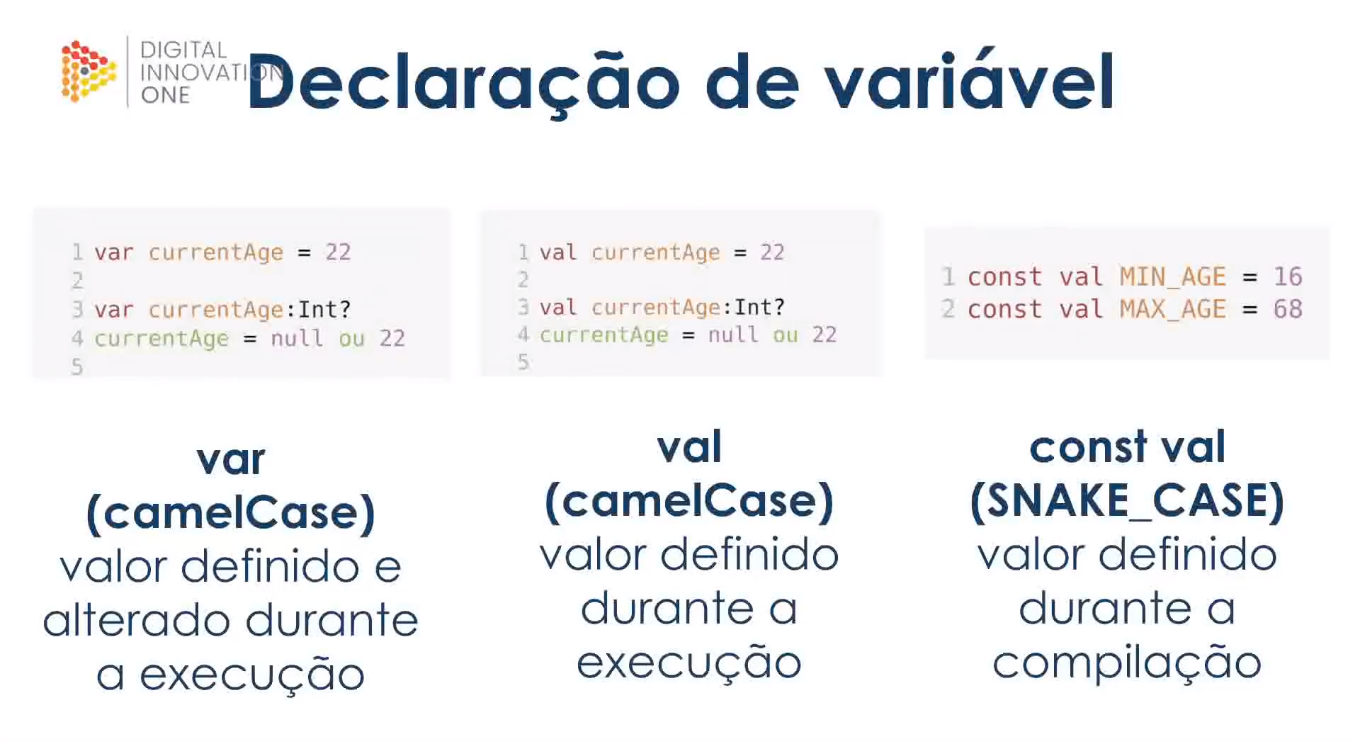
Exemplo 1)



Exemplo 2)



**3. Como declarar uma variável?**

****

. A partir de agora, você compreenderá um pouco melhor do que foi exemplificado acima;

. Uma variável não pode ser declarada sem o computador ter conhecimento do seu tipo;

. Ao declarar um tipo de uma variável, sempre devemos manter o valor com o mesmo segmento de tipo. Por exemplo, se for uma variável com um valor numérico, a próxima variável que o substituirá deverá ser um número também, não podendo ser uma palavra ou caractere;

. Observação, nos próximos tópicos você entenderá melhor o que é uma função.

. Tipos de variáveis e constantes:

1. var (valor mutável, estrutura de escrita camelCase):

. Variável que pode ter o seu valor alterado durante o código;

. Sempre terá a possibilidade de ter o seu valor alterado;

. Estará localizada dentro de uma função, sempre.

1. val (valor imutável, estrutura de escrita camelCase):

. Constante que terá o seu valor atribuído apenas uma vez;

. Estará localizada dentro de uma função, sempre.

1. const val (valor imutável, estrutura de escrita SNAKE\_CASE):

. Utilizado para identificar um objeto, você aprenderá mais sobre ao decorrer do Bootcamp;

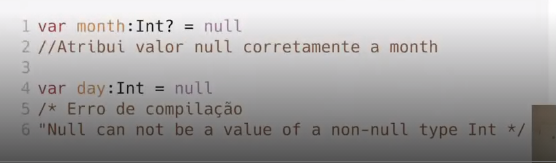
. Constante que terá o seu valor atribuído apenas uma vez;

. Estará localizado fora de uma função, sempre.

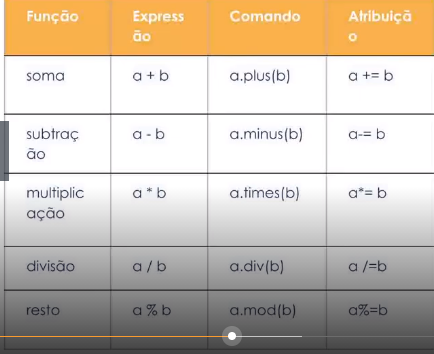
**4. Valor nulo (null) e Operadores Aritméticos;**

- Nullability:

. Qualquer tipo de variável pode ter um valor nulo. Entretanto, deve-se explicitar na declaração da variável com o uso de uma interrogação (?) para identificar a nullability;



- Operadores Aritméticos:



. Este toṕico está diretamente relacionado com os operadores matemáticos básicos, estude eles a parti dos seguintes links para melhor compreender:

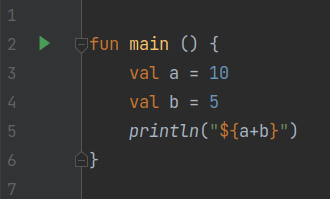
<https://www.youtube.com/watch?v=r_6Ku38sA-c&t=11454s>

. Os operadores podem ser chamados tanto como expressão quanto comando. O resultado sempre será o mesmo;

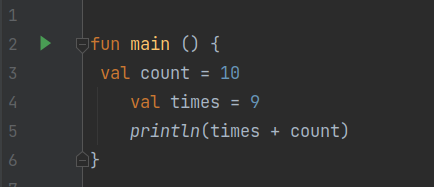
. A função de soma também pode retornar Strings (agrupar);

. Observe o simples exemplo a seguir para notar alguns detalhes:

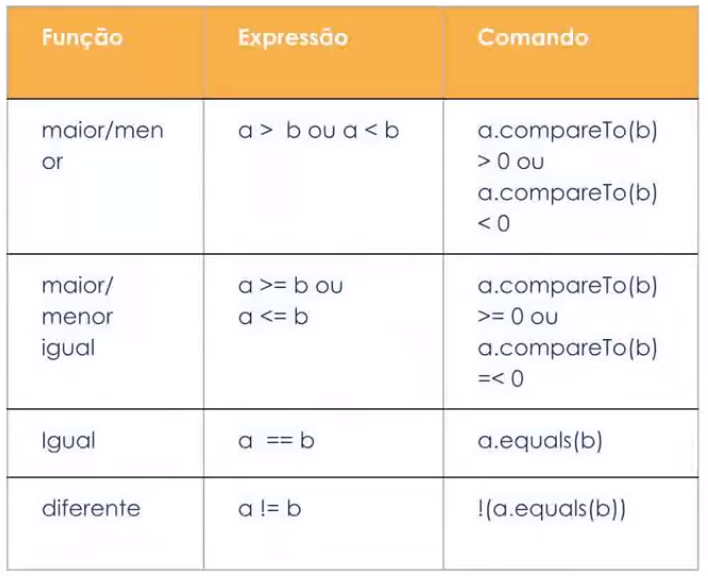
Exemplo 1)



Exemplo 2)



- Operadores Comparativos:



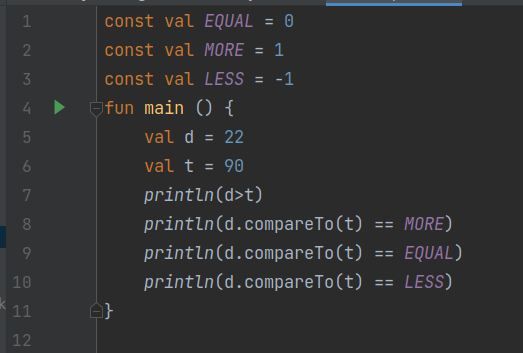
. O comando compareTo retornam valores -1 (menor que), o (igual) ou 1 (maior) e o retorno é sempre um Boolean;

. Lembrar que o operador Boolean retorna true ou false;

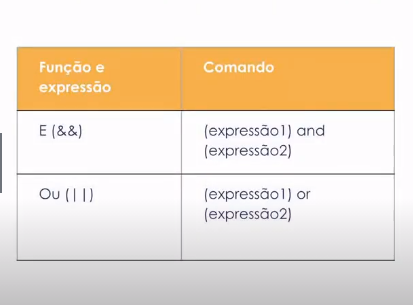
. O campo equals retorna um Booleano.

. Repare o seguinte exemplo que utiliza um pouco do que fora dado anteriormente;

Exemplo 1)



- Operadores Lógicos:



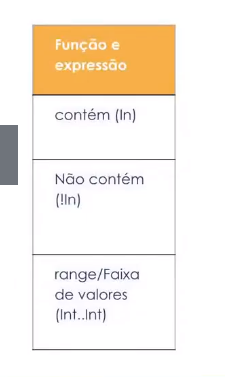
. Quando se utiliza o comando lógico, é recomendado colocar a expressão entre parênteses. A intenção desta pontuação é deixar o código mais claro;

. Neste caso, o Operador Lógico retorna um Boolean;

. Para exemplos, nada melhor do que ver a prática passo a passo. Para isso, veja o seguinte vídeo do Youtube:

<https://www.youtube.com/watch?v=r_6Ku38sA-c&t=11454s>

- Operadores in e range:



. Este comando identifica se o valor está presente em uma lista ou uma faixa (range) de valores;

. range cria um intervalo de valores que inicia em um primeiro valor e termina no último indicado;

. Dois pontos significa que o valor desejado está entre a condição imposta;

. Para melhor interpretação veja o seguinte exemplo:



- Manipulação de Strings:

. Define uma melhor comunicação com o usuário;

. As Strings possuem diversos métodos associados, como:

*Indexação, concatenação, comparação e formatação;*

. Concatenação pode ser identificada como um operador soma;

- Indexação:

. É uma String em forma de array (Matriz ou Vetor);

. First() indica a posição inicial de um número, por exemplo;

. last() indica a posição final de um número por exemplo;

. String.length indica a quantidade de elementos;

. String[index].

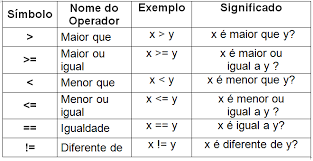
- Concatenação:

. Para concatenar duas Strings usa-se o plus ou o +. Ou seja, devemos, constantemente, revisar os operadores matemáticos conforme prosseguimos o assunto;

. Para concatenar uma variável a uma String, os símbolos $nome ou ${nome} devem ser inseridos conforme for mais relevante.

- Comparação:

. Assim como nos operadores comparativos, temos os valores de x e y como String, neste caso.

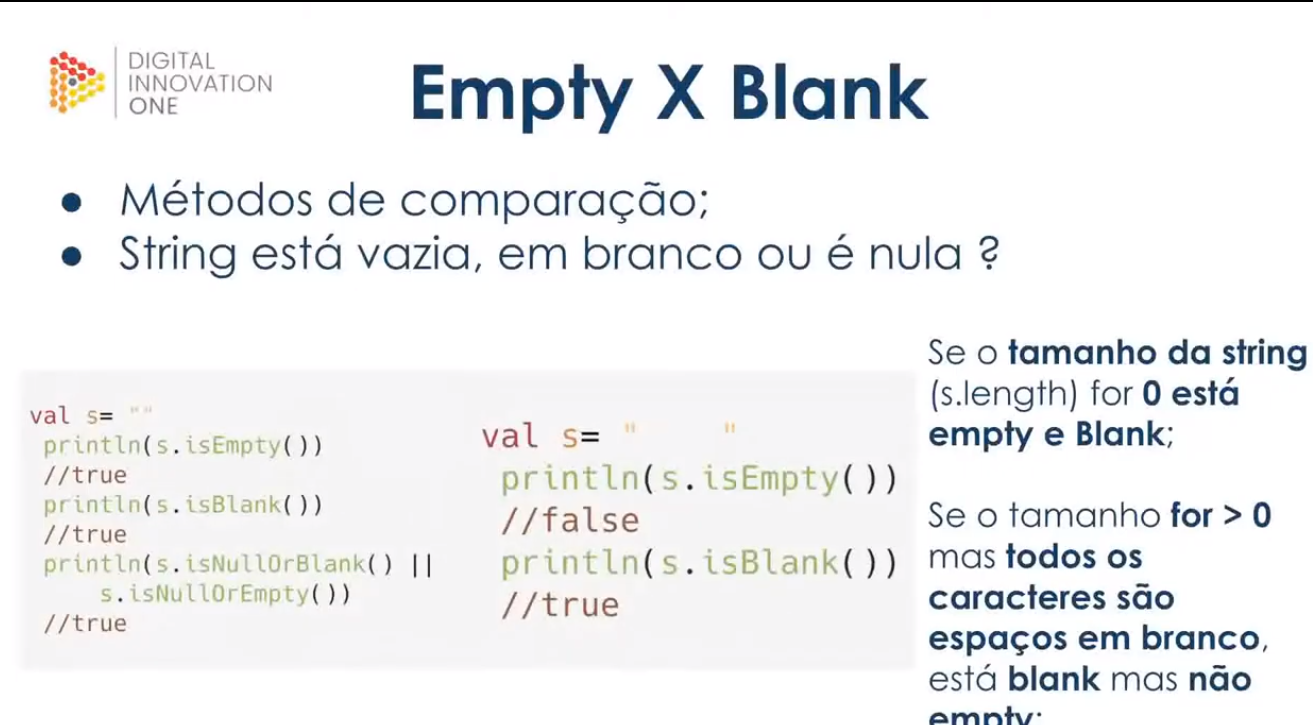


- Formatação:

. Este tópico é importante para manipular as Strings da forma que desejar;



- Empty e Blank:

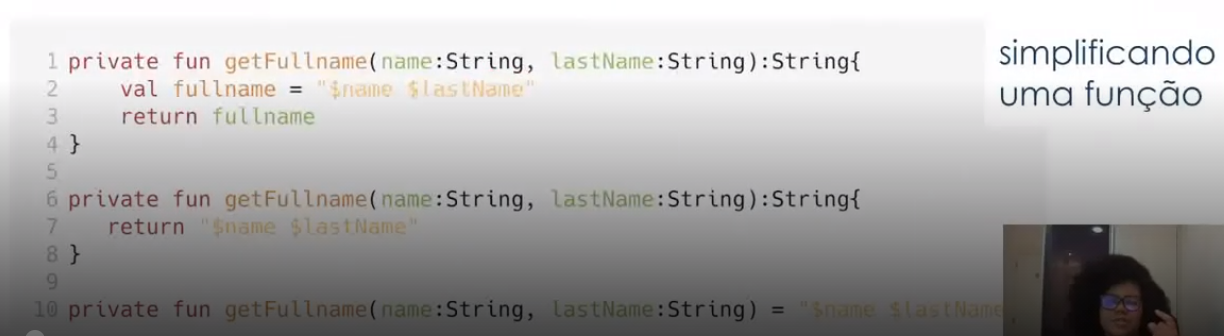


. Neste caso, são utilizados os métodos de comparação;

. Empty equivale ao vazio;

. Blank equivale ao nulo.

**5. Introdução a funções;**

****

- Introdução:

. As funções contêm blocos de códigos que executam uma ação;

. As funções servem para criar uma ação para algo;

. Estrutura de uma função:

fun nomeDaFuncao (nome: Tipo): Retorno{

}

. fun main é a função principal executada dentro de um arquivo;

. Ao longo do curso você compreenderá melhor as funções;

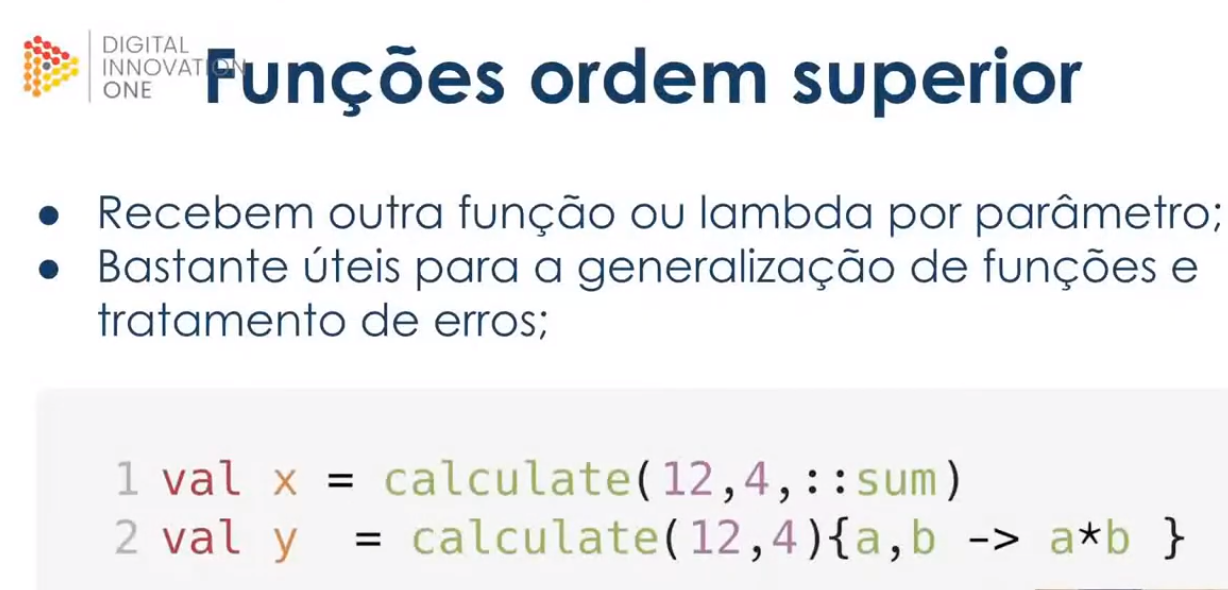
. Função = Método.

- Função de Ordem Superior:

. Recebem outra função ou lambda por parâmetro;

. São bastante úteis para a generalização de funções e tratamento de erros.

Exemplo:



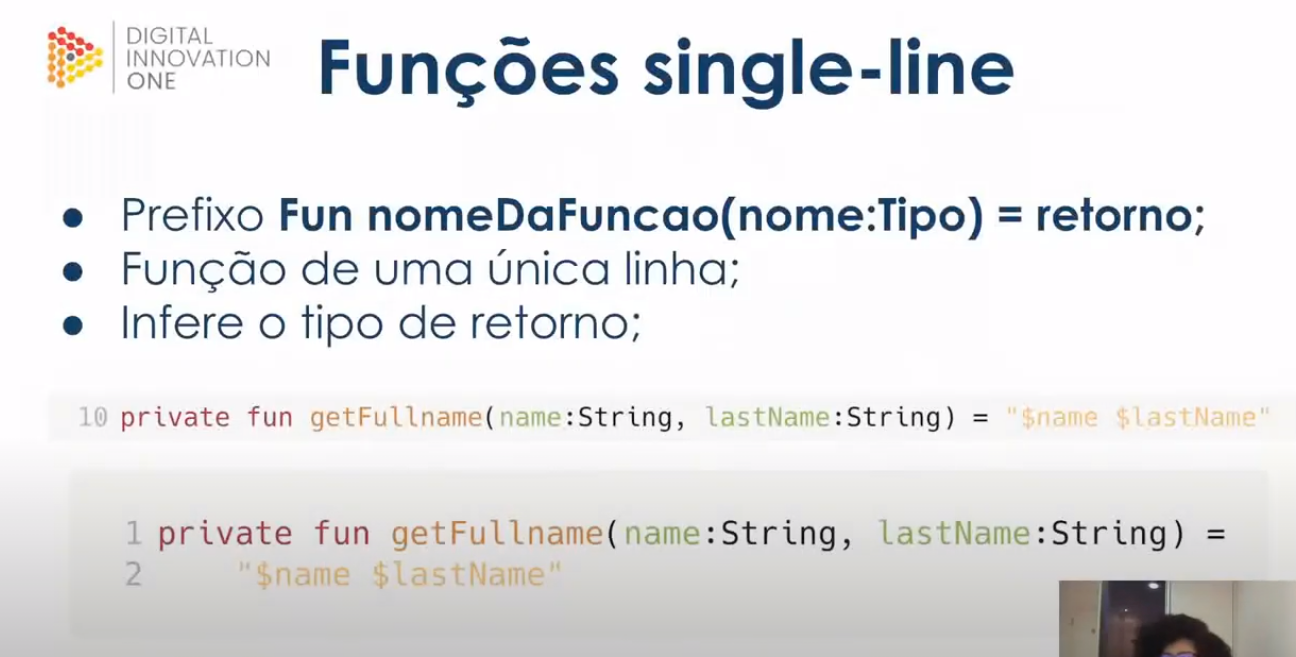
- Função Single-Line:

. Prefixo fun nomeDaFUncao (nome: Tipo) = retorno;

. Função de uma única linha. Ou seja, ótima para simplificar o código;

. Infere um retorno específico.

Exemplo:

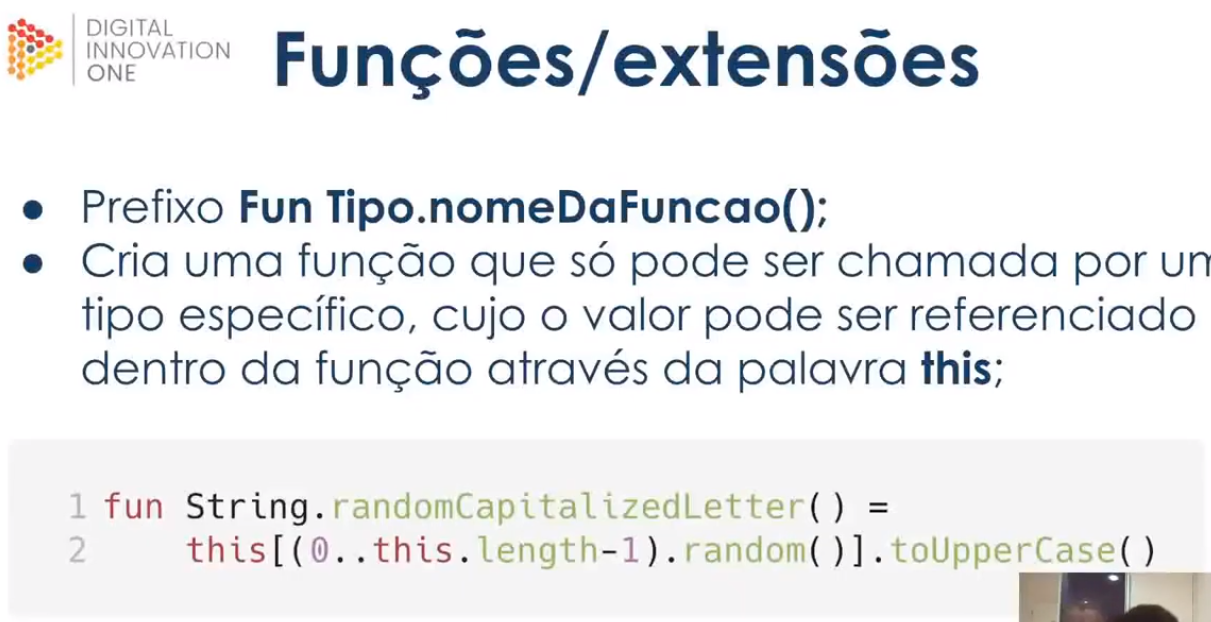


- Funções e Extensões:

. Prefixo: fun tipo.nomeDaFunção();

. Cria uma função que só pode ser chamada para outra função, cujo o valor pode ser referenciado e segue as exigências da função extensão;

Exemplo:



**6. Estruturas de Controle ou Controle de Fluxo;**

. O motivo de eu não ter feito um resumo sobre este tópico é porque ele é extremamente importante para prosseguir o Bootcamp. Por isso, vocês devem procurar não só um, mas VÁRIOS artigos / vídeos / exercícios e meios que acrescentem o conhecimento de vocês;

. É importante ressaltar que, no Bootcamp, temos um módulo que fala muito melhor sobre as estruturas de controle, vá assistí-lo antes de prosseguir os outros módulos:

*“Padrões e convenções profissionais de codificação em Kotlin”*

*(É o penúltimo módulo do Bootcamp everis Kotlin Developer)*

. Obs.: Não prossiga o Bootcamp se não entender este tópico.

- Tipos que devemos conhecer:

1. if/else:
2. when:
3. elvis operator;
4. While;
5. do While;
6. for;
7. forEach;
8. in;
9. range;
10. until;
11. downTo;
12. step.

- Link de um vídeo no Youtube com a melhor explicação do tema:

Este é o link principal que eu assisti quando comecei o Bootcamp, ele me ajudou a ter uma ideia do que eu ia enfrentar ao decorrer dos módulos. Embora eu não tenha compreendido ele 100% no início, ele me ajudou a “montar o terreno” em minha mente para que no final eu chegasse no conhecimento necessário e suficiente para prosseguir os estudos sozinha, juntamente com o Bootcamp everis, claro!

<https://www.youtube.com/watch?v=r_6Ku38sA-c&t=11454s>

- Link do meu repositório do Github para melhor compreensão:

Ao decorrer dos meus estudos eu fiz diversas anotações, como este documento, para solidificar o meu conhecimento e compartilhar com a comunidade DIO o que é importante em cada módulo do Bootcamp. Por conta disso, anotei muitas informações nos seguintes repositórios do Github:

<https://github.com/GuerraAna/Theoretical.Basis.Kotlin>

<https://github.com/GuerraAna/KotlinCollections>