AfroDev

Java Básico - Aula 1 - Parte 1

Cronograma

Introdução

- O que é JAVA?
  - Linguagem de programação criada pela SUN e atualmente mantida pela ORACLE
- Onde é usado?
  - SE: desktop e backend de aplicações
  - ME: dispositivos móveis
  - TV: Televisão Digital
  - FX: Internet rica
  - EE: aplicações enterprise (web e toda sua complexidade)
- Vantagens
- Desvantagens

Vantagens

- Freeware
- Write once, run everywhere
- Comunidade ampla
- Frameworks
- VM que roda mais de uma linguagem
- Retrocompatibilidade

~~Desvantagens~~ (ou características)

- Evolução da linguagem em si é “lenta”
- É uma linguagem de produção, ou seja, nenhuma feature experimental
- Código muito verboso
- Compilação tende a ser mais lenta

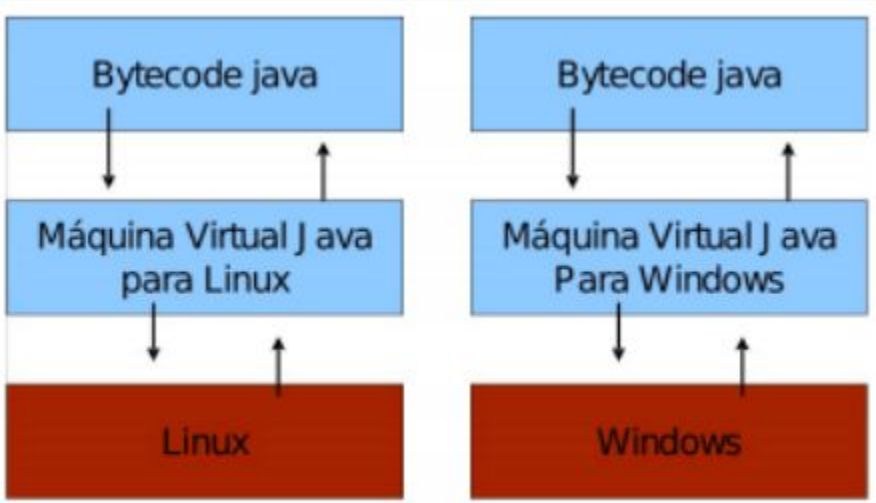
JVM

Máquina virtual onde roda bytecode JAVA

Compilador JAVA gera bytecode universal. Cada SO tem sua JVM que interpreta o mesmo bytecode de maneira diferente.

A VM traduz o bytecode para o SO em uso

Programador não se preocupa com onde o sistema irá rodar



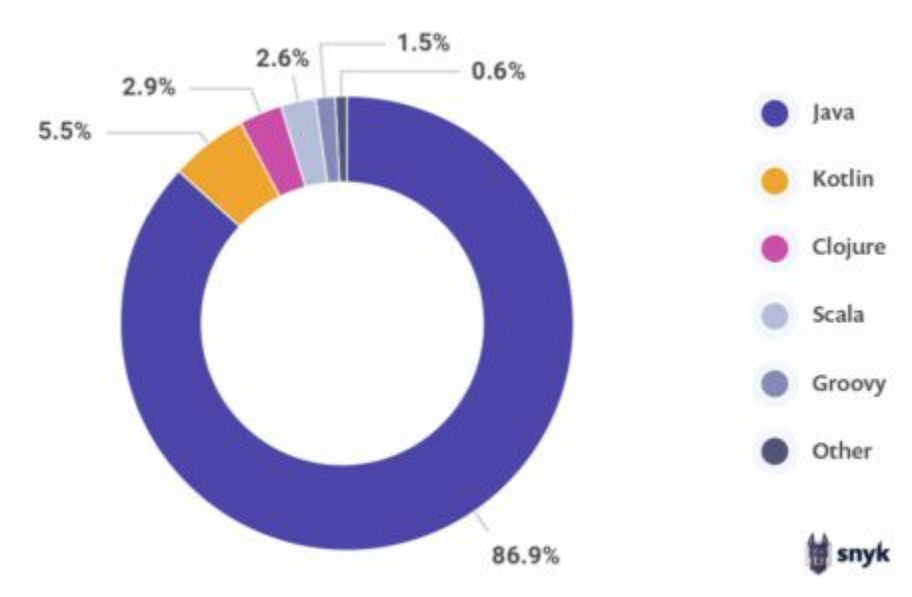
JVM

JVM não roda só JAVA

Scala, Groovy, Kotlin e Clojure são exemplos

Qual linguagem escolher?

Cada problema tem mais de solução, mas sempre tem uma solução mais adequada ou menos pior



Tipos primitivos

JAVA é uma linguagem estaticamente tipada, ou seja, variáveis precisam ser declaradas ANTES de ser usadas.

Cada variável precisa de: TIPO e NOME

int idade = 32;

Somente letras minúsculas e são palavras reservadas

Lógico: boolean

Caracter: char (16)

Byte: byte (8)

Inteiros: short (16) / int (32) / long (64)

Reais: float (32) / double (64)

Tipos primitivos

Tipo	Valores	Inicialização	Exemplo
byte (8 bits)	-128 a 127	0	byte numero = 50;
short (16 bits)	-32,768 a 32,767	0	short numero = 32,000;
int (32 bits)	-2^31 a 2^31-1	0	int numero = 2,147,483,647;
long (64 bits)	-2^63 a 2^63-1	0L	long num = 9,223,372,036,854,775,807;
float (32 bits)	(até 7 dígitos)	0.0f	float distancia = 91.03;
double (64 bits)	(até 15 dígitos)	0.0d	double altura = 1.85;
boolean (1 bit)	true ou false	false	boolean ligado = true;
char (16 bits Unicode)	'\u0000' (0) a '\uffff' (65,535)	'\u0000' (0)	char letra = ‘J’;

Tipos primitivos - String

String não é tipo primitivo, mas tem um suporte para esse tipo (Objeto para cadeira de caracteres)

Aspas duplas indicam que os caracteres são uma string na verdade

Exemplo: String nome = “Maria”;

Strings são imutáveis. Depois que o valor é atribuído ele não é mais alterado

Declaração de variáveis

Precisa de um tipo e um nome e precisa ser inicializada ANTES da sua leitura:

```
int numero;

numero = 1;

System.out.println(numero);
```

Pode ser uma referência

```
String nome = new String(“Maria”);
```

Pode ser um wrapper

```
char letra = ‘c’;

Character c = new Character(letra);
```

Declaração de variáveis - Nomes

São case sensitive: casa, cAsa, caSa, etc são variáveis diferentes

Começar com letra, $ ou _. Exemplos: a, $a, _a

Evite usar _ e nunca use $

Espaços em branco não são permitidos. Exemplo: int a casa; //erro

Use nomes completos e não abreviaturas. Exemplo: int vel; //nok int velocidade; //ok

Se variável tiver apenas 1 nome: todas letras minúsculas. Exemplo: int peso;

Se variável tiver mais de 1 nome: primeiro nome todo minúsculo, segundo nome com a primeira letra maiúscula e demais minúsculas. Repetir esse procedimento para demais nomes (camel case): int anoDeNascimento;

Constantes: todas as letras maiúsculas e a palavras separadas por _. Exemplo: VALOR_VELOCIDADE_DO_SOM

Declaração de campos/atributos

Igual a declaração de variável, porém não precisa ser inicializada. Inicialização default acontece nesse caso

Precisa de um tipo e nome

Dentro de um contexto para descrever um objeto em termos de atributos. Exemplo do carro ao lado

Declaração de métodos

Método é uma operação/serviço que pode ser executada no objeto existente.

Tem acesso aos atributos

Podem ser invocados a partir de uma “mensagem”.

```
public void acelerar(int velocidade) {

...

}
```

Escopo

Limitado pelos caracteres “{“ e “}”. Condicional com somente 1 instrução, não precisa de “{“ e “}”

Podem ser aninhados (visibilidade do mais externo ao mais interno)

Não pode “duplicar” definição de variáveis

```
//a tem visibilidade aqui e no bloco interno

int a = 1;

{

        int b = a + 1; //ok, “a” visível no bloco mais interno

}

System.out.println(b); / /”b” fora do escopo -> erro
```

Escopo

Variáveis têm um ciclo de vida:

1. São criadas
2. Ficam vivas dentro de um escopo
3. Coletor de lixo pode entrar em ação

```
{

Carro c = new Carro();

} //fim do escopo

// aqui a variável “c” não pode ser mais acessada. O que ela consumiu de memória poderá ser reclamado/liberado a partir daqui
```

Garbage Collection

1. Limpeza AUTOMÁTICA de objetos sem referência (sem uso)
2. Retira essa responsabilidade/preocupação do programador
3. Time de especialistas trabalhando nesse algoritmo
4. Processo complexo e pesado, porém “transparente” para o programador

Dicas:

Saber que existe

Saber que existe a possibilidade de configurar o GC

Mexer o menos possível e somente quando necessário

Prática

1. Entrar em: https://github.com/lfmrocha88/basic-java
2. Procurar Aula1Parte1
3. Analisar a classe Carro
    a. quais seu atributos?
    b. quais seus métodos/serviços?
4. Se tiver um Eclipse, tente executar o programa
5. Escreva um método/serviço que altere algum atributo do carro

Desafio!

- Tente executar o programa
- Escreva um método/serviço que altere algum atributo do carro

Revisão

- Aprendemos que JAVA é uma linguagem de produção largamente utilizada pela comunidade e com muitas opções de frameworks
- JVM é a máquina virtual que interpreta o bytecode gerado a partir de um código JAVA
- Vimos os seus tipos primitivos e respectivos tamanhos
- Aprendemos sobre variáveis que representam os atributos de um objeto e métodos que podem alterar esses atributos
- A execução dos códigos em JAVA tem um escopo delimitado e a visibilidade de variáveis precisa respeitar esse escopo
- A limpeza de memória de objetos sem referência ativa pode ocorrer através do GC