Estrutura de Dados e Linguagem de Programação

Java - Ivan Ferreira

Ivan Marcelo Ferreira

- Tecnologia em Análise e Desenvolvimento de Sistemas - Integrado (2010)
- Kanban System Design - Kanban University (2020)
- Programador web desde 2010
- Tech Lead Super Professor

Java

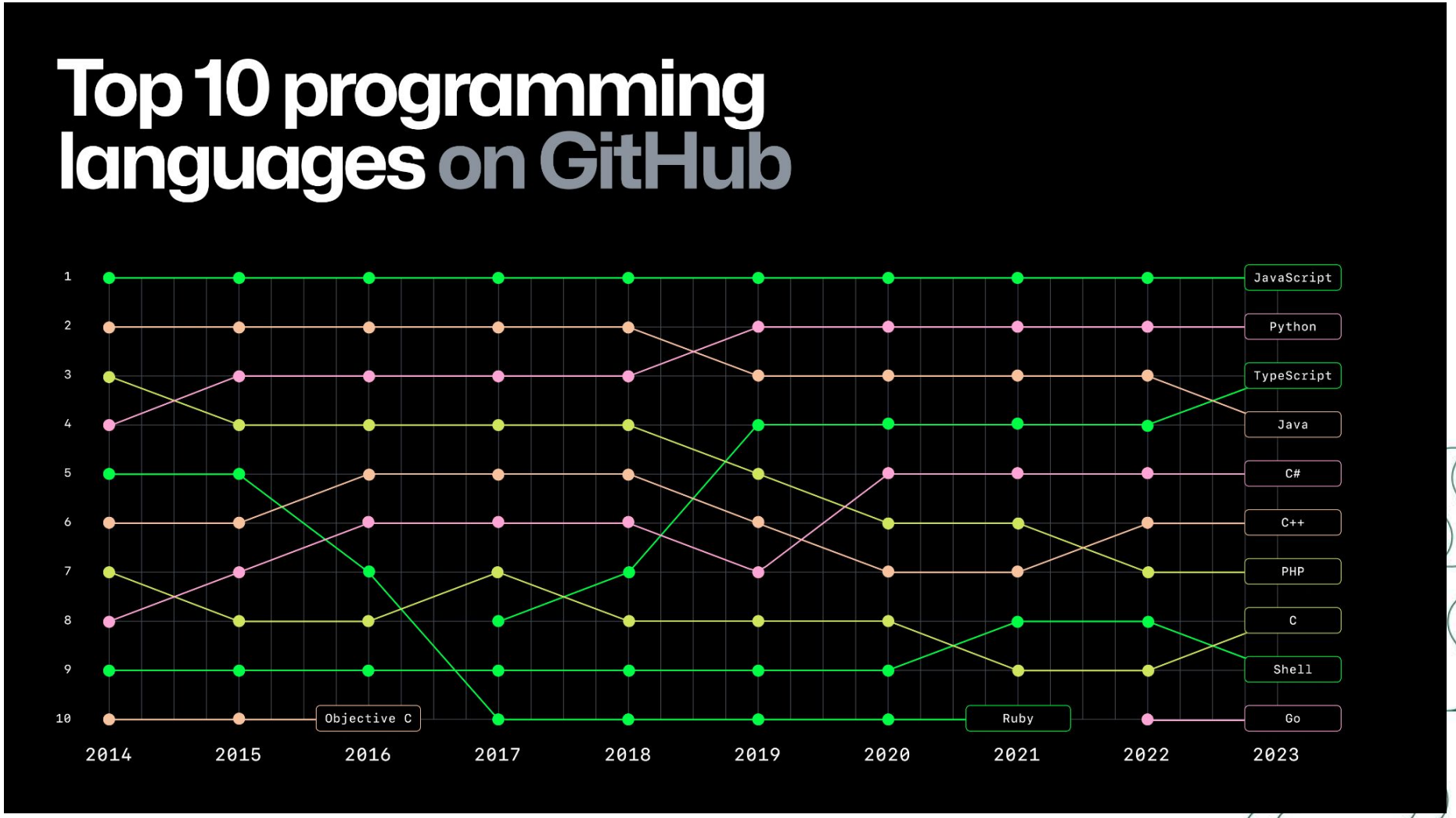
- Criada em 1995 pela Sun Microsystems
- Linguagem criada para ser multiplataforma
- Compilada em bytecode e executada pela JVM
- Utiliza o paradigma da Programação Orientada a Objetos (POO) - *OOP*
- Tipagem forte (estática);
- Em 2004 atingiu a marca de 3 milhões de desenvolvedores em todo mundo
- Em 2008 foi adquirida pela Oracle por US$ 7,4 bilhões;

Dados

- O que são dados?
  - int
  - long
  - byte
  - short
  - float
  - double
  - boolean
  - char

Estrutura de Dados

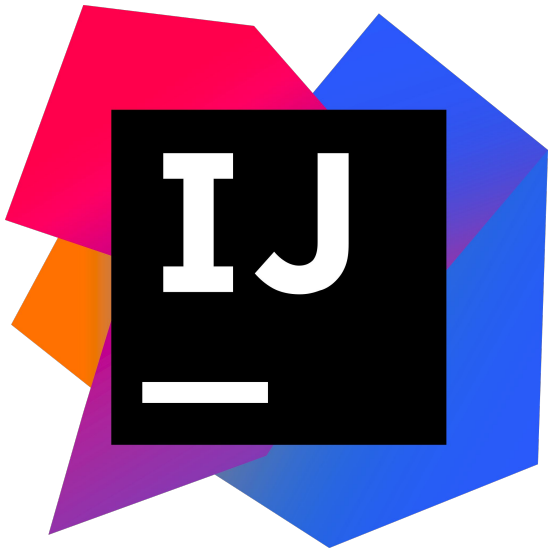
- O que são estruturas de dados?
  - Formas de organização/agrupamento dos dados
- Cada estrutura de dados tem um conjunto de métodos próprios para realizar operações como:
  - Inserir ou excluir elementos;
  - Buscar ou localizar elementos;
  - Ordenar elementos;



Fonte: https://github.blog/2023-11-08-the-state-of-open-source-and-ai/

Live coding

- IntelliJ



Convenções de código

- Case sensitive
- Classes x variáveis/métodos
  - Classe sempre começa em maiúsculo e variável sempre minúsculo
  - sem caracteres especiais (exceto o underscore)
  - não começar com um número

Tipos primitivos

- byte
- short
- int
- long
- boolean
- char
- float
- double

Tipos especiais

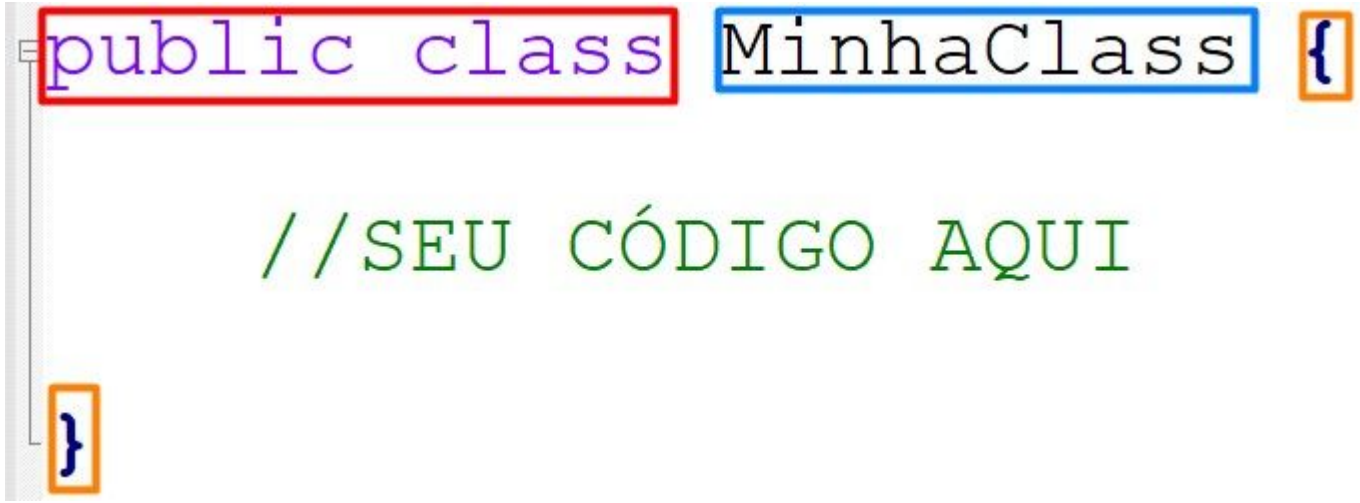
- String;
- Classes Wrapper;
- vetores
- matrizes
- Objetos;*

Tabela 1. Classes `wrapper` e seus construtores.

Primitivo	Classe Wrapper	Argumentos do construtor
boolean	Boolean	boolean ou String
Byte	Byte	byte ou String
Char	Character	char
double	Double	double ou String
Float	Float	float, double ou String
Int	Integer	int ou String
Long	Long	long ou String
short	Short	short ou String

Step back!

Anatomia da classe Java



Anatomia da classe Java

```
public class MinhaClass {

    //CORPO DA CLASSE

    public static void main (String [] args) {

        //CORPO DO MÉTODO main

    }

}
```

Declaração de variáveis

```
<Tipo> <nomeVariavel> <atribuicaoDeValorOpcional>
```

Exemplos:

```
int idade; //Tipo "int", nome "idade", com nenhum valor atribuído.
int anoFabricacao = 2021; //tipo "int", nome "anoFabricacao", com valor 2021.
double salarioMinimo = 2.500; //tipo "double", nome "salarioMinimo", valor 2.500.
```

Constantes

```
final double VALOR_DE_PI = 3.14;
```

Imutáveis;
Por convenção, sempre em letras maiúsculas;

Métodos

```
somar(int n1, int n2){}
findById(int id){} // inglês é MUITO COMUM
```

Toda e qualquer ação/comportamento da aplicação

Escopo

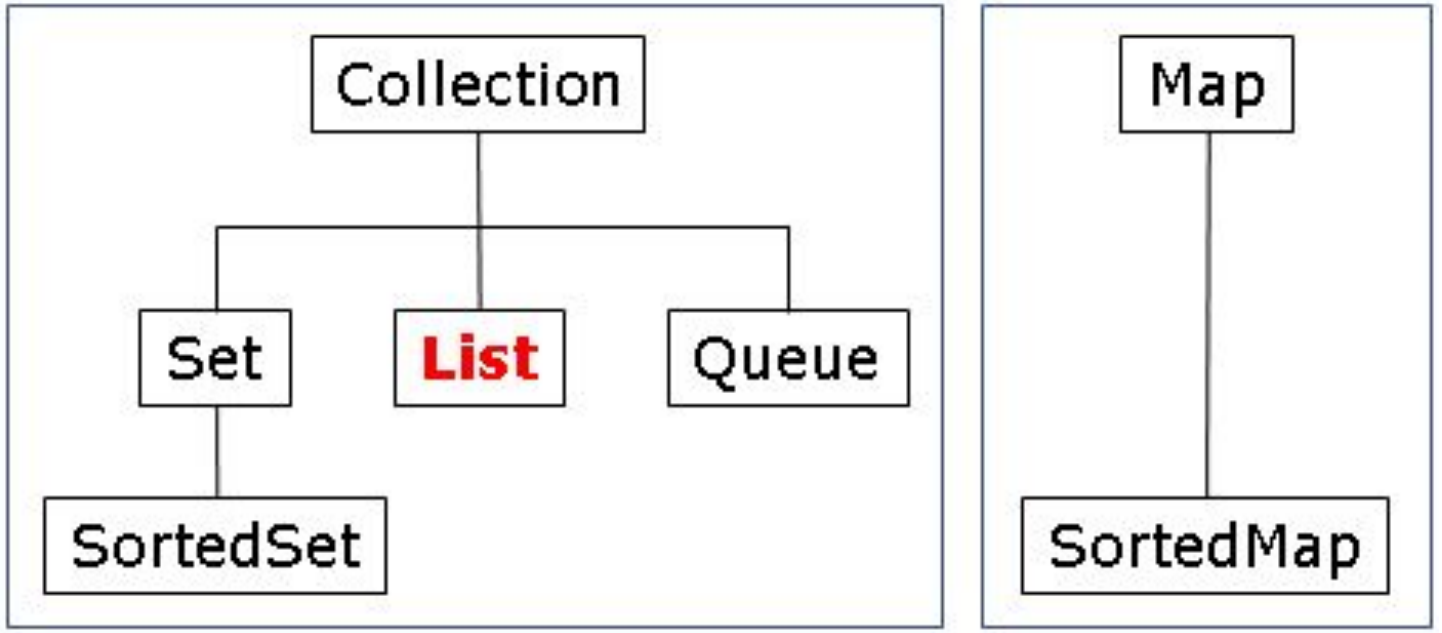
```java
public class Conta {
    //variavel da classe conta
    double saldo=10.0;

    public void sacar(Double valor) {
        //variavel local de método
        double novoSaldo = saldo - valor;
    }
    public void imprimirSaldo(){
        //disponível em toda classe
        System.out.println(saldo);
        //somente o método sacar conhece esta variavel
        System.out.println(novoSaldo);

    }
```

Depende de ONDE a variável foi declarada

Coleções



Coleções

- Lista (List)
- Conjunto (Set)
- Fila (Queue)
- Mapa (Map)

List

```
String [] meuStringArray = new String[3];
```

```
List<String> lista = new ArrayList<>();
lista.add("Pessoa 1");
lista.add("Pessoa 2");
lista.add("Pessoa 3");
```

List (ArrayList)

- new ArrayList<>() // instancia um objeto
- add(item) // adiciona um item
- remove(index) // remove um item pela posição
- indexOf(item) // retorna o índice de um item
- set(index, item) // seta um item em uma posição determinada
- clear() // limpa a lista toda

Datas!!

```
LocalDate today = LocalDate.now();
// o LocalDate contém apenas a data

LocalTime time = LocalTime.now();
// o LocalDateTime traz também o horário
```

Set

```
Set<E> set = new HashSet<E>;


set.add(element1);
set.add(element2);
set.add(element3);
```

Set (HashSet)

- new HashSet<>() // instancia um objeto
- add(item) // adiciona um item
- addAll(collection) // adiciona uma coleção (union)
- contains(item) // verifica se contém um item
- containsAll(collection) // verifica se contém uma seleção
- clear() // limpa o conjunto todo
- remove(item) // remove um item
- removeAll(collection) // remove uma coleção (diferença)
- retainAll(collection) // acha a intersecção

Equals

```java
public class Pessoa {
 private String nome;
 private int idade;

 // Construtor, getters e setters

 @Override
 public boolean equals(Object obj) {
     if (this == obj) return true;
     if (obj == null || getClass() != obj.getClass()) return false;
     Pessoa pessoa = (Pessoa) obj;
     return idade == pessoa.idade && Objects.equals(nome, pessoa.nome);
 }
}
```

Map

```
Map<String,String> example = new HashMap<String,String>();


example.put( "K1", "V1");
example.put( "K2", new String( "V2" ));
example.put( "K3", new String( "V3" ));
```

Map (HashMap)

- put(key, value) // adiciona um item
- get(key) // busca um item pela chave
- remove(key) // remove um item pela chave
- clear() // remove todos os elementos
- size() // retorna o tamanho
- keySet() // retorna todas as chaves
- values() // retorna todos os valores

Queue

```
// Criando uma fila
Queue<String> fila = new LinkedList<>();

// Adicionando elementos na fila
fila.offer("Elemento 1");

// removendo um elemento
fila.poll();
```

Queue (LinkedList)

- offer(element) // adiciona um item na fila
- poll() // retorna e remove o primeiro elemento da fila
- peek() // retorna e mantém o primeiro elemento da fila
- … demais métodos das listas

Stack

```
// Criando uma pilha
Stack<String> pilha = new Stack<>();

// Adicionando elementos na pilha
pilha.push("Elemento 1");
pilha.push("Elemento 2");
pilha.push("Elemento 3");

// Acessando e removendo o elemento do topo da pilha
String topoDaPilha = pilha.pop();
```

Stack

- push(element) // adiciona um item na pilha
- pop() // retorna e remove o primeiro elemento da pilha
- peek() // retorna e mantém o primeiro elemento da pilha
- … demais métodos das listas

Obrigado!

 ivan.ferreira@superprofessor.com.br

 @aiwans

 Ivan Ferreira

Super Professor®