

PARCERIA EXCLUSIVA
FORMAÇÃO COMPLETA

Profissão: Desenvolvedor Full Stack Java

Desenvolva habilidades completas em programação, abrangendo todas as etapas do desenvolvimento de softwares e aplicativos, desde a interface até o sistema e bancos de dados. Domine Java para criar soluções complexas de ponta a ponta, impulsionando suas habilidades como desenvolvedor Full Stack. Esta é uma oportunidade exclusiva para colaboradores da Stefanini.



Palestra

Aprofundando-se em Expressões Lambdas e Streams

Atualizado até o Java 22 &
Eclipse 2024-03

Prof. Antonio B. C. Sampaio Jr



06 de maio – 18h às 19h

REPOSITÓRIO GITHUB



antonio-sampaio-jr / **palestra-stefanini**

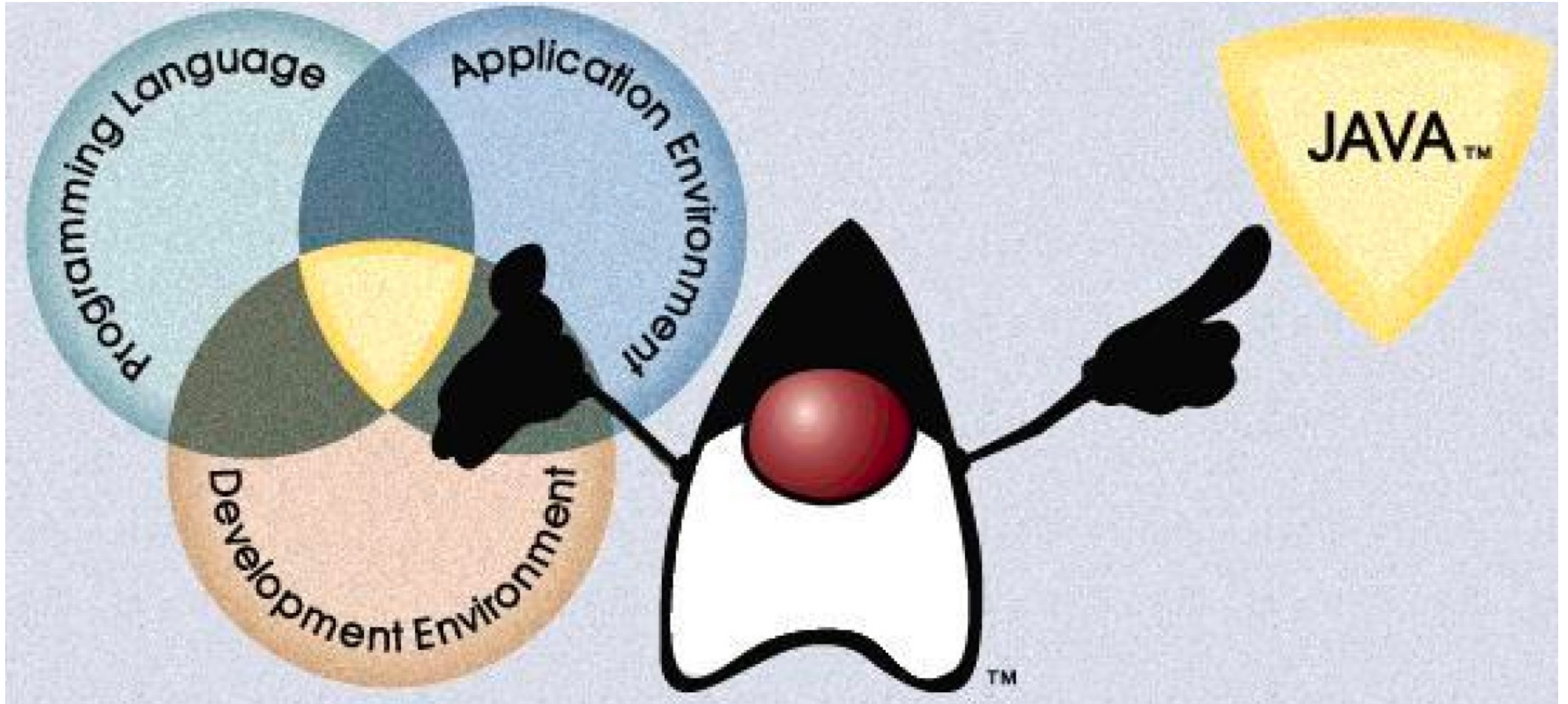
TÓPICOS DA PALESTRA



- CONCEITOS BÁSICOS DE JAVA
- PROGRAMAÇÃO ORIENTADA A OBJETOS
- PROGRAMAÇÃO FUNCIONAL
- EXEMPLO PRÁTICO

CONCEITOS BÁSICOS DO JAVA

O QUE É O JAVA?



O QUE É O JAVA?

- **Definição – O Java é:**
 - Uma linguagem de programação de alto nível, orientada a objetos/funcional e amplamente utilizada na indústria de software.
 - Acompanhada por uma vasta coleção de APIs (*Application Programming Interfaces*), que consistem em classes, componentes e frameworks, facilitando o desenvolvimento de aplicações em diversas plataformas.
 - Suportado por um ambiente de execução (*Java Virtual Machine - JVM*) que permite a execução de código Java em uma variedade de dispositivos e ambientes, incluindo navegadores, mainframes, sistemas operacionais, dispositivos móveis, cartões inteligentes e até mesmo eletrodomésticos inteligentes. Isso confere ao Java a capacidade de ser uma das tecnologias mais versáteis e ubiquamente utilizadas na programação de software.

O QUE É O JAVA?



Top 5 Languages at Fortune 25

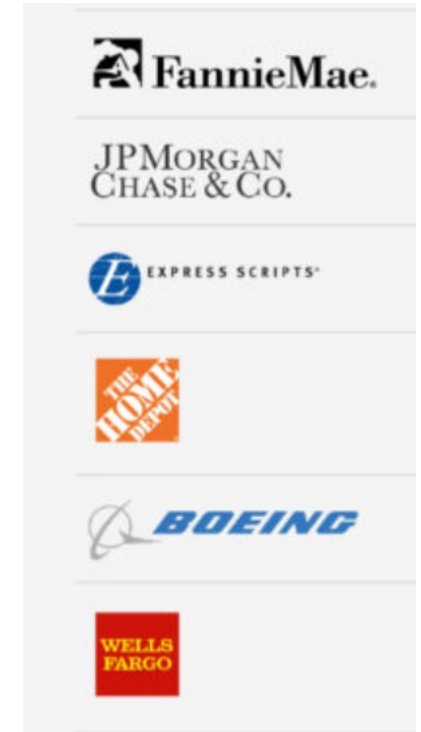
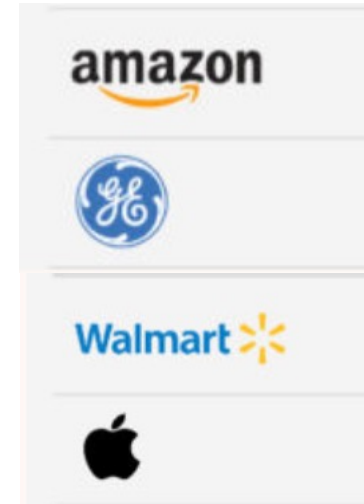
- 1 JavaScript
- 2 Java
- 3 Python
- 4 Ruby
- 5 Perl



of Fortune 500
Companies use Java



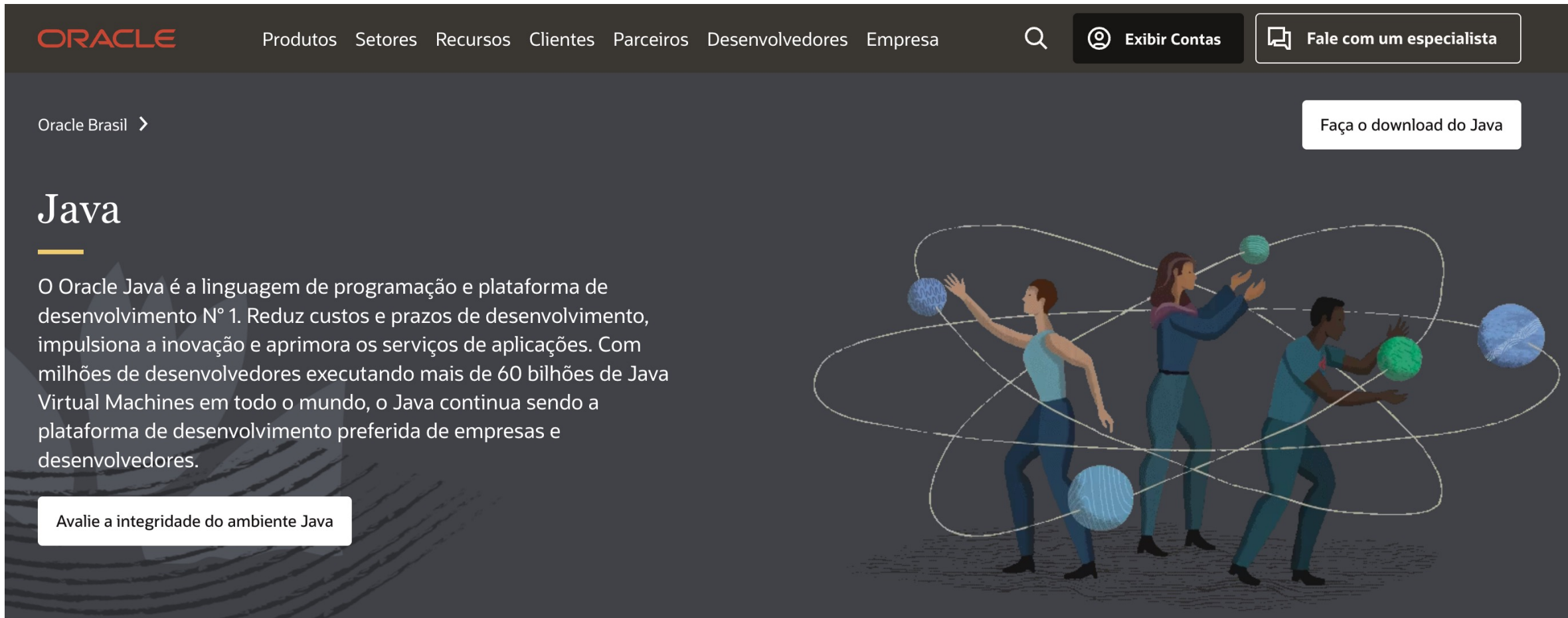
Languages Employed
on Average at
Companies



- Com mais de 60 bilhões de *Java Virtual Machines* em todo o mundo, o Java continua sendo a plataforma de desenvolvimento preferida de empresas e desenvolvedores.

O QUE É O JAVA?

- Site Oficial



The screenshot shows the Oracle Java website homepage. At the top is a dark navigation bar with the Oracle logo on the left and links for 'Produtos', 'Setores', 'Recursos', 'Clientes', 'Parceiros', 'Desenvolvedores', and 'Empresa' in the center. On the right of the navigation bar are a search icon, a 'Exibir Contas' button with a user icon, and a 'Fale com um especialista' button with a speech bubble icon. Below the navigation bar, on the left, is the text 'Oracle Brasil >'. To the right of this is a white button that says 'Faça o download do Java'. Below the navigation bar, the main content area has a dark background. On the left, the word 'Java' is written in a large, white, serif font, underlined with a short yellow line. Below 'Java' is a paragraph of white text: 'O Oracle Java é a linguagem de programação e plataforma de desenvolvimento N° 1. Reduz custos e prazos de desenvolvimento, impulsiona a inovação e aprimora os serviços de aplicações. Com milhões de desenvolvedores executando mais de 60 bilhões de Java Virtual Machines em todo o mundo, o Java continua sendo a plataforma de desenvolvimento preferida de empresas e desenvolvedores.' Below this paragraph is a white button that says 'Avalie a integridade do ambiente Java'. On the right side of the main content area is a large illustration of three people (two men and one woman) in business casual attire, standing and interacting with several glowing blue and green spheres. White, glowing orbital lines connect the spheres, creating a complex, interconnected network that symbolizes the Java ecosystem.

ORACLE

Produtos Setores Recursos Clientes Parceiros Desenvolvedores Empresa

🔍

👤 Exibir Contas

💬 Fale com um especialista

Oracle Brasil >

Faça o download do Java

Java

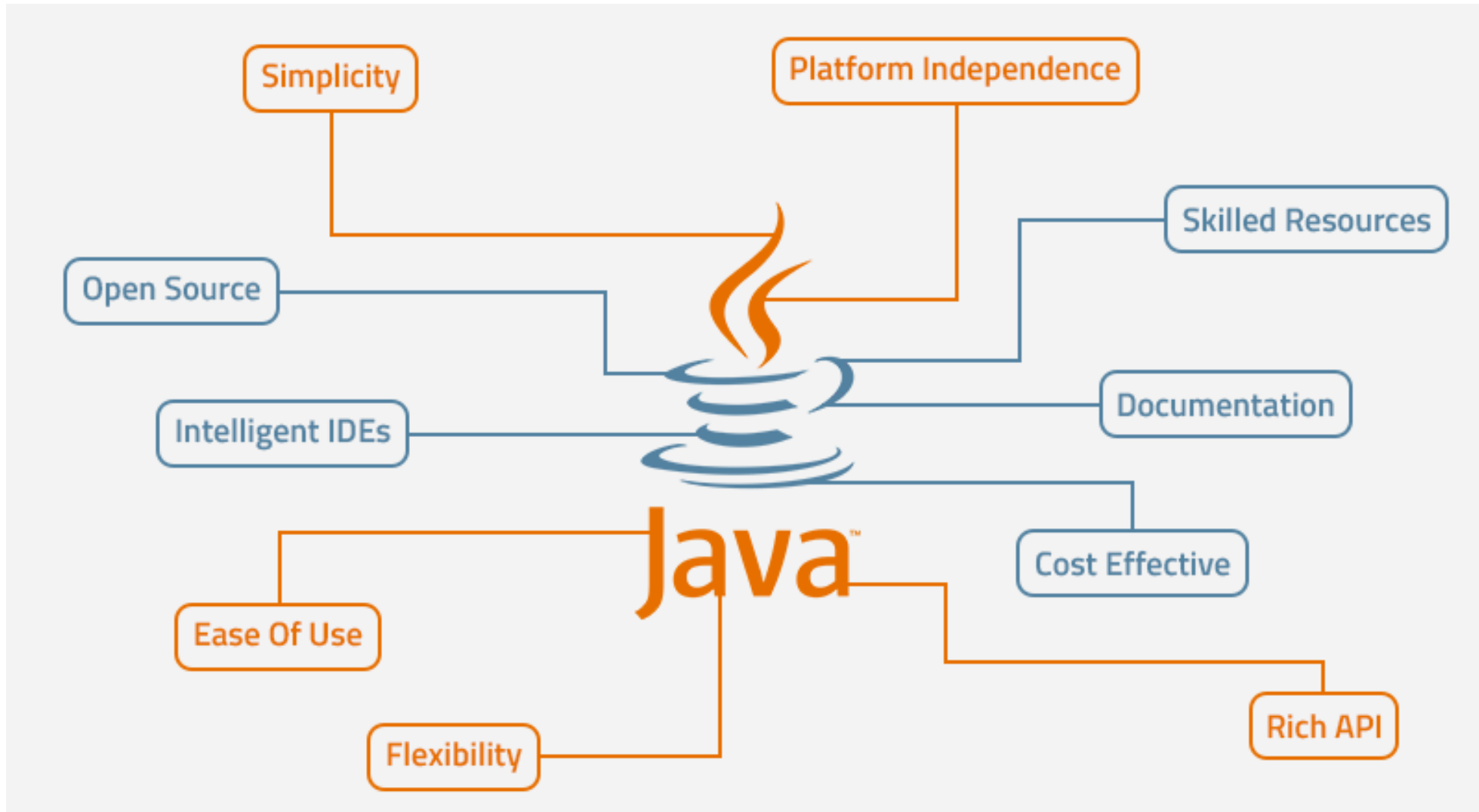
O Oracle Java é a linguagem de programação e plataforma de desenvolvimento N° 1. Reduz custos e prazos de desenvolvimento, impulsiona a inovação e aprimora os serviços de aplicações. Com milhões de desenvolvedores executando mais de 60 bilhões de Java Virtual Machines em todo o mundo, o Java continua sendo a plataforma de desenvolvimento preferida de empresas e desenvolvedores.

Avalie a integridade do ambiente Java

<https://www.oracle.com/br/java/>

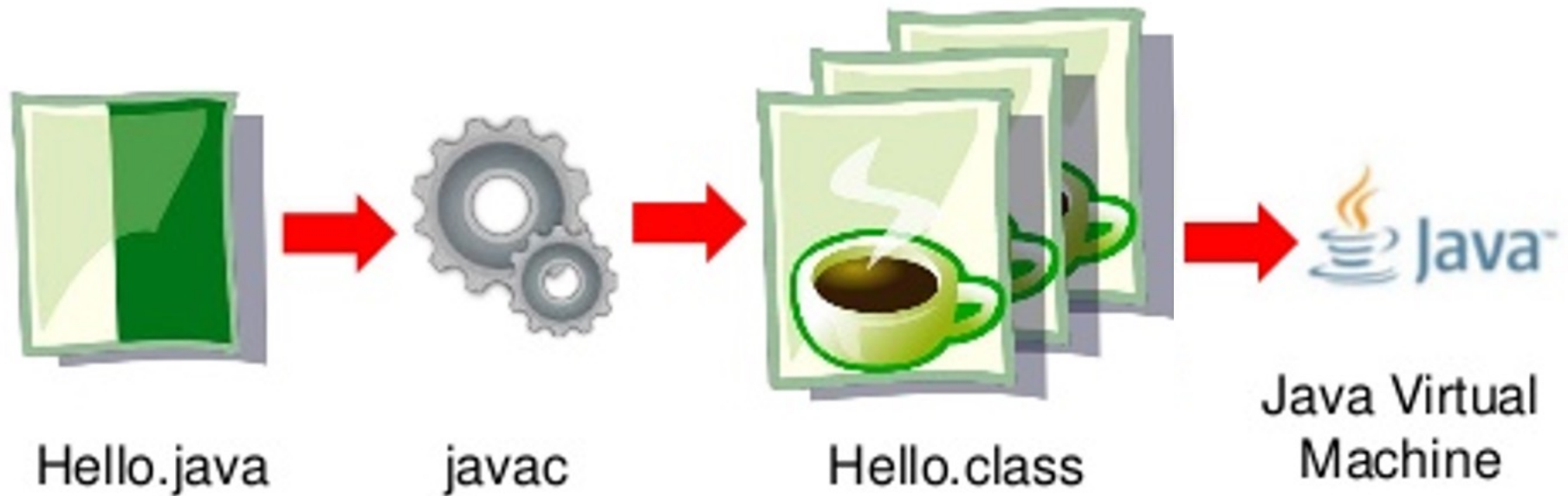
PRINCIPAIS CARACTERÍSTICAS DO JAVA

- Resumo



PRINCIPAIS CARACTERÍSTICAS DO JAVA

- **JVM:** O código-fonte é compilado para um código intermediário (“bytecode”) e depois interpretado/executado pela Máquina Virtual Java.



PRINCIPAIS CARACTERÍSTICAS DO JAVA

- Projeto Orientado a Objetos



- Quais classes:
Cliente, Pedido, Produto
idCliente, idProduto, idProduto
↳ CPF
↳ nome
↳ end
↳ vetor
↳ total Pedido
↳ Descrição
↳ Valor

PRINCIPAIS CARACTERÍSTICAS DO JAVA

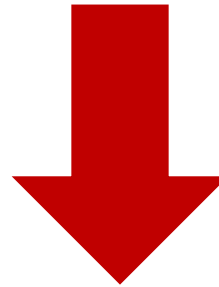
- **Funcional:** Além do foco na programação orientada a objetos (POO), a partir do Java 8 conceitos fundamentais de programação funcional foram incorporados à linguagem.

- **Expressões Lambda:** Expressões lambda permitem tratar funções como cidadãos de primeira classe, o que significa que você pode passar funções como argumentos para outros métodos, retorná-las de métodos e armazená-las em variáveis.
- **Interfaces Funcionais:** São interfaces que contêm exatamente um método abstrato. Elas são frequentemente usadas como tipos para expressões lambda.
- **Streams:** São uma parte essencial da programação funcional em Java. Elas representam uma sequência de elementos e suportam operações de alto nível como mapeamento, filtragem, redução, etc.
- **Optional:** A classe `java.util.Optional` é uma maneira de representar valores opcionais ou nulos de forma segura, sem causar exceções de ponteiro nulo (`NullPointerException`).

PRINCIPAIS CARACTERÍSTICAS DO JAVA

- Projeto Baseado em Programação Funcional

1. Listar Funcionários
2. Listar Cidades Funcionários
3. Calcular Folha Funcionários
4. Listar Funcionários Idade < 30
4. Listar Funcionários Idade >= 30



```
{  
    "Total de Funcionários >= 30 anos": 5,  
    "Média Salarial": 6000,00  
}
```

PRINCIPAIS CARACTERÍSTICAS DO JAVA

- Projeto Baseado em Programação Funcional

```
@GetMapping("/listarFuncionariosIdadeMaiorIgual30")
public ResponseEntity<String> listarFuncionariosIdadeMaiorIgual30()
{
    List<Funcionario> funcionarios = funcionarioService.listAll();

    long totalFuncionarios = funcionarios.stream().filter(f -> f.getIdade() >= 30).count();
    double somaSalarios = funcionarios.stream().
        filter(f -> f.getIdade() >= 30).mapToDouble(Funcionario::getSalario).sum();

    double mediaSalarial = somaSalarios/totalFuncionarios;

    //Java 13 -> Text Blocks
    String resposta = """
        {
            "Total de Funcionários >= 30 anos": %d,
            "Média Salarial": %.2f
        }
        """.formatted(totalFuncionarios,mediaSalarial);

    return new ResponseEntity<String>(resposta,HttpStatus.OK);
}
```

PRINCIPAIS CARACTERÍSTICAS DO JAVA

- **Open Source:** Java é uma plataforma de código aberto que incentiva a colaboração e a contribuição da comunidade para o desenvolvimento contínuo da linguagem e da plataforma.

- A implementação principal da plataforma Java é mantida e desenvolvida pelo projeto OpenJDK (Open Java Development Kit), que é uma colaboração de desenvolvedores de várias empresas e da comunidade de código aberto.
- O OpenJDK inclui o compilador Java (javac), a Máquina Virtual Java (JVM) e as bibliotecas Java padrão, juntamente com outras ferramentas e utilitários relacionados. Ele está disponível sob uma licença de código aberto conhecida como a Licença Pública Geral GNU (GPL).

PRINCIPAIS CARACTERÍSTICAS DO JAVA

- *Open Source*



PROGRAMAÇÃO ORIENTADA A OBJETOS

O QUE É A ORIENTAÇÃO A OBJETOS?

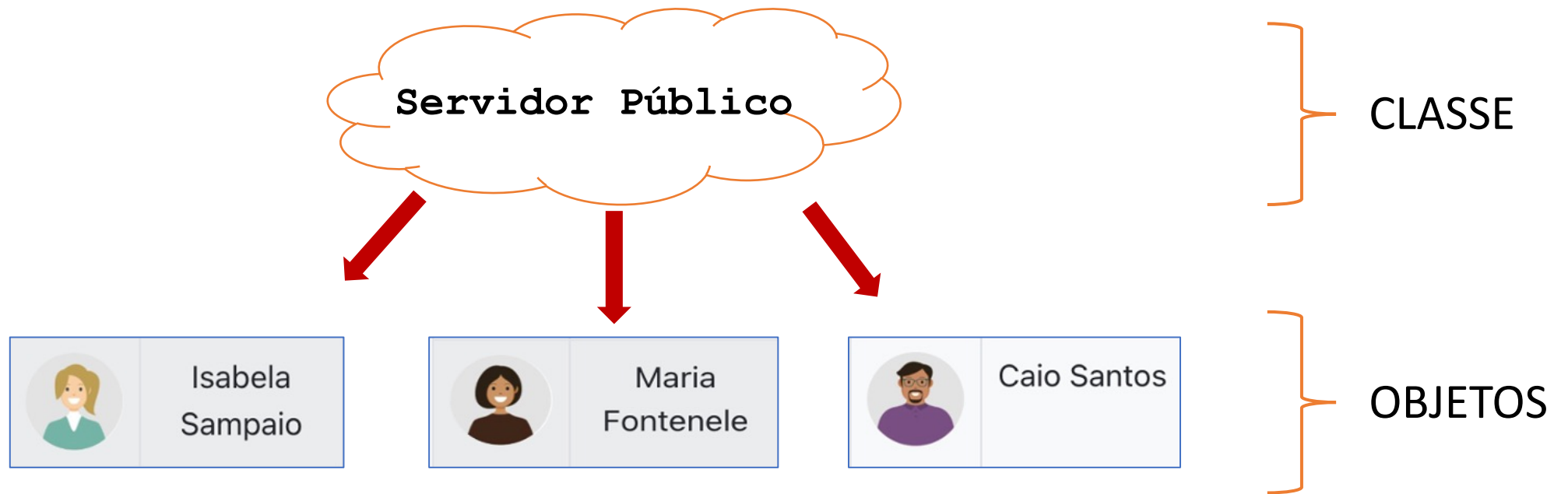
- **Definição**

- É um paradigma de programação que usa os conceitos de objetos e classes como elementos centrais para representar e processar dados usados nos programas. Os objetos e classes nada mais são do que representações do mundo real.



O QUE É A ORIENTAÇÃO A OBJETOS?

- **Exemplo**
 - Os “objetos” são criados a partir de modelos chamados de “classes”.



O QUE É A ORIENTAÇÃO A OBJETOS?

- Exemplo da TELA PRINCIPAL do **Sistema de Capacitação de Servidores Públicos (SISCAPACIT)**


Foto	Nome	Cargo	Órgão	Lotação	Email	Cursos	Ações
	Isabela Sampaio	Auditor	ANVISA	Brasília	assoftbel@gmail.com	Não Tem Cursos	Detalhar Alterar Excluir
	Heila Ghassan	Estagiário	STN	Brasília	heila@gmail.com	Não Tem Cursos	Detalhar Alterar Excluir
	Maria Fontenele	Analista	ENAP	Brasília	mariafontenele@enap.br	Cursos Matriculados	Detalhar Alterar Excluir
	Caio Santos	Analista Tributário	RFB	Rio de Janeiro	caiosantos@rfb.gov.br	Cursos Matriculados	Detalhar Alterar Excluir

04 Objetos da Classe “Servidor Público”

O QUE É A ORIENTAÇÃO A OBJETOS?

- OBJETO **SERVIDORPUBLICO** 'ISABELA SAMPAIO'

Isabela Sampaio



Informações Funcionais

Matrícula: 6	Lotação: Brasília
Órgão: ANVISA	Exercício: SUPEN
Cargo: Auditor	Vínculo: Estatutário
Data de Admissão: 2022-11-21T12:55:40.496Z	

Informações Pessoais

Email: assoftbel@gmail.com	CPF: 00000000000
Telefone: (91)991124552	Data de Nascimento: 2022-08-03T12:55:40.496Z
Celular: (91)981144444	Naturalidade: Belém

Voltar

O QUE É A ORIENTAÇÃO A OBJETOS?

- **Membros de uma Classe - Atributos e Métodos**
 - Toda Classe define um conjunto de atributos (também conhecidos como propriedades) e um conjunto de métodos (que definem o seu comportamento).

Atributos

- matricula
- nome
- orgao
- cargo
- celular

substantivos



Métodos

- calcularTempoAposentadoria()
- listarContatos()
- verificarDireitoBeneficios()
- listarInformacoesPessoais()
- listarInformacoesProfissionais()

verbos

O QUE É A ORIENTAÇÃO A OBJETOS?

- Declaração de Atributos e Métodos no Java

```
public class ServidorPublico
{
    //Definição dos Atributos
    private int matricula; private String nome; private String foto;
    private String orgao; private String vinculo; private double salario;
    private int idade; private int tempoContribuicao; private String cargo;
    private String lotacao; private String exercicio; private String email;
    private String telefone; private String celular; private String cpf;
    private String naturalidade;

    //Definição dos Métodos
    public int calcularTempoAposentadoria() { ... }
    public void listarContatos() { ... }
    public boolean verificarDireitoBeneficios() { ... }
    ...
}
```

Representação em UML

ServidorPublico

- matricula: int
- nome: String
- foto: String
- opção: String

+ listarContatos(): void

O QUE É A ORIENTAÇÃO A OBJETOS?

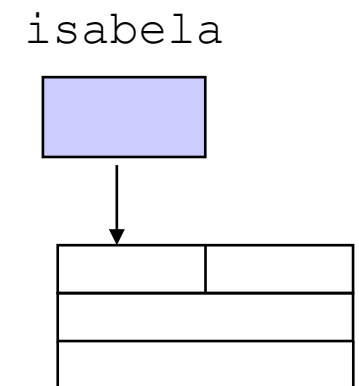
- OBJETO **SERVIDORPUBLICO** 'ISABELA SAMPAIO'

```
public class ServidorPublico
{
    private int matricula; private String nome;
    private String foto;
    private String orgao;
    private String vinculo;
    private double salario;
    private int idade;
    private int tempoContribuicao;
    private String cargo;
    private String lotacao;
    private String exercicio;
    private String email;
    private String telefone;
    private String celular;
    private String cpf;
    private String naturalidade;
}
```

Criando o Objeto
"Isabela Sampaio"

```
ServidorPublico isabela = new
ServidorPublico(6, "Isabela
Sampaio", "isabela.png", "ANVISA",
"Estatutário", "20000", "25", "7", "Aud
itor", "Brasília", "SUPEN", "assoftbel
@gmail.com", "(91)991124552", "(91)98
1144444", "123.4567.789-
01", "Belém");
```

- Um objeto de uma classe é criado utilizando-se a palavra **new**.
- Aloca um espaço em memória.



PROGRAMAÇÃO FUNCIONAL

EXPRESSÕES LAMBDA

- **Definição**

- Até a versão 7 Java pode ser definida como uma linguagem imperativa Orientada a Objetos. A partir da versão 8, Java também passa a incorporar funcionalidades do paradigma funcional, quando adota em sua sintaxe o cálculo Lambda!
- O paradigma funcional fornece muitos benefícios na construção de aplicações baseadas em um hardware composto por chip de vários núcleos, visto que favorece enormemente a execução das instruções de máquina de forma paralela.
- O paradigma funcional é baseado no conceito de função, isto é, pode-se definir a programação funcional como a simples avaliação de expressões matemáticas.

(parâmetros) -> expressão

EXPRESSÕES LAMBDA

- Exemplo

```
public class LambdaApp {  
    public static void main(String[] args) {  
        List<Integer>integers = Arrays.asList(1, 2, 3, 4, 5);  
        //Expressão Lambda  
        integers.forEach(x->System.out.println(x));  
    }  
}
```

```
public class LambdaApp2 {  
    public static void main(String[] args) {  
        List<Integer>integers = Arrays.asList(1, 2, 3, 4, 5);  
        integers.forEach(x ->{  
            x = x + 10;  
            System.out.println(x);  
        });  
    }  
}
```

STREAMS

- **Definição**

- O Java 8 introduziu o Stream. O Stream traz para o Java uma forma mais funcional de se trabalhar com coleções, usando uma interface específica para isso.
- Um Stream representa uma abstração de um “fluxo de dados”. No caso do Java 8, foi criado um pacote específico (`java.util.stream`) para permitir manipulações e transformações em coleções.
- A interface `Stream<T>` representa uma sequencia de elementos nos quais uma ou mais operações (*filter, sorted, map, match, count, reduce*) poderão ser executadas.

OPERAÇÕES COM STREAMS

- **MAP**

- Em Java 8, o método **map** é uma operação oferecida por objetos da classe Stream que permite transformar os elementos de um stream de acordo com uma função especificada. Ele retorna um novo stream contendo os resultados das transformações.



- O método **map** é especialmente útil quando se deseja transformar os elementos de um stream de maneira personalizada antes de prosseguir com outras operações.

OPERAÇÕES COM STREAMS

- **MAP - EXEMPLO**

- Em Java 8, o método `map` é uma operação oferecida por objetos da classe `Stream` que permite transformar os elementos de um stream de acordo com uma função especificada. Ele retorna um novo stream contendo os resultados das transformações.

```
. List<Integer> numeros = Arrays.asList(1,2,3,4,5);  
  
List<Integer> quadrados = numeros.stream()  
    .map(numero -> numero * numero)  
    .collect(Collectors.toList());  
  
System.out.println(quadrados);  
// Saída: [1, 4, 9, 16, 25]
```

OPERAÇÕES COM STREAMS

- **FILTER**

- Em Java 8, o método **filter** permite filtrar elementos de um stream com base em uma condição especificada, retornando um novo stream contendo apenas os elementos que atendem a essa condição.



- O método **filter** é uma parte fundamental das operações de filtragem e processamento de streams em Java.

OPERAÇÕES COM STREAMS

- **FILTER - EXEMPLO**

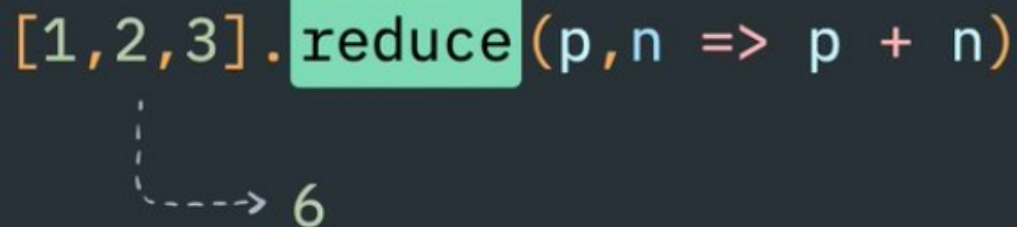
- Em Java 8, O método **filter** é usado para selecionar elementos de um stream com base em critérios específicos, o que é útil para processar ou exibir apenas os elementos desejados.

```
List<Integer> numeros = Arrays.asList(1,2,3,4,5,6,7,8,9,10);  
  
List<Integer> numerosPares = numeros.stream()  
    .filter(numero -> numero % 2 == 0)  
    .collect(Collectors.toList());  
  
System.out.println(numerosPares);  
// Saída: [2, 4, 6, 8, 10]
```

OPERAÇÕES COM STREAMS

- **REDUCE**

- Em Java 8, o método **reduce** é usado para combinar os elementos de um fluxo (stream) em um único resultado. Ele aceita um identificador inicial (também conhecido como valor de identidade) e uma função de acumulação que combina os elementos em sequência.



```
[1,2,3].reduce(p,n => p + n)
```

A diagram illustrating the `reduce` operation. It shows the expression `[1,2,3].reduce(p,n => p + n)` where `reduce` is highlighted in a green box. A dashed line with an arrow points from the array `[1,2,3]` to the number `6`, indicating the result of the reduction.

- A partir do Java 9, a classe **Stream** foi enriquecida com métodos ainda mais poderosos para operações de redução, como `reduce(...)` sem valor inicial e `collect` para coletar os resultados em diferentes tipos de coleções.

OPERAÇÕES COM STREAMS

- **REDUCE - EXEMPLO**

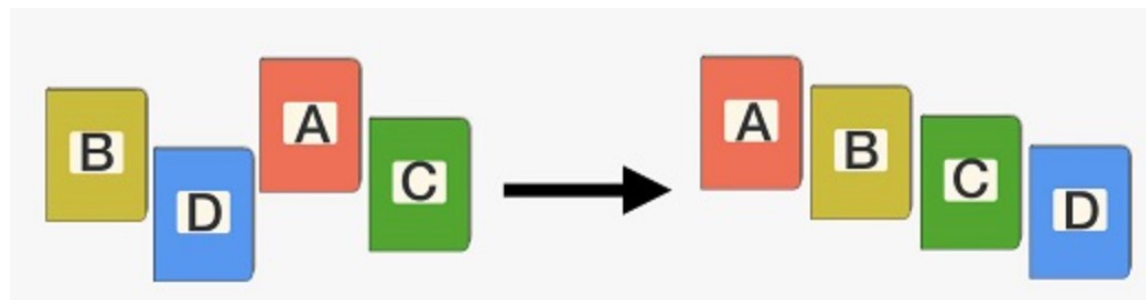
- A ideia principal por trás do reduce é reduzir (daí o nome) os elementos de um fluxo para um único valor.

```
List<Integer> numbers = Arrays.asList(1,2,3,4,5);  
  
int sum = numbers.stream()  
                  .reduce(0, (a, b) -> a + b);  
//Usando o método reduce para somar todos os elementos  
  
System.out.println("Soma: " + sum);  
// Saída: Soma: 15
```


OPERAÇÕES COM STREAMS

- **SORTED**

- Em Java 8, o método `sorted` permite ordenar os elementos de um fluxo com base em uma ordem específica. Ele retorna um novo fluxo contendo os elementos ordenados de acordo com um critério definido.



- Este método não recebe argumentos, pois a ordenação é baseada na ordem natural dos elementos.

OPERAÇÕES COM STREAMS

- **SORTED - EXEMPLO**

- O método **sorted** é bastante útil para situações em que você precisa ordenar os elementos de um fluxo de acordo com um critério específico, tornando a ordenação de coleções mais simples e mais legível.

```
List<Integer> numbers = Arrays.asList(5,2,8,1,3);  
  
List<Integer> sortedNumbers = numbers.stream()  
                                     .sorted()  
                                     .collect(Collectors.toList());  
  
System.out.println(sortedNumbers);  
// Saída: [1, 2, 3, 5, 8]
```

OPERAÇÕES COM STREAMS

- **COUNT**

- Em Java 8, o método `count` é usado para contar o número de elementos em um fluxo.
- É frequentemente usado para obter a contagem de elementos de um fluxo após a aplicação de várias transformações ou operações.

```
List<Integer> numbers = Arrays.asList(1,2,3,4,5,6,7,8,9,10);  
  
long totalCount = numbers.stream().count();  
  
System.out.println("Total de números: " + totalCount);  
// Saída: Total de números: 10
```

OPERAÇÕES COM STREAMS

- **SUM**

- Em Java 8, o método `sum` permite calcular a soma dos valores numéricos de um fluxo. Está disponível para fluxos de tipos numéricos, como *int*, *long*, *double*, e é usado para somar os valores desses tipos em um fluxo.

```
. IntStream numbers = IntStream.of(1,2,3,4,5);  
  
int sum = numbers.sum();  
  
System.out.println("Soma dos números: " + sum);  
// Saída: Soma dos números: 15
```

Exemplo PRÁTICO

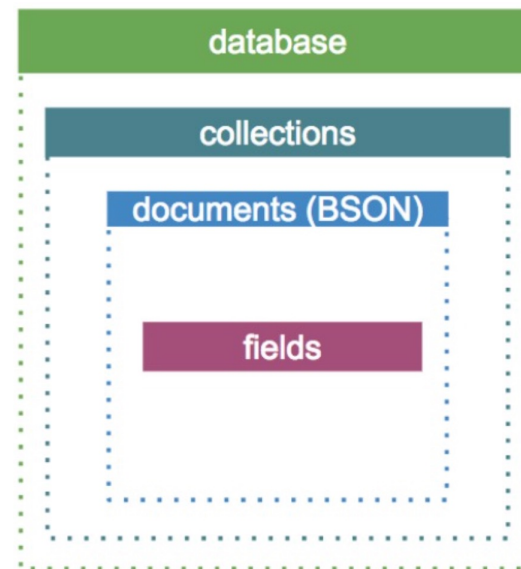


SPRING BOOT

- **VISÃO GERAL**

- O Java Spring Boot (Spring Boot) é uma ferramenta que torna **mais rápido e fácil o desenvolvimento de aplicações Web e de microsserviços** com o Spring Framework.
- São três os seus principais recursos:
 - (1) Autoconfiguração
 - (2) Abordagem Opinativa
 - (3) Aplicações Autônomas

- É um banco de dados não relacional gratuito, *open source*, de alta performance e flexível, sendo considerado o principal banco de dados NoSQL.
- O MongoDB é orientado a documentos, ou seja, os dados são armazenados como documentos, ao contrário de bancos de dados de modelo relacional, onde trabalhamos com registros em linhas e colunas. Os documentos podem ser descritos como dados no formato de chave-valor, no caso, utilizando o formato JSON.





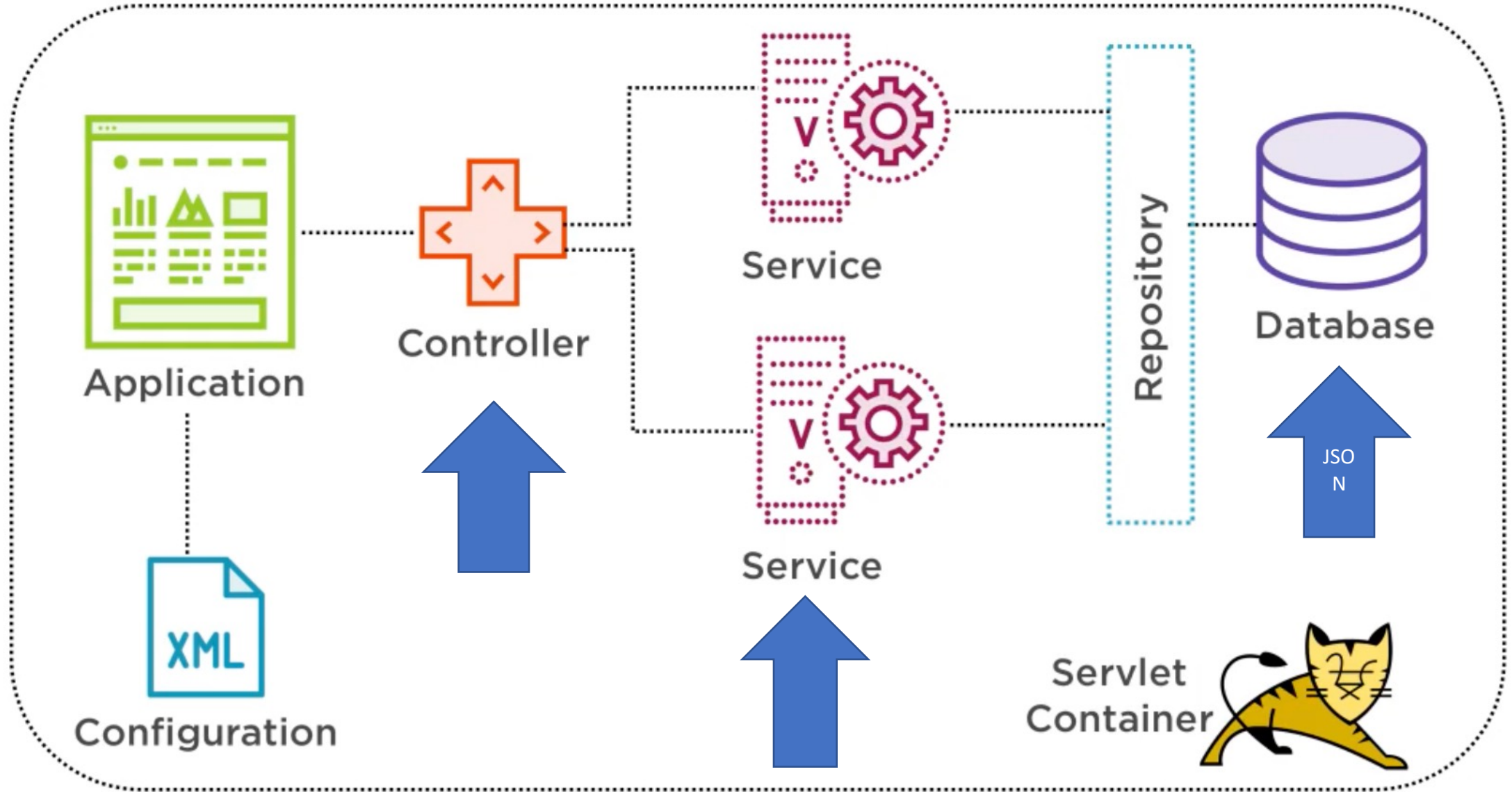
FORMATO JSON

- JSON (*JavaScript Object Notation*) é um formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas, especificado por Douglas Crockford em 2000, que utiliza texto legível a humanos, no formato atributo-valor.
- É ideal para enviar e receber informações pela Internet.

```
{
  "id": 123,
  "nome": "JSON T-Shirt",
  "preco": 99.99,
  "estoque": {
    "deposito": 300,
    "loja": 20
  }
}
```

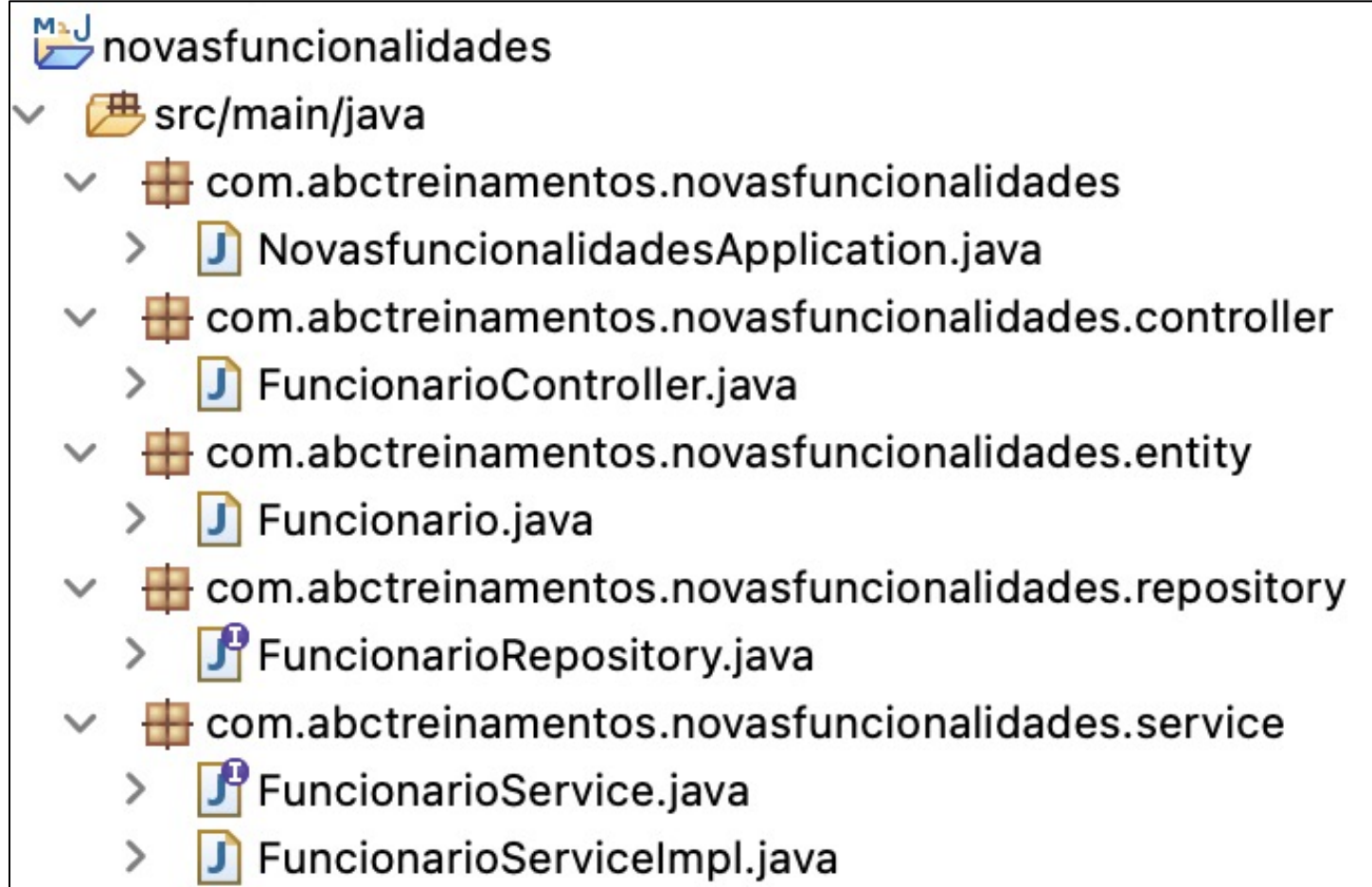


ARQUITETURA SPRING BOOT





ARQUITETURA DO PROJETO





CONSTRUÇÃO DO PROJETO

Passos:

- Instalar o JAVA 22
- Instalar o Eclipse 2024
- Instalar o MongoDB Server & Compass
- Gerar o Projeto (Spring Initializr)

PARTE PRÁTICA_1

- ATIVIDADES:
 - Criar os métodos abaixo na classe `FuncionarioController`:
 - (1) *listarCidadesFuncionarios()* que irá listar todas as cidades onde estão situados os funcionários.
 - (2) *calcularFolhaFuncionarios()* que irá calcular o total da folha de pagamentos dos funcionários.
 - (3) *listarFuncionariosIdadeMenor30()* que irá listar apenas os funcionários com idade inferior a 30 anos.
 - (4) *listarFuncionariosIdadeMaiorIgual30()* que irá listar apenas os funcionários com idade igual ou superior a 30 anos.

PARTE PRÁTICA_2

- ATIVIDADES:
 - Criar os métodos abaixo na classe `FuncionarioController`:
 - (1) *listarFuncionarioMaiorSalario()* que irá listar apenas o funcionário com o maior salário.
 - (2) *listarFuncionarioMenorSalario()* que irá listar apenas o funcionário com o menor salário.
 - (3) *listarSalariosFuncionariosOrdenados()* que irá listar os salários dos funcionários ordenados.
 - (4) *listarAnoNascimentoFuncionarioMaisAntigo()* que irá listar o provável ano de nascimento do funcionário mais antigo.