SENAI

Serviço Nacional de Aprendizagem Industrial

PELO FUTURO DO TRABALHO

Base Java

Desenvolvimento de Sistemas

Prof. Me. Reneilson Santos

Fevereiro/2024

SENAI
PELO FUTURO DO TRABALHO

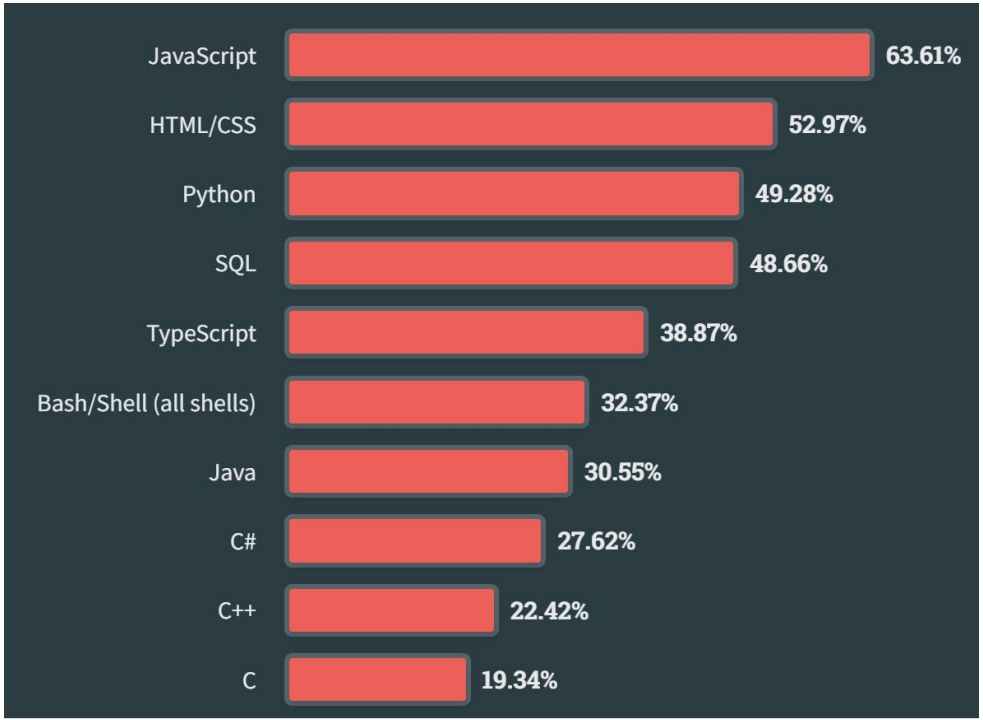
Agenda

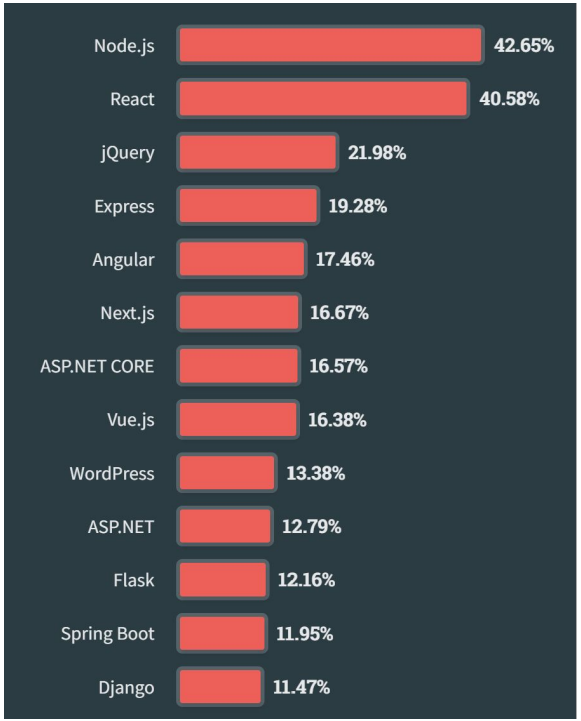
- Java
  - Primeiro Programa
  - Entrada de Dados
  - Operadores
  - Estrutura Condicional
  - Laços de Repetição
  - Listas
  - Dicionários
  - Strings
  - Formatação de Strings
  - Tratamento de Exceção

Java

Java

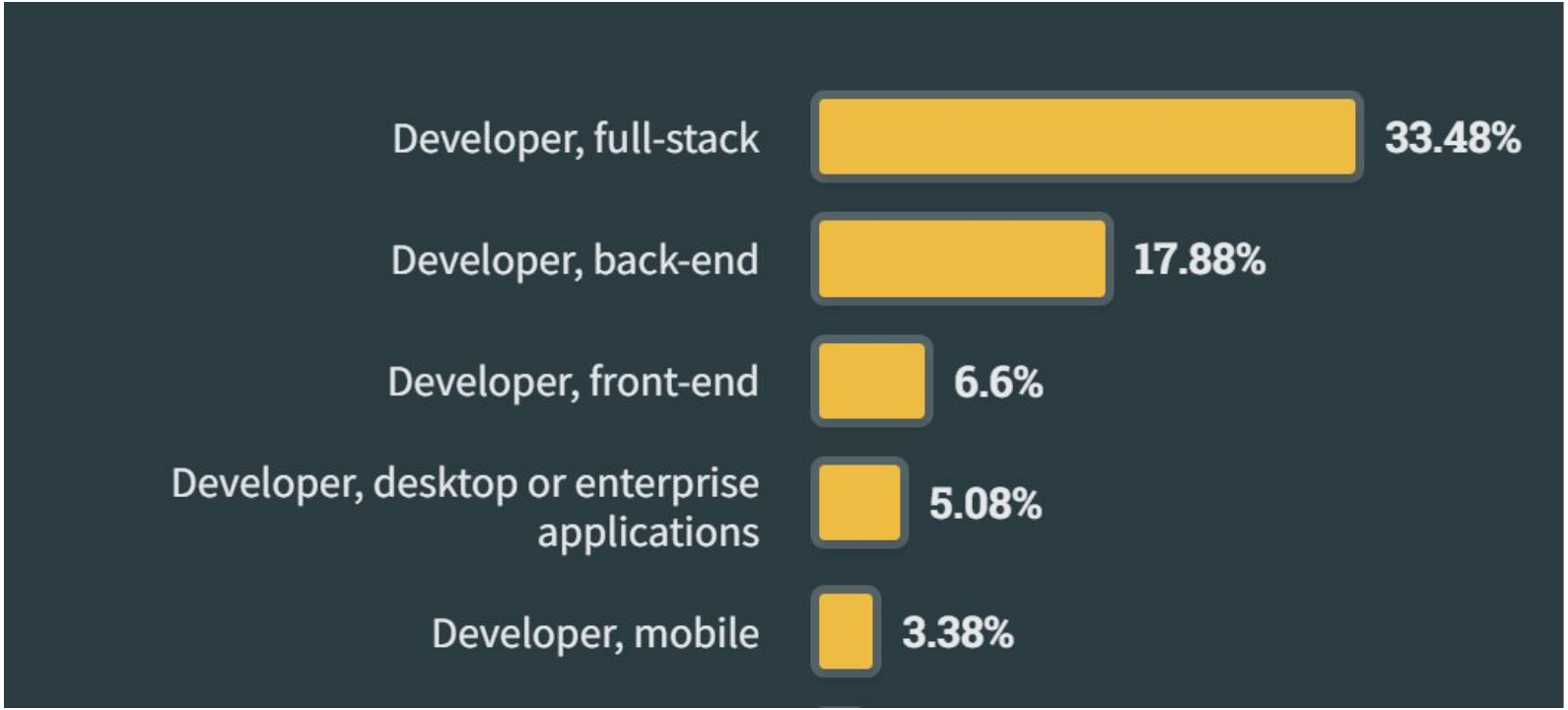
Créditos: "https://survey.stackoverflow.co/2023/ ", StackOverflow





Desenvolvedores

Créditos: "https://survey.stackoverflow.co/2023/ ", StackOverflow



Java

Java mantém sua popularidade devido à sua portabilidade, versatilidade e confiabilidade. Sua capacidade de funcionar em diferentes plataformas, o vasto ecossistema de bibliotecas e frameworks, além da robustez na construção de aplicações empresariais, contribuem para sua longevidade. A linguagem também evoluiu com recursos modernos, mantendo-se relevante para desenvolvimento web, móvel e corporativo.

Primeiro Programa em Java

Criando Hello World

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Classe HelloWorld

public: A palavra-chave "public" é um modificador de acesso que indica que a classe "HelloWorld" pode ser acessada de qualquer lugar no programa. ***Em Java, uma classe pública deve ter o mesmo nome do arquivo que a contém.***

class: A palavra-chave "class" é usada para definir uma classe em Java. Neste caso, estamos criando uma classe chamada "HelloWorld".

HelloWorld: É o nome da classe que estamos criando. ***O nome da classe sempre deve começar com uma letra maiúscula.***

Método Main

O método main é o ponto de entrada para a execução do programa Java.

Quando você executa um programa Java, o sistema procura pelo método "main" para começar a execução.

A assinatura do método é:

public static void main(String[] args)

Método Main

public: a palavra-chave "public" indica que o método "main" pode ser acessado de qualquer lugar no programa.

static: A palavra-chave "static" indica que o método "main" pertence à classe em vez de pertencer a instâncias (objetos) dessa classe. Isso permite que o método seja chamado sem a necessidade de criar um objeto da classe.

void: "void" é um *tipo* de retorno do método, o que significa que o método "main" não retorna nenhum valor.

main: "main" é o nome do método. É o nome padrão do método que o sistema procura ao iniciar a execução.

Método Main - Parâmetros

String[] args

Isso é chamado de parâmetro/argumento do método "main".

Ele permite que você passe argumentos da linha de comando para o programa.

Quando você executa um programa Java a partir da linha de comando e fornece argumentos, eles são passados para o método "main" como um array de strings.

Corpo do Método

System.out.println("Hello, World!");: Esta é uma instrução de saída que imprime a mensagem "Hello, World!" no terminal.

System: "System" é uma classe predefinida em Java que representa o ambiente do sistema.

out: "out" é um objeto da classe "System" que representa a saída padrão. Neste caso, é o console (terminal) do sistema.

println: "println" é um método de "out" que *imprime a mensagem entre parênteses no console* e, em seguida, move o cursor para a próxima linha.

Entrada de Dados

Entrada de Dados em Java

Em uma aplicação Java pura, fora do contexto web, para solicitarmos a entrada de dados via terminal para o usuário podemos utilizar a classe **Scanner**, que, ao contrário do System, não é uma classe padrão e portanto deve ser importada no nosso código.

A classe Scanner fará a leitura do que o usuário digitar e armazenará em alguma variável dentro do nosso código.

Em Java, toda variável é **tipada**, ou seja, é **obrigatório definir qual o tipo da variável** e caso o usuário digite algo distinto daquilo que estava sendo aguardado, erros acontecerão no nosso programa.

```java
import java.util.Scanner;

public class EntradaDados {
    public static void main(String[] args) {
        // Criação do objeto Scanner para ler a entrada do teclado
        Scanner scanner = new Scanner(System.in);
        // Solicita o nome da pessoa
        System.out.print("Digite o seu nome: ");
        String nome = scanner.nextLine();
        // Fecha o objeto Scanner, pois não será mais utilizado
        scanner.close();
        // Imprime a saudação personalizada
        System.out.println("Olá, " + nome + "! Bem-vindo(a) ao Java!");
    }
}
```

Tipos em Java

Família	Tipo Primitivo	Classe Invólucro	Tamanho	Exemplo
Lógico	boolean	Boolean	1 bit	true
Literais	char	Character	1 byte	'A'
	-	String	1 byte/cada	"JAVA"
Inteiros	byte	Byte	1 byte	127
	short	Short	2 bytes	32 767
	int	Integer	4 bytes	2 147 483
	long	Long	8 bytes	2^{63}
Reais	float	Float	4 bytes	$3.4e^{+38}$
	double	Double	8 bytes	$1.8e^{+308}$

Operadores

Operadores Java

OPERADOR	FUNÇÃO	OPERADOR	FUNÇÃO
+	Adição	~	Complemento
-	Subtração	<<	Deslocamento à esquerda
*	Multiplicação	>>	Deslocamento à direita
/	Divisão	>>>	Desloc. a direita com zeros
%	Resto	=	Atribuição
++	Incremento	+=	Atribuição com adição
--	Decremento	-=	Atribuição com subtração
>	Maior que	*=	Atribuição com multiplicação
>=	Maior ou igual	/=	Atribuição com divisão
<	Menor que	%=	Atribuição com resto
<=	Menor ou igual	&=	Atribuição com AND
==	Igual	|=	Atribuição com OR
!=	Não igual	^=	Atribuição com XOR
!	NÃO lógico	<<=	Atribuição com desl. esquerdo
&&	E lógico	>>=	Atribuição com desloc. direito
||	OU lógico	>>>=	Atrib. C/ desloc. a dir. c/ zeros
&	AND	? :	Operador ternário
^	XOR	(*tipo*)	Conversão de tipos (cast)
|	OR	instanceof	Comparação de tipos

Atividade 1

Digite a temperatura em graus Celsius: 25.5
A temperatura em graus Fahrenheit é: 77.9

Peça ao usuário para inserir uma temperatura em graus Celsius (número real) e, em seguida, imprima a temperatura equivalente em graus Fahrenheit.

Use a fórmula: Fahrenheit = Celsius * 9/5 + 32.

Atividade 2

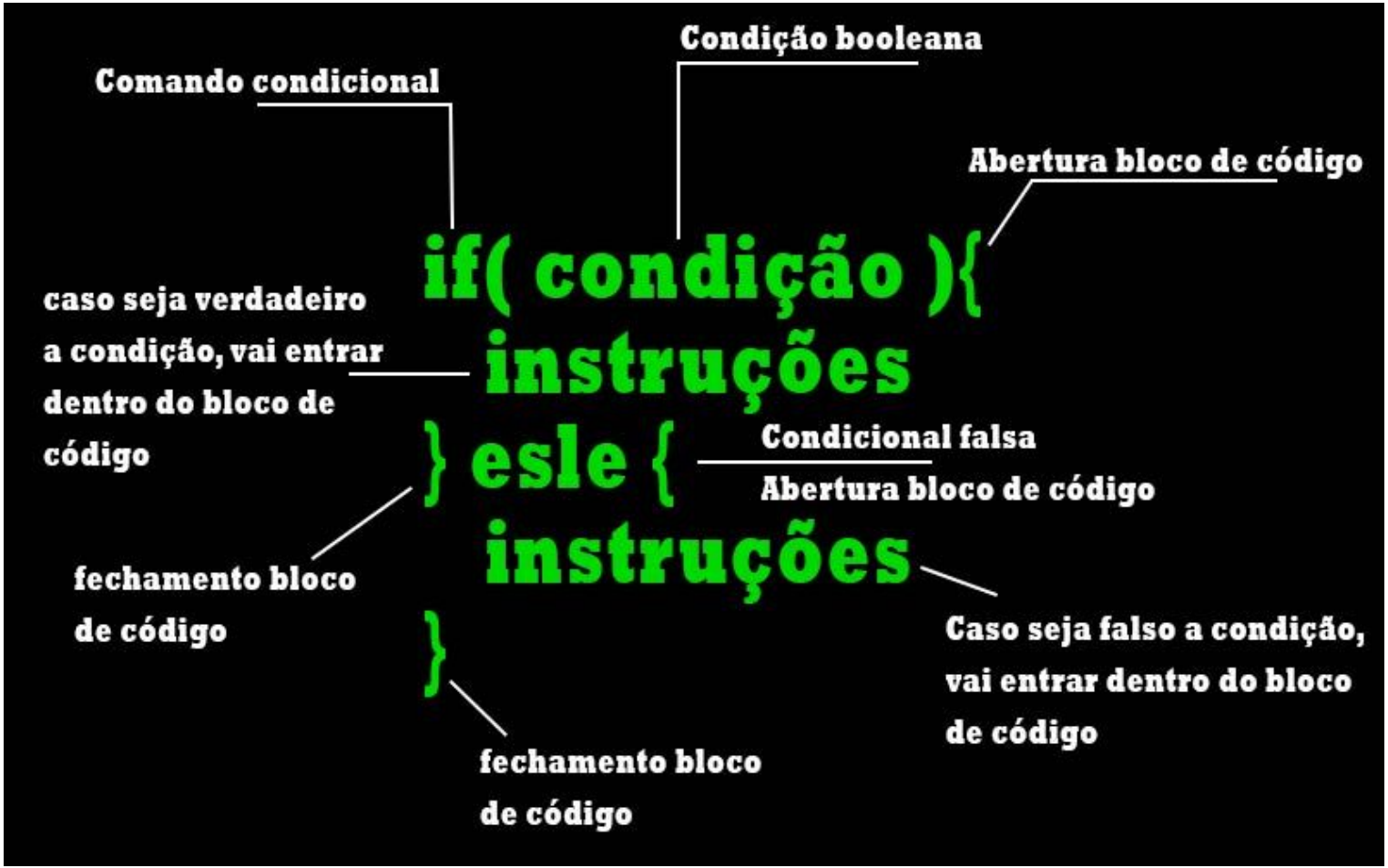
Digite um número inteiro: 6
O número é par? true
O número é positivo? true

Peça ao usuário para inserir um número inteiro e, em seguida, imprima se o número é par e se é positivo.

Não usar if-else. Utilize operadores booleanos para verificar essas condições.

Estruturas Condicionais

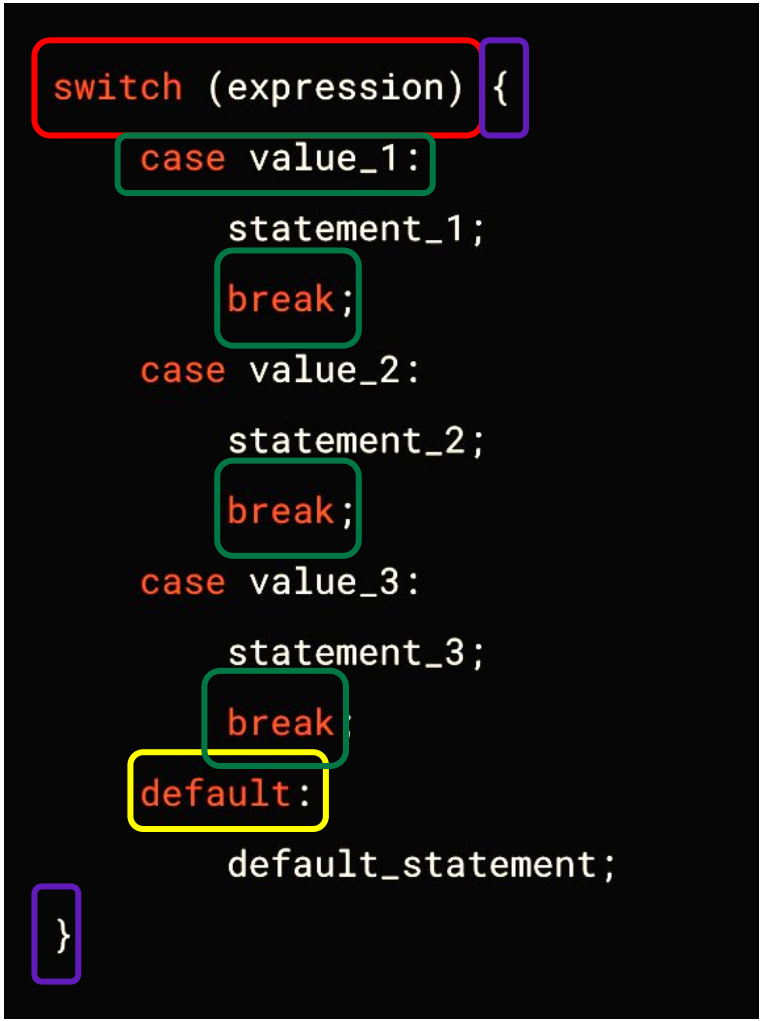
If/Else


Comando condicional
Condição booleana
Abertura bloco de código
if(condição){
instruções
caso seja verdadeiro a condição, vai entrar dentro do bloco de código
} esle {
Condicional falsa
Abertura bloco de código
instruções
fechamento bloco de código
Caso seja falso a condição, vai entrar dentro do bloco de código
}
fechamento bloco de código

If/Else

```
if (resposta == 10) {
    System.out.println("A resposta é exatamente 10!");
} else if (resposta > 10) {
    System.out.println("A resposta é maior que 10!");
} else {
    System.out.println("A resposta é menor que 10!");
}
```

Switch/Case


switch (expression) {
case value_1:
statement_1;
break;
case value_2:
statement_2;
break;
case value_3:
statement_3;
break;
default:
default_statement;
}

Switch/Case

```
switch (mes) {
    case 1:
        System.out.println("Domingo");
        break;
    case 2:
        System.out.println("Segunda-feira");
        break;
    case 3:
        System.out.println("Terça-feira");
        break;
    case 4:
        System.out.println("Quarta-feira");
        break;
    case 5:
        System.out.println("Quinta-feira");
        break;
    case 6:
        System.out.println("Sexta-feira");
        break;
    case 7:
        System.out.println("Sábado");
        break;
    default:
        System.out.println("Dia inválido");
        break;
}
```

Atividade 3

Digite o tamanho do primeiro lado do triângulo: 5.5
Digite o tamanho do segundo lado do triângulo: 3.2
Digite o tamanho do terceiro lado do triângulo: 7.0
É possível formar um triângulo? true

Peça ao usuário para inserir os tamanhos dos três lados de um triângulo (números reais) e, em seguida, imprima se é possível formar um triângulo com esses lados.

Utilize operadores comparativos para verificar se a soma de dois lados é maior do que o terceiro lado.

Laços de Repetição

While

```
while (<condição>) {
    // Trecho de código a ser repetido
 }
```

```
public int potencia(int number, int expoente){
   int result = 1;
   while(expoente > 0){
       result = result * number;
       expoente = expoente - 1;
   }
   return result;
}
```

For

```
for (<variável de controle>; <análise da variável de controle>; <incremento da variável de controle>) {
    // Código a ser executado
}
```

```
public int potencia(int number, int expoente){
    int result = 1;
    for(int i=expoente; i>0; i--){
        result = result * number;
    }
    return result;
}
```

Atividade 4

Tabuada

Crie um programa que permita ao usuário calcular a tabuada de um dado número. Ou seja, esse número multiplicado por 1 até 10.

Listas

Array

Os arrays em Java fazem parte do pacote **java.util**.

Um array pode ser declarado de forma estática sem utilização de classes extras, com um tamanho pré-definido de itens:

```
int[] a = new int[4];
```

ou já com os valores predefinidos:

```
int[] arr = {12,32,54,6,8,89,64,64,6};
```

Array

```
public static void printArray(){
        int[] arr = {12,32,54,6,8,89,64,64,6};
        for(int i = 0; i<arr.length; i++){
            System.out.println(arr[i]);
        }
    }
```

List e Array List

Enquanto o array em java é estático, sendo inicializado com um número específico de itens, uma lista vinda da classe List (pacote **java.util**) cria um array dinâmico, que permite a inserção de valores tantos quantos forem necessários.

```
List<Class> lista = new ArrayList<Class>();
```

Onde "`Class`" pode ser tanto uma classe padrão do java como Integer, String, Boolean ou uma classe do projeto ou de outras bibliotecas.

```java
List<Integer> lista = new ArrayList<Integer>();
// Adição de elementos na lista
lista.add(1);
lista.add(2);
// Alterando o primeiro valor da lista para 4
lista.set(0, 4);
// Capturando o valor na posição 1 da lista (lembrando que começa
pela posição 0)
lista.get(1);
// Removendo o item 0 da lista
lista.remove(0)
// Apagando itens da lista
lista.clear();
// Capturando tamanho da lista
int tamanho = lista.size();
```

List - Principais Métodos

– **add(elemento):** adiciona elemento no final da lista

– **add(posição, elemento):** adiciona elemento em uma posição da lista

– **set(posição, elemento):** troca o elemento em uma posição da lista

– **remove(elemento):** remove um elemento da lista

– **remove(posição):** remove o elemento que está em uma posição da lista

List - Principais Métodos

– **clear():** remove todos os elementos da lista

– **get(posição):** retorna o elemento em uma posição da lista

– **indexOf(elemento):** retorna a posição de um elemento da lista

– **isEmpty():** informa se a lista está vazia

– **size():** informa o número de elementos da lista

For (com listas)

```
List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9);
for(Integer n: list) {
   System.out.println(n);
}
ou
list.forEach(n -> System.out.println(n));
```

Atividade 5

Crie um programa em Java que solicita ao usuário que digite um número inteiro N, que representa a quantidade de notas a serem lidas. Em seguida, o programa deve ler as N notas digitadas pelo usuário e calcular a média aritmética dessas notas.

Dicionários

Dicionários

- Os dicionários (que na linguagem de programação Java se criam com o Map) são estruturas de dados que implementam mapeamentos (coleção de associações entre pares de valores);
- O primeiro elemento do par é chamado de chave (identificador) e o outro é chamado de valor.
- Declarando um Map e instanciando um HashMap [implementação padrão java para o Map]:

```
Map<Integer,String> pessoas = new HashMap<>()
```

Tipo de elemento para a chave

Tipo de elemento para o valor

Hashmap

```
Map<String,String> pessoas = new HashMap<>()

pessoas.put("Nelson Silva", 25);
pessoas.put("Larissa Fernandes", 37);
pessoas.put("Pedro Henrique", 52);
pessoas.put("Raquel Soares", 68);

pessoas.replace("Pedro Henrique", 100);
pessoas.remove("Larissa Fernandes");

System.out.printf("Nome das pessoas: %s\n", pessoas.keySet());
System.out.printf("Idade das pessoas: %s", pessoas.values());
```

Map - Principais Métodos

– **put(chave, valor)**: adiciona uma chave indexando um valor no dicionário

– **get(chave)**: retorna o valor indexado pela chave

– **getOrDefault(chave, valor):** retorna o valor indexado pela chave ou o valor default informado

– **keySet():** retorna um conjunto com todas as chaves do dicionário

– **remove(chave):** remove o valor indexado pela chave no dicionário

– **clear():** remove todas as entradas do dicionário

– **isEmpty():** informa se o dicionário está vazio

– **size():** informa o número de entradas do dicionário

Strings

Strings

Em Java, uma string é um **objeto** que representa uma sequência de caracteres.

As strings **são imutáveis**, o que significa que não é possível modificar o conteúdo de uma string existente.

Em vez disso, ***as operações de string geralmente criam novas strings com base no conteúdo de strings existentes***.

A classe **String** é usada para representar strings em Java. Ela inclui muitos métodos úteis para manipulação de strings, como concatenação, comparação, pesquisa e extração de substrings.

Strings

Imutabilidade: Como mencionado anteriormente, as strings são imutáveis. Isso significa que, se você quiser modificar uma string existente, é necessário criar uma nova string com base no conteúdo da string original. Por exemplo, "Hello".toUpperCase() retorna uma nova string "HELLO".

String pool: Em Java, uma *string pool* é mantida para armazenar strings literais. Isso significa que, se duas strings literais tiverem o mesmo conteúdo, elas serão armazenadas na mesma área de memória, economizando espaço e melhorando o desempenho. Por exemplo, "Hello" e "Hello" referenciam o mesmo objeto de string na pool de strings.

Métodos em Strings

Concatenação: É possível concatenar duas ou mais strings usando o operador +. Por exemplo, "Hello" + "world" resulta em uma nova string "Helloworld". Pode-se usar também o método concat(): “Hello”.concat(“World”)

Comparação: É possível comparar duas strings usando o método equals(). Por exemplo, "Hello".equals("World") retorna false.

Conversão: É possível converter outros tipos de dados em strings usando o método valueOf() da classe String. Por exemplo, String.valueOf(42) retorna a string "42".

Métodos em Strings

charAt(int position) - retorna o caracter na posição *position* passada como parâmetro.

compareTo e **compareToIgnoreCase -** realiza a comparação de duas strings (semelhante ao equals) mas o *ToIgnoreCase* é *case insensitive*, ou seja, desconsidera diferenças entre maiúsculas e minúsculas. No entanto, ao contrário do equals, não retorna um booleano, mas um valor numérico, 0 caso sejam iguais.

endsWith e **startsWith** - verifica se uma string termina ou começa com a substring passada como parâmetro.

Métodos em Strings

isEmpty - retorna um booleano dizendo se a string é ou não vazia.

split - cria um array de Strings com base no “regex” passado como parâmetro, ou seja, divide a string em várias strings. Exemplo: se fizermos o split de “uc1, uc2, uc3” com base na vírgula teremos 3 strings “uc1”, “uc2” e “uc3”.

substring(int i, int j) e **subSequence**- retornam uma parte da String com base nos parâmetros de entrada, ou seja, “corta” a String começando na posição i até a posição j.

Métodos em Strings

toLowerCase - retorna uma nova string com todos os caracteres minúsculos.

toUpperCase - retorna uma nova string com todos os caracteres maiúsculos.

trim - remove espaços em branco do início e do fim da string.

format - permite formatar uma String de acordo com as especificações (formatar número de casas decimais de um ponto flutuante, por exemplo).

Atividade 6

Contador de Palavras em uma Frase

Crie um programa em Java que solicita ao usuário que digite uma frase. O programa deve contar o número de palavras na frase e exibir o resultado na tela.

Para contar as palavras na frase, você pode seguir as seguintes etapas:

1. Divida a frase em uma lista de palavras. Para fazer isso, utilize o método split() da classe String em Java.
2. Obtenha o tamanho da lista de palavras. Para fazer isso, utilize o atributo length do array.
3. Exiba o número de palavras na frase na tela.

Format String

O método String.format() usa uma string de formato que especifica a maneira como a string formatada deve ser construída. A string de formato contém zero ou mais especificadores de formato que são **precedidos pelo caractere %**. Cada especificador de formato especifica o tipo de argumento que deve ser formatado e como ele deve ser formatado.

O método String.format() retorna uma nova string que é criada a partir da string de formato e dos argumentos fornecidos. O número e o tipo de argumentos dependem da string de formato e dos especificadores de formato usados.

Format String

Especificador	Formato
%s	String de caracteres
%d	Número inteiro decimal
%u	Número inteiro decimal sem sinal
%o	Número inteiro octal sem sinal
%x, %X	Número inteiro hexadecimal sem sinal, minúsculo ou maiúsculo
%f	Float
%2f	Double
%e, %E	Número real, em notação científica, minúsculo ou maiúsculo
%b	Boolean
%c	Caractere (char)

Format String

Cada especificador de formato pode incluir opções adicionais para controlar a largura do campo, o alinhamento e a precisão. Por exemplo, **%10s** especifica um campo de largura 10 para uma string e **%.2f** especifica um número decimal com duas casas decimais de precisão.

É importante notar que os especificadores de formato são case-sensitive, ou seja, %d e %D são diferentes.

É necessário fornecer argumentos suficientes para preencher todos os especificadores de formato na string de formato. Caso contrário, uma exceção MissingFormatArgumentException será lançada.

Atividade 7

Formatação de Texto

Solicitar um número inteiro positivo do usuário e retornar:

1. Seu valor em decimal
2. Seu valor em hexadecimal
3. Seu valor em octal
4. Seu valor convertido em caractere.

Se o usuário passar um número negativo, retorna uma mensagem de erro.

Métodos de Conversão de String

Parse

Para converter uma String para número, por exemplo, utilizamos o método parse do objeto referente:

Para inteiro: Integer.parseInt(str)

Para float: Float.parseFloat(str)

Para double: Double.parseDouble(str)

Isso é útil para quando recebemos um número digitado pelo usuário e precisamos convertê-lo antes de armazenar, por exemplo, em um banco de dados.

Tratamento de Exceção

Try - Catch - Finally

Um bloco **try** é chamado de bloco ***protegido*** porque, caso ocorra algum problema com os comandos dentro do bloco, **a execução desviará para os blocos catch correspondentes.**

O try é uma maneira mais robusta de tratar possíveis erros na execução de um código, evitando a quebra da execução do programa por falha. Com o try podemos evitar esta queda brusca e então tratar o erro da melhor forma.

```
try {
   // código que inclui comandos/invocações de métodos
   // que podem gerar uma situação de exceção/erro.
} catch (XException ex) {
   // bloco de tratamento associado à condição de
   // exceção XException ou a qualquer uma de suas
   // subclasses, identificada aqui pelo objeto
   // com referência ex
} catch (YException ey) {
   // bloco de tratamento para a situação de exceção
   // YException ou a qualquer uma de suas subclasses
} finally {
   // bloco de código que sempre será executado após
   // o bloco try, independentemente de sua conclusão
   // ter ocorrido normalmente ou ter sido interrompida
}
```

Exemplo

```
try {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Digite um número: ");
    Integer numero = scanner.nextInt();
    System.out.println("O número digitado foi: " + numero);
} catch (Exception e) {
    System.out.println("Valor inválido!");
} finally {
    System.out.println("Programa encerrado!");
}
```

Atividade 8

Divisão

Crie um programa em Java que receba dois valores, o divisor e o dividendo e retorne o resultado da divisão.

Deve ser validado de a entrada é um número válido (tem que ser número, não pode ser letra).

Usar bloco try-catch para fazer o tratamento da exceção.

Referências

Referências

- Blocos Try-Catch. Disponível em <https://www.devmedia.com.br/blocos-try-catch/7339>. Acessado em 21/02/2023.
- Dicionários em Java. Disponível em: <https://caffeinealgorithm.com/blog/dicionarios-map-em-java/>. Acessado em 21/02/2023.
- Java 8 Streams. Disponívem em: <https://www.sitepoint.com/java-8-streams-filter-map-reduce/>. Acessado em 21/02/2023.
- Tipos Enum no Java. Disponível em: https://www.devmedia.com.br/tipos-enum-no-java/25729. Acessado em 21/02/2023.