Google olabi

<ESTRUTURADEDADOS EMJAVA>

{ Operadores Java

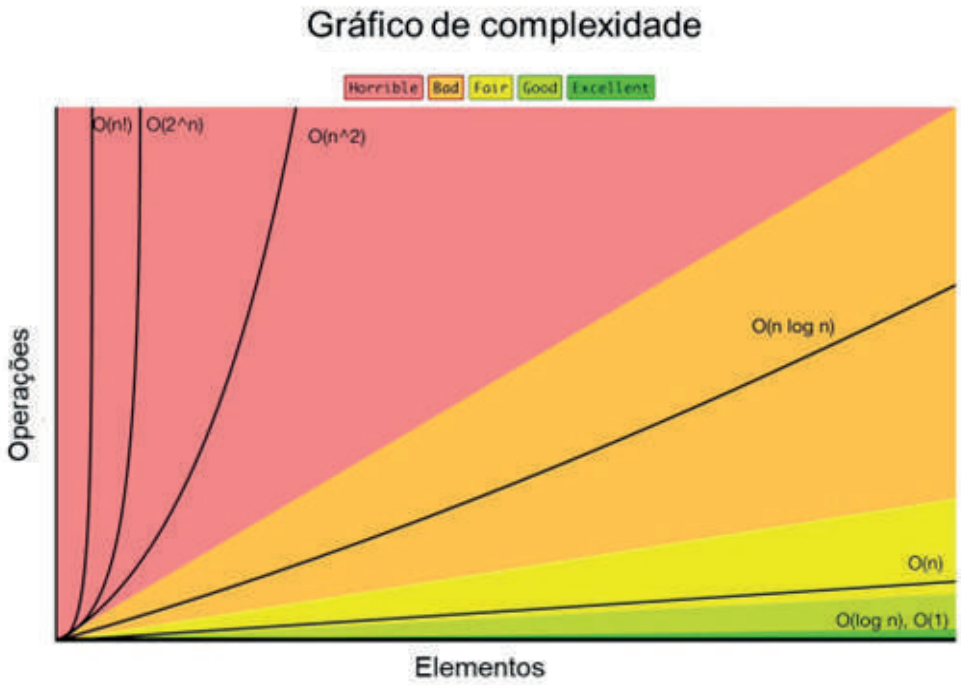
Com Java, quando precisamos tratar grandes quantidades de dados, temos a possibilidade de utilizarmos estruturas de dados. Com essas estruturas é possível manipular dados através de meios especiais, como por arrays (também conhecidos como vetores), arraylists, listas, filas e pilhas, entre outros.

Conceitos que veremos durante a aula:

<Notação Assimptótica (Grande – O)>

"A notação Big O é uma notação matemática que descreve o comportamento limitante de uma função quando o argumento tende a um valor específico ou ao infinito. Ela pertence a uma família de notações inventadas por Paul Bachmann, Edmund Landau e outros, coletivamente chamadas de notação Bachmann–Landau ou de notação assintótica."

- Definição da Wikipédia de notação Big O



Falando de forma simples, conseguimos com a notação Big O descrever a complexidade do nosso código através de termos algébricos.

O Big O conhecido como quadrática é representado pelo $O(n^2)$, onde o "n" representa o tamanho da entrada, enquanto a função " n^2 " interna ao "O()" nos dá a ideia da complexidade do algoritmo em relação ao tamanho da entrada.

Um algoritmo típico tem definido como nível de complexidade o $O(n^2)$ e é chamado de algoritmo selection sort (ordenação por seleção). Esse algoritmo itera a ordenação de uma lista, garantindo que cada elemento do índice seja refletido.

<Java libraries>

No Java existem bibliotecas com uma coleção de recursos que são funcionalidades já prontas para facilitar o desenvolvimento.

Vamos detalhar algumas que são padrão no Java:

A **Java Standard Library** é uma biblioteca bastante popular e usada pelos desenvolvedores. Contém uma lista de bibliotecas dentro dela, sem as quais não é possível realizar atividades básicas. Essas bibliotecas são chamadas em tempo de execução pela JVM (Java Virtual Machine).

Não podemos escrever nenhum programa em Java sem **String, Enum, Double** etc...

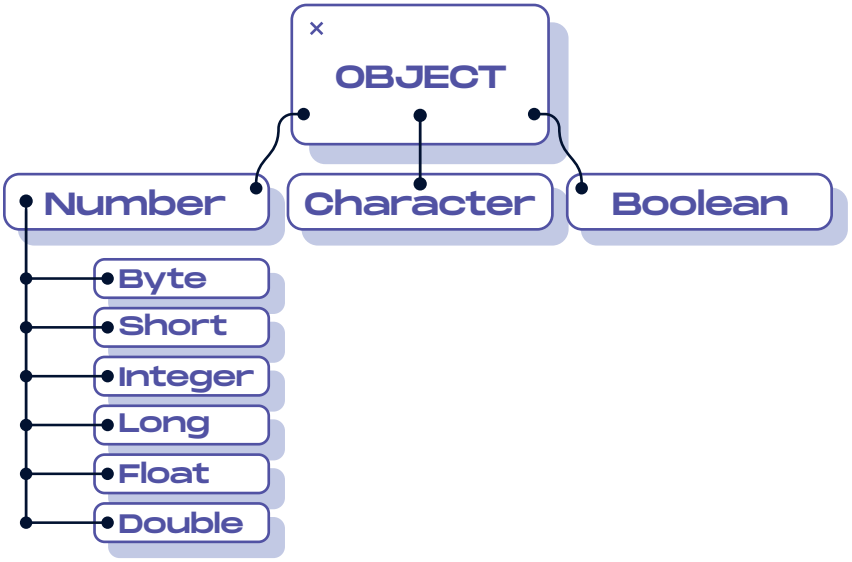
A biblioteca **java.lang** nos fornece o necessário para escrever nosso código. Para usar estruturas e coleções de dados em Java, precisamos da classe **java.util** porque ela contém a definição de todas as estruturas e coleções de dados.

A biblioteca **math** é uma das bibliotecas usadas para cálculos matemáticos, como a soma de BigInteger ou BigDecimal.

{ Numbers

O tipo de dado mais básico de qualquer linguagem de computador é o número. Geralmente, ao trabalhar com números em Java, usamos tipos de dados primitivos que são byte, short, int, long, float e double. Como sabemos que esses tipos de dados são apenas valores de dados e não objetos de classe, às vezes precisamos de valores numéricos na forma de objetos. Java resolve este problema fornecendo classes wrapper. As classes wrapper fazem parte da biblioteca padrão do Java java.lang , que converte os tipos de dados primitivos em um objeto. O diagrama a seguir mostra uma visão hierárquica dessas classes de wrapper:

WRAPPER CLASS IN JAVA



A tabela a seguir mostra todos os tipos de dados numéricos com sua classe wrapper correspondente:

Tipos de dados numéricos	Wrapper class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

{ Métodos

A classe Number vem com vários métodos que são úteis para realizar operações em números.

• **typeValue:** Converte o valor do objeto da classe Number para o tipo de dados de número primitivo especificado e o retorna. Aqui o type representa tipos de dados de números primitivos: byte, short, int, long, float, double.

• **compareTo:** Compara o argumento de entrada com o objeto Number. Retorna 1 (número positivo) se o valor do objeto Number for maior que o argumento, -1 (número negativo) se for menor que o valor do argumento e 0 se for igual ao argumento. Mas tanto o argumento quanto o número devem ser do mesmo tipo, então somente eles podem ser comparados. O tipo de referência pode ser um byte, double, float, long ou short.

• **equals:** Verifica se o objeto Number é igual ao argumento (também de Number Type). Tanto o objeto Number quanto o argumento podem ser de tipos diferentes. Retorna verdadeiro se os valores forem iguais, caso contrário, retorna falso.

• **parseInt:** Ao trabalhar com strings, às vezes precisamos converter um número representado na forma de string em um tipo inteiro. O método parseInt() é geralmente usado para converter String para Integer em Java. Também podemos passar o argumento radix neste método que representa o tipo decimal, octal ou hexadecimal como saída. Este método retorna o tipo de dados primitivo de uma String.

• **toString:** É usado para obter a representação do objeto String de qualquer objeto Number. Esse método recebe um tipo de dados primitivo como argumento e retorna um objeto String que representa o valor do tipo de dados primitivo. Existem três variantes deste método: toBinaryString(int i) toHexString(int i) toOctalString(int i).

• **IntergerValueOf:** Converte o valor de um argumento no objeto Number. O argumento pode ser qualquer tipo de dado numérico primitivo ou String. É um método estático que pode ser chamado diretamente através da classe Integer. O método pode receber dois argumentos, onde um é uma String ou um tipo de dado primitivo e o outro é uma raiz.

{ Strings

O tipo String trata-se de uma sequência de caracteres que tem o papel de representar um texto, e possui uma série de características.

Sempre iniciado com o S maiúsculo, a String é imutável, nunca alterando seu valor durante o fluxo do código.

```
String texto = "Qualquer texto entre aspas é uma String";
```

Algumas das características das Strings são os métodos. Vamos conhecer alguns a seguir:

Para testar a igualdade entre duas Strings, podemos usar o método **equals**, que é presente em todas as classes que derivam da classe Object.

Ex.:

```
String nome1 = "Amanda";
String nome2 = "Renata";
if (nome1.equals(nome2)) {
    System.out.println("Os nomes são iguais!");
} else {
    System.out.println("Os nomes são diferentes!");
```

Também existe o método **equalsIgnoreCase**, com a finalidade de comparar a string ignorando se há letras maiúsculas e minúsculas.

O método **valueOf** recebe um argumento de qualquer tipo e converte em um objeto que é do tipo string.

```java
// Declarar algumas variáveis de tipos primitivos
boolean valorBooleano = false;
int valorInteiro = 13;
char valorCaractere = 'F';
float valorFloat = 3.14f;
double valorDouble = 1.61803;

// Verificar os valores das variáveis
System.out.printf("Booleano: %s%n",
String.valueOf(valorBooleano));
System.out.printf("Inteiro: %s%n",
String.valueOf(valorInteiro));
System.out.printf("Caractere: %s%n",
String.valueOf(valorCaractere));
System.out.printf("Float: %s%n",
String.valueOf(valorFloat));
System.out.printf("Double: %s%n",
String.valueOf(valorDouble));
```

Outro método muito utilizado é o **compareTo.** Ele retorna em forma de número inteiro as respostas. Por exemplo: se as Strings forem iguais, retorna 0. Caso a comparação seja menor, ele retorna um número negativo, e caso for maior, retorna um número positivo.

Outros métodos de uma String estão listados a seguir:

• **Concat:** Existem duas formas de unir dois ou mais textos. Um deles é usando o operador de concatenação representado por +, e o outro é utilizando o método concat.

• **Length:** Retorna o tamanho do texto.

• **CharAt:** Retorna a localização de um caractere específico. O parâmetro do método é um inteiro, que é usado como índice, e o caractere que será retornado é o que consta nessa posição.

• **IndexOf:** Localiza a primeira ocorrência de um caractere na string.

• **LastOf:** Localiza a última ocorrência de um caractere.

• **Substring:** Permite copiar uma parte da string original, transformando em uma nova string.

- **Replace:** Faz a substituição de algum caractere na string por outro.
- **ToUpperCase:** Converte todos os caracteres para letras maiúsculas.
- **ToLowerCase:** Converte todos os caracteres para letras minúsculas.
- **Trim:** Remove os espaços em brancos existentes no início e final da string.

{ Arrays

Arrays são conhecidos também como matrizes em Java e fazem parte do pacote java.util. São objetos que têm um número fixo de valores, todos com um único tipo e com o seu comprimento fixo, determinado já quando criado.

Cada item dentro de um array são elementos e esses elementos são representados pela palavra índice. Todos os arrays sempre começam com o índice 0, e assim por diante.

A seguir é possível observar um exemplo de array:

3	27	16	42	88
0	1	2	3	4

Onde os números 3, 27, 16, 42 e 88 são os itens de dentro do Array, e cada item tem uma posição, por exemplo, o número 3 começa na posição 0, e se quisermos saber em que posição o 42 está, é retornado o índice 3 do deste Array.

Para declarar um array, existem algumas maneiras diferentes. O primeiro passo é definir o tipo, depois nomear o array e definir a quantidade de índices acrescentando esse valor dentro de colchetes [ ]. É possível, também, passar essa quantidade junto com o tipo.

Ex.:

```
tipo meuArray[];
ou
tipo[] meuArray;
```

É possível também declarar com a palavra reservada **new:**

```
meuArray = new tipoDado [tamanho];
```

Quando já sabemos quais são os itens e o tamanho do array, podemos declarar toda essa informação:

```
int[] numeros = {5, 9, 12, 3, 4};
```

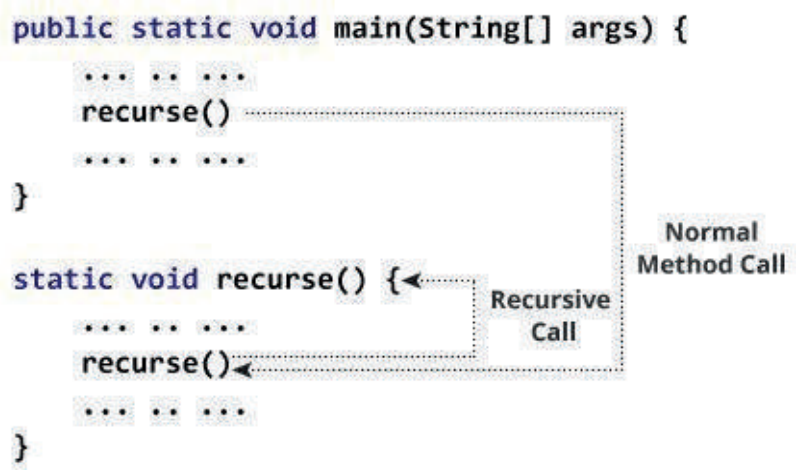
O retorno deste array será:

5	9	12	3	4
0	1	2	3	4

Para descobrir o tamanho de um array, também podemos utilizar o método **length.**

{ Recursão

A recursão é a técnica de fazer uma função chamar ela mesma. Esta técnica fornece uma maneira de quebrar problemas complicados em problemas simples que são mais fáceis de resolver.



No exemplo acima, chamamos o recurse() método de dentro do método “main”. E, dentro do método recurse(), estamos novamente chamando o mesmo método recurse. Esta é uma chamada recursiva.

Para interromper a chamada recursiva, precisamos fornecer algumas condições dentro do método. Caso contrário, o método será chamado infinitamente. Portanto, usamos a instrução if...else (ou abordagem semelhante) para encerrar a chamada recursiva dentro do método.

Qual é a diferença entre recursão direta e indireta?

Uma função fun é chamada de recursiva direta se chama a mesma função fun. Uma função fun é chamada de recursiva indireta se ela chama outra função, digamos fun_new e fun_new chama fun direta ou indiretamente.

Recursão direta	Recursão indireta

void directRecFun(){
// Algum código....
directRecFun();
// Algum código...
}
	void indiretoRecFun1(){
	// Algum código...
	indiretoRecFun2();
	// Algum código...
	}
	void indiretoRecFun2(){
	// Algum código...
	indiretoRecFun1();
	// Algum código...
	}

Como a memória é alocada para diferentes chamadas de função na recursão?

Quando qualquer função é chamada de main(), a memória é alocada a ela na pilha. Uma função recursiva chama a si mesma, a memória para a função chamada é alocada em cima da memória alocada para a função de chamada e uma cópia diferente das variáveis locais é criada para cada chamada de função. Quando o caso base é atingido, a função retorna seu valor para a função pela qual é chamada e a memória é desalocada e o processo continua.

Existem algumas vantagens e desvantagens quanto ao uso de funções recursivas.

Vantagens	Desvantagens
• Maneira limpa e simples de escrever código.	• Maiores requisitos de espaço do que o programa iterativo, pois todas as funções permanecerão na pilha até que o caso base seja alcançado; • Maiores requisitos de tempo devido a chamadas de função e sobrecarga de retorno.

Referências

https://towardsdatascience.com/top-10-libraries-every-java-developer-should-know-37dd136dff54

https://www.w3schools.com/java/java_data_types_numbers.asp

https://www.devmedia.com.br/string-em-java-entendendo-e-utilizando-essa-classe/25503

https://www.devmedia.com.br/trabalhando-com-arrays-em-java/25530

https://www.guj.com.br/t/entendendo-recursao/93595

Google olabi

<ESTRUTURADEDADOS EMJAVA>

/exercícios/

1 ➔ Modele uma conta. A ideia nesse momento é apenas modelar, isto é, identificar quais informações são importantes. Desenhe no papel tudo o que uma Conta tem e tudo o que ela faz. Ela deve ter o nome do titular (String), o número (int), a agência (String), o saldo (double) e uma data de abertura (String). Além disso, ela deve fazer as seguintes ações: saca, para retirar um valor do saldo; deposita, para adicionar um valor ao saldo; calculaRendimento para devolver o rendimento mensal dessa conta, que é de 10% sobre o saldo.

2 ➔ Escreva um main Java que solicita 8 inteiros ao usuário e guarda esses valores em um array. Depois, o programa deve descobrir e exibir qual a posição do elemento de maior valor.

3 ➔ Faça um programa (utilizando recursividade) que peça para o usuário digitar um número. Em seguida, faça a soma de todos os algarismos do número.

Exemplos:

1111 = 1+1+1+1 = 4
2090 = 2 + 0 + 9 + 0 = 11

4 ➔ Faça um programa que utilize duas funções: uma iterativa e uma recursiva que recebam um valor n passado pelo usuário, e imprima em contagem regressiva a partir deste valor. Exemplo: se o usuário passar o valor 5, o programa imprimirá 5, 4, 3, 2, 1, 0.

As funções devem ser executadas seguidamente e seus resultados exibidos de forma identificada.