Atividade 05 – PESQUISA: Classes Wrappers

➔ **Respondas as questões abaixo e poste suas respostas no Moodle em um arquivo PDF.**

1. Na tecnologia Java, defina o que são e para que servem as Classes Wrappers.

As classes wrappers servem para usar tipos de dados primitivos como objetos.

2. De forma **sucinta**, comente a aplicação das seguintes classes:

   a. Integer: pode verificar qual o mínimo ou o máximo entre dois valores inteiros, verificar quantos números de bits zero seguindo o bit de menor ordem, etc.

   b. Boolean: permite operações lógicas com métodos utilitários: **logicalAnd** retorna true se ambos os operandos forem true; **logicalOr** retorna true se ao menos um operando for true; e **logicalXor** retorna true apenas se os operandos forem diferentes. Esses métodos melhoram a clareza em expressões booleanas complexas.

   c. Character: facilita a manipulação de caracteres com métodos úteis: **isAlphabetic** verifica se um caractere é alfabético, útil para validações de texto; **isDigit** identifica se o caractere é um dígito, comum em validações numéricas; e **isWhitespace** indica se o caractere é um espaço em branco, ideal para formatações e limpeza de entradas de texto. Esses métodos ajudam a garantir que o processamento de caracteres siga critérios específicos de maneira prática e eficiente.

   d. Double: fornece métodos para manipulação avançada de valores de ponto flutuante: **isInfinite** verifica se um valor Double é infinito, útil para evitar operações com resultados indefinidos; **toHexString** converte um valor double em uma representação hexadecimal, facilitando visualizações específicas ou compatibilidade com sistemas de baixa precisão.

   e. Byte: facilita a manipulação de números inteiros pequenos (de -128 a 127) com métodos úteis: **decode** converte uma String em um valor Byte, útil para interpretar entradas de texto como números; **compareTo** compara dois objetos Byte numericamente, retornando um valor que indica sua ordem, o que é prático em ordenações e comparações diretas.

   f. Short: é útil para manipular números inteiros pequenos (de -32.768 a 32.767) com métodos específicos: **compare** compara dois valores short numericamente, retornando um valor que indica a ordem relativa, ideal para ordenações; **toUnsignedInt** converte um valor short em int sem sinal, expandindo a faixa de representação para valores positivos.

g. Float: permite manipulações específicas de valores de ponto flutuante de precisão simples: **floatToIntBits** retorna a representação em bits de um valor float seguindo o padrão IEEE 754, útil para análises e armazenamento binário; **isFinite** verifica se o valor float é finito, retornando false para NaN e infinitos, o que ajuda a evitar cálculos inválidos.

h. Long: oferece métodos para manipulação de números inteiros grandes (de -2^63 a 2^63-1): **decode** converte uma String em um valor Long, útil para interpretar entradas de texto como números inteiros grandes; **divideUnsigned** retorna o quociente sem sinal ao dividir dois valores long, interpretando-os como valores sem sinal, o que é útil para trabalhar com números grandes e operações de divisão sem considerar sinais.

Duas constantes muito interessantes dos tipos numéricos são MIN_VALUE e MAX_VALUE, que são úteis para representar os limites mínimo e máximo que um tipo numérico pode armazenar na linguagem Java.

3. Considere as classes citadas no item “2”. Escolha 3 destas classes e construa um código simples que mostre um exemplo de sua aplicação para elas. Copie e cole o código de teste abaixo.

```
public class Calculator {
    Integer a;
    Integer b;
    Boolean isDivisible;

    Calculator(Integer a, Integer b) {
        if (Integer.valueOf(b) == 0) {
            throw new IllegalArgumentException("Impossível dividir por
0");
        }
        this.a = a;
        this.b = b;
        this.isDivisible = Integer.valueOf(a) % Integer.valueOf(b) ==
0;
    }

    public String printResult() {
        if (Boolean.TRUE.equals(isDivisible)) return String.format("%s
é divisível por %s", this.a, this.b);
        else return String.format("%s não é divisível por %s", this.a,
this.b);
    }

    public static void main(String[] args) {
        Calculator calculator = new Calculator(10, 0);
        System.out.println(calculator.printResult());
    }
}
```