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 são um conjunto de classes que fornecem um invólucro (wrapper) para os tipos primitivos da linguagem. Cada tipo primitivo possui sua respectiva classe wrapper associada, permitindo que valores primitivos sejam tratados como objetos. As classes wrappers estão localizadas no pacote java.lang e são automaticamente importadas em programas Java, o que significa que não é necessário importá-las explicitamente.

As principais classes wrappers e seus respectivos tipos primitivos são:

- Integer: Invólucro para o tipo primitivo int.
- Long: Invólucro para o tipo primitivo long.
- Double: Invólucro para o tipo primitivo double.
- Float: Invólucro para o tipo primitivo float.
- Boolean: Invólucro para o tipo primitivo boolean.
- Byte: Invólucro para o tipo primitivo byte.
- Character: Invólucro para o tipo primitivo char.
- Short: Invólucro para o tipo primitivo short.

As classes wrappers são utilizadas para diversos propósitos em programas Java:

Tratamento de Valores Primitivos como Objetos: As classes wrappers permitem que valores primitivos sejam usados em contextos que requerem objetos, como coleções genéricas (por exemplo, List<Integer>) e algumas bibliotecas que operam exclusivamente com objetos.

Conversão entre Tipos Primitivos e Objetos: As classes wrappers facilitam a conversão entre valores primitivos e objetos. Esse processo é chamado de "autoboxing" quando um valor primitivo é convertido automaticamente para o correspondente wrapper, e "unboxing" quando um objeto wrapper é convertido automaticamente para o correspondente tipo primitivo.

Manipulação e Operações com Dados: As classes wrappers oferecem métodos úteis para realizar operações específicas com cada tipo de dado. Por exemplo, Integer possui métodos para converter strings em valores inteiros (parseInt()), para realizar cálculos matemáticos (add(), subtract(), etc.) e para converter inteiros em strings (toString()).

Representação de Valores Nulos: As classes wrappers permitem que variáveis de tipos primitivos tenham o valor nulo (null). Isso é útil em situações em que é necessário representar a ausência de um valor.

Reflection e Serialização: As classes wrappers são amplamente utilizadas em reflexão (reflection) e serialização, onde objetos precisam ser examinados ou transformados em fluxos de bytes.

Em resumo, as Classes Wrappers são classes que encapsulam os tipos primitivos, permitindo que sejam tratados como objetos e fornecendo métodos e funcionalidades adicionais para trabalhar com esses dados. Seu uso é comum em situações que envolvem coleções de dados, conversão de tipos, reflexão, serialização e outras operações que requerem o tratamento de tipos primitivos como objetos.

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

    a. Integer;
    A classe Integer é utilizada para representar números inteiros em Java. Ela é amplamente aplicada em situações onde valores inteiros precisam ser tratados como objetos, como em coleções genéricas, algoritmos que operam em objetos e em APIs que requerem valores inteiros em forma de objeto.

    b. Boolean;
    A classe Boolean é utilizada para representar valores booleanos, ou seja, verdadeiro (true) ou falso (false). Ela é comumente aplicada em estruturas de controle condicional, como if e while, para representar condições lógicas. Também é útil em APIs que requerem valores booleanos em forma de objeto.

    c. Character;
    A classe Character é utilizada para representar caracteres individuais em Java. Ela é amplamente aplicada em operações que envolvem caracteres, como manipulação de strings, análise de texto e formatação.

    d. Double;
    A classe Double é utilizada para representar números de ponto flutuante de precisão dupla em Java. Ela é aplicada em cálculos matemáticos mais precisos e em situações onde números decimais requerem tratamento como objetos.

    e. Byte;
    A classe Byte é utilizada para representar números inteiros de 8 bits em Java. Ela é frequentemente aplicada em operações de baixo nível, manipulação de arquivos binários e quando é necessário representar um valor numérico compacto.

    f. Short;
    A classe Short é utilizada para representar números inteiros de 16 bits em Java. Assim como Byte, ela também é aplicada em operações de baixo nível, mas com uma faixa maior de valores possíveis.

    g. Float;
    A classe Float é utilizada para representar números de ponto flutuante de precisão simples em Java. Embora menos comum do que Double, ela é aplicada em situações que requerem menor consumo de memória e quando a precisão dupla não é necessária.

    h. Long.
    A classe Long é utilizada para representar números inteiros longos em Java. Ela é aplicada em situações onde números inteiros muito grandes precisam ser tratados como objetos, como timestamps ou IDs extensos.

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.

   Obs.: Não é preciso fazer exemplos complexos; todos eles podem ser feitos dentro de um mesmo método main.

```
public class ExemploClassesWrappers {
   public static void main(String[] args) {
      // Exemplo com a classe Integer
      Integer numeroInteiro = 42;
      System.out.println("Exemplo com Integer:");
      System.out.println("Valor inteiro: " + numeroInteiro);
      System.out.println("Valor em dobro: " + numeroInteiro * 2);

      // Exemplo com a classe Boolean
      Boolean valorVerdadeiro = true;
      Boolean valorFalso = false;
      System.out.println("\nExemplo com Boolean:");
      System.out.println("Valor verdadeiro: " + valorVerdadeiro);
      System.out.println("Valor falso: " + valorFalso);

      // Exemplo com a classe Float
      Float numeroDecimal = 3.14f;
      System.out.println("\nExemplo com Float:");
      System.out.println("Valor decimal: " + numeroDecimal);
      System.out.println("Valor multiplicado por 2: " + numeroDecimal * 2
   }

}
```