Switch case

Atividade 1

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner (System.***in***);

System.***out***.println("Selecione uma das opções:");

System.***out***.println("1 - Se você ama Java");

System.***out***.println("2 - Se você não gosta de Java");

System.***out***.println("3 - Se entre estudar Java e PHP, você prefere dormir");

String opcao = sc.nextLine();

**switch** (opcao) {

**case** "1":

System.***out***.println("Parabéns, Java é a melhor linguagem.");

**break**;

**case** "2":

System.***out***.println("Eu não gosto de quem não gosta de Java.");

**break**;

**case** "3":

System.***out***.println("Eu te entendo, mas Java ainda é melhor.");

**break**;

**default**:

System.***out***.println("Não existe a opção " + opcao);

**break**;

}

sc.close();

}

}

Atividade 2

Embora **switch-case** e **if-else** sejam estruturas condicionais em linguagens de programação, elas apresentam diferenças importantes em termos de funcionalidade, legibilidade e performance. Compreender essas distinções é crucial para escolher a ferramenta ideal para cada situação e escrever código mais eficiente e organizado.

**Funcionalidade:**

* **Switch-Case:**
  + Projetado para lidar com **várias comparações com um mesmo valor**.
  + Cada **case** representa um valor possível para a variável de comparação.
  + O código correspondente é executado quando há uma correspondência.
  + A instrução **break** é essencial para finalizar a execução do bloco e evitar a ativação de casos subsequentes.
* **If-Else:**
  + Mais flexível para **comparações complexas e encadeadas**.
  + Permite testar diversas condições booleanas e executar diferentes blocos de código em cada caso.
  + Suporta o uso de operadores lógicos (&&, ||) para combinar condições de forma mais precisa.

**Analogia:**

* **Switch-Case:** Imagine um menu com diversas opções. Cada opção representa um "case", e o código correspondente é a ação a ser executada quando a opção é selecionada.
* **If-Else:** Imagine uma série de perguntas e respostas. Cada pergunta representa uma condição booleana, e as respostas são os blocos de código a serem executados com base na resposta dada.

**Legibilidade:**

* **Switch-Case:**
  + Pode ser mais **conciso e legível** para **casos simples com muitos valores possíveis**.
  + Apresenta uma estrutura clara e organizada com blocos de código agrupados por valor, facilitando a visualização e compreensão do código.
* **If-Else:**
  + Em **cenários com muitas condições aninhadas**, o **if-else** pode se tornar **menos legível e confuso**.
  + A aninhação excessiva de blocos de código pode dificultar a compreensão do fluxo de lógica e a identificação de erros.

**Performance:**

* **Switch-Case:**
  + Geralmente oferece **melhor performance** para **casos com um número limitado de valores possíveis**.
  + A comparação é realizada através de uma tabela de hash, otimizando a busca pelo valor correspondente e acelerando a execução do código.
* **If-Else:**
  + Em **situações com muitas comparações complexas**, o **if-else** pode ter um **desempenho inferior**.
  + A avaliação das condições booleanas pode ser mais lenta, impactando negativamente na performance geral do código.

**Fatores a Considerar:**

* **Número de opções de comparação:** Switch-case ideal para poucas opções.
* **Complexidade das condições:** If-else adequado para condições booleanas complexas.
* **Prioridade:** Legibilidade, performance ou concisão do código?

**Recomendação:**

* **Switch-Case:**
  + Utilize quando tiver **comparações simples com um número limitado de valores possíveis**.
  + Priorize a legibilidade e organização do código para casos simples.
* **If-Else:**
  + Opte por **if-else** para **casos com condições booleanas complexas e encadeadas**.
  + Priorize a flexibilidade e expressividade para situações complexas.

Atividade 3

**Switch-case:**

* **Imagine:** Um menu com várias opções (como um painel de botões).
* **Perfeito para:** Muitas escolhas com base em um valor (ex: tipo de fruta).
* **Vantagens:** Código organizado, legível e rápido (às vezes).
* **Desvantagem:** Menos flexível.

**If-else:**

* **Imagine:** Uma série de perguntas e respostas (como um interrogatório).
* **Perfeito para:** Casos simples com lógica baseada em condições (ex: par ou ímpar).
* **Vantagens:** Mais flexível, compatível com diversas linguagens.
* **Desvantagem:** Código menos legível em projetos grandes.

**Escolha:**

* **Switch-case:** organização e clareza para muitas opções.
* **If-else:** simplicidade e flexibilidade para casos simples.