Conforme Deitel preconiza “Normalmente, instruções em um programa são executadas uma após a outra na ordem em que são escritas.” (pag. 83). Durante a década de 1960 muita dificuldade era encontrada devido a utilização da expressão “goto”, que permite ao programador redirecionar o fluxo de controle dentro do algoritmo de software. Foi a partir da pesquisa de Bohm e Jacopini que demonstrou que programas deveriam ser escritos utilizando três estruturas de controle: a estrutura de sequência, a estrutura de seleção e a estrutura de repetição, chamadas desde entao em Java de “instruções de controle”.

As estruturas de controle em Java são essenciais para criarmos programas dinâmicos e moldáveis, pois permitem que o fluxo de execução seja adaptado de acordo com diferentes condições e requisitos. Sem essas estruturas, os programas apenas executariam instruções de maneira linear e sequencial, limitando sua capacidade de reagir a cenários variados. Com as estruturas de controle, é possível construir algoritmos que tomam decisões, repetem tarefas e se ajustam conforme necessário, ampliando significativamente as possibilidades de uso e a complexidade que um programa pode alcançar. Em Java, temos três principais tipos de estruturas de controle: sequência, seleção e repetição.

1. Estrutura de Sequência

Segundo Deitel, “A estrutura de sequência está incorporada ao Java. A não ser que seja instruído de outra forma, o computador executa as instruções Java uma depois da outra na ordem em que elas são escritas — isto é, em sequência.”. (pag. 83)

Desta forma, a estrutura de sequência significa que a execução será linear, na qual o código é executado linha por linha, ou seja, na sequência de leitura do código.

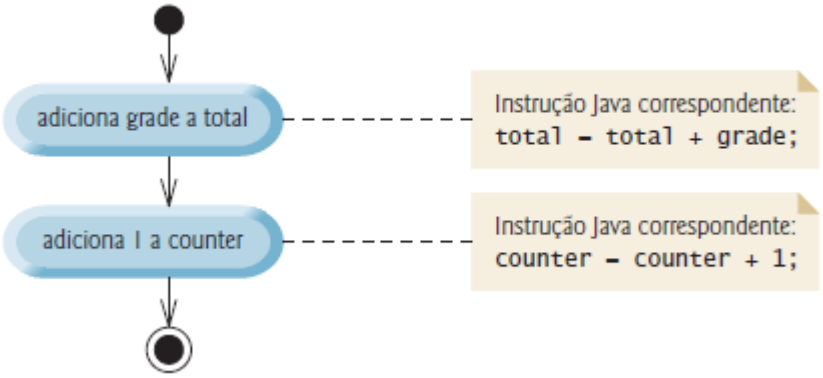

Fig. 01 – Diagrama de estrutura de sequência.

Fonte: Java como programar – pag. 83

No círculo sólido superior representa o estado inicial do algoritmo, o circulo solido circulado ao final representa ao fim do programa, ao centro as elipses representam as duas ações do código, a primeira adicionar grade ao total, que no contexto pode ser uma nota de um aluno e a outra elipse um contador que adiciona um a conter, uma única vez, que pode ser usado como o número de notas adicionadas, caso houvesse uma segunda contagem, o que aqui nessa estrutura simples não é o caso.

Essa estrutura é essencial para definir a ordem básica das operações no programa, mas, sozinha como vimos acima, não permite adaptação a condições ou a execução de blocos de código de maneira flexível. A sequência é a base para o controle de fluxo, mas é complementada pelas estruturas de seleção e repetição, que veremos a seguir, para alcançarmos instruções mais satisfatórias.

2. Estrutura de Seleção

Para Deitel, "O Java contém três tipos de instruções de seleção. A instrução if realiza uma ação (seleciona) se uma condição for verdadeira ou pula a ação se a condição for falsa. A instrução if...else realiza uma ação se uma condição for verdadeira e realiza uma ação diferente se a condição for falsa. A instrução de seleção switch realiza uma de muitas ações diferentes, dependendo do valor de uma expressão." (pag. 84).

As estruturas de seleção permitem ao programador inserir no código desvios de comportamento, no qual ao aparecer uma condição o algoritmo possa seguir direções diferentes dependendo da condição selecionada. Ou seja, o programa toma decisões com base em condições especificas.

Essas estruturas possibilitam a execução condicional de blocos de código, permitindo que o programa siga diferentes caminhos de execução dependendo de variáveis ou condições lógicas.

A instrução if realiza uma única verificação, desta forma é nomeada por Deitel como uma *instrução de seleção única*, que em seguida poderá executar um comando ou um bloco de comandos. A instrução if...else é conhecida como instrução de seleção dupla pois concede ao algoritmo duas opções de sequência. Se o if for verdadeiro executa o comando ou bloco de comando do if, caso contrário executará o comando ou bloco de comando do else. "Os programas utilizam instruções de seleção para escolher entre cursos alternativos de ações" como por exemplo no código:

```
if (nota >= 60)
 System.out.println("Aluno aprovado");
```

O programa determina que a nota do aluno tem que ser maior ou igual a 60, caso seja, o programa imprime "Aluno aprovado", caso contrario segue o fluxo e não executa a impressão.

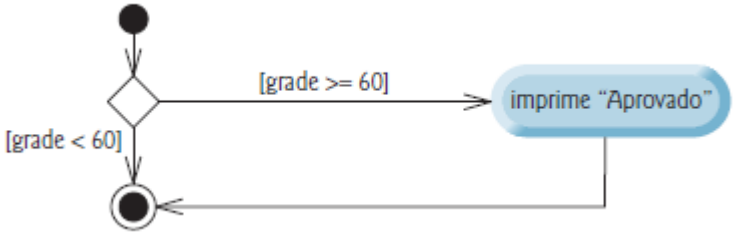

Figura 02: Diagrama de atividades UML de uma instrução de seleção única if.

Fonte: Java como programar - pag. 85

A estrutura if como já mencionado realiza somente uma ação se a condição for verdadeira, caso seja uma condição falsa o código pulo essa estrutura e não executa nada. Para solucionar essa questão é utilizada a estrutura if...else, muito parecido com o if simples porem caso a verificação constate que é false o código executará a condição ou bloco de condições da condição else.

```
if (nota >= 60)
 System.out.println("Aluno aprovado");
else

System.out.println("Aluno reprovado");
```

Nesse exemplo a condição nota for verdadeira (true) imprime "Aluno aprovado" caso a condição seja falsa(false) imprime "Aluno reprovado", claramente vemos 2 opções de fluxo do algoritmo diferentes.

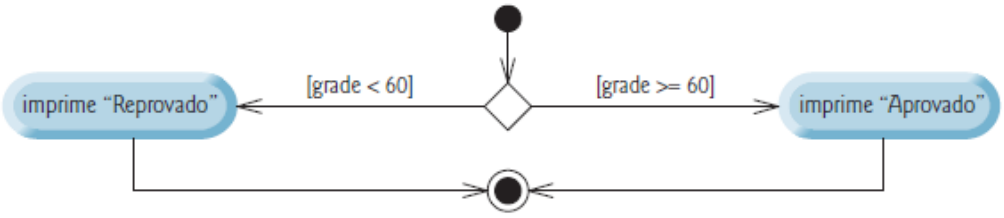

Figura 03: Diagrama de atividades da UML de instrução de seleção dupla if...else.

Fonte: Java como programar - pag. 86

Vale ressaltar que essa estrutura de seleção pode ser representada pelo chamado “operador ternário” que é uma outra forma de se escrever o if...else. Ele é representado pelo operador condicional (?;) como no exemplo a seguir, e segundo Deitel esse é o único operador ternário em Java.

```
System.out.println(nota>=60 ? “Aluno aprovado” : “Aluno reprovado”);
```

Desta forma, podemos dizer que as estruturas, if, else-if e else, verificam uma condição booleana que se for verdadeira, o bloco de código associado é executado; caso contrário, passa para a próxima condição ou bloco else.

“Um programa pode testar múltiplos casos colocando instruções if...else dentro de outras instruções if...else para criar instruções if...else aninhadas.” (Deitel, pag.86).

Como nos esclarece Deitel podemos testar várias alternativas de fluxo de condições com os operadores if..else, chamamos esta estrutura de instruções if...else aninhadas, darei um breve exemplo a seguir:

```
if (nota >= 80)
 System.out.println(“Aluno aprovado com louvor”);
else if (nota>=60)

System.out.println(“Aluno aprovado”);
else
System.out.println(“Aluno reprovado”);
```

Podemos escrever as estruturas de instruções if...else sem chaves {} ou com chaves indicando uma forma de escrever as instruções em blocos e tornando mais assertiva e visual ao programador o que de fato ele quer que seja resolvido em cada instrução.

```
if (nota >= 60){
        System.out.println(“Aluno aprovado”);
        System.out.println(“Parabéns você está apto para progredir no curso”);


}else{
        System.out.println(“Aluno reprovado”);
        System.out.println(“Você precisará refazer a matéria.”);
}
```

Outra estrutura de seleção pode ser utilizada com o comando Switch. Segundo Deitel, “A instrução switch é chamada de instrução de seleção múltipla, pois seleciona entre muitas ações diferentes (ou grupos de ações).” (pag. 84)

Switch é, desta forma, uma estrutura que permite a escolha entre múltiplas alternativas (case) com base no valor de uma variável. Cada valor potencial é associado a um case e o bloco correspondente é executado, quando o valor da variável coincide com um case específico. O switch é útil quando se deseja testar uma variável contra diversos valores possíveis, como em menus de um caixa de banco ou configurações diversas em um painel de instrumentos.

A estrutura de seleção é fundamental para a construção de lógica condicional, permitindo que o programa reaja a diferentes cenários e tome decisões de acordo com a entrada ou o estado atual.

3. Estrutura de Repetição

“O Java fornece três instruções de repetição (também chamadas instruções de loop) que permitem que programas executem instruções repetidamente contanto que uma condição (chamada condição de continuação do loop) permaneça verdadeira. As instruções de repetição são as instruções while, do...while e for.” (Deitel, pag. 84).

A estrutura de repetição permite que o programa execute um bloco de código várias vezes, o que é útil para processar listas, executar tarefas repetitivas ou iterar até que uma condição seja satisfeita. Em Java, como Deitel nos ensinou, existem três estruturas principais de repetição:

For: Permite a repetição de um bloco de código um número fixo de vezes, ou seja, é finito, geralmente usado quando o número de iterações é conhecido ou pode ser calculado antecipadamente. O for pode ser utilizado para percorrer arrays e listas de maneira controlada.

A forma de se escrever um for em Java é a seguinte:

for (<inicialização>; <teste condicional>; <incremento ou decremento>) {

<instruções>;

};

Inicialização é um comando de atribuição em que colocamos um valor inicial de verificação para o laço de repetição.

Teste condicional é uma expressão relacional que irá nos dizer quando esse laço irá acabar.

Variável de incremento ou decremento é uma variável de controle que irá variar cada vez que o laço for repetido. Exemplificando segue um exemplo de for:

```
int cont = 0;

for(cont=0;  cont<=10; cont++){

System.out.println(“Aluno aprovado com sucesso!”);

}
```

Esse código imprime a mensagem "Aluno aprovado com sucesso!" 11 vezes (de 0 a 10).

While: Executa o bloco de código enquanto uma condição booleana for verdadeira. É útil quando o número de repetições não é conhecido antecipadamente e depende de uma condição.

A forma geral do comando while é:

```
while (<condição>)

{

  <instruções>;

}
```

Segue o exemplo do for escrito com while.

```
int cont = 0;

while(cont<=10){

System.out.println(“Aluno aprovado com sucesso!”);

cont++;  //incrementa a variável cont em 1 cada vez que o laço se repetir

}
```

Observe que o laço irá repetir enquanto o contador for menor ou igual a 10, imprimindo 11 vezes tambem a frase “Aluno aprovado com sucesso”.

Do-while: Similar ao while, mas garante que o bloco de código seja executado pelo menos uma vez, pois a condição é verificada após a execução do bloco. Segue o exemplo:

```
int cont = 0;

do{

System.out.println(“Aluno aprovado com sucesso!”);

cont++;

}while(cont<=10)
```

Essas estruturas de repetição são essenciais para a execução eficiente de tarefas em série, tornando o código mais compacto e permitindo a automação de processos. Vale ressaltar que as estruturas while e for não executam a ação, ou seja, se a condição de execução inicial for falsa a ação não será executada, ou executam a ação uma ou mais vezes se a condição inicial for verdadeira. Do...while diferentemente das duas expressões anteriores irá executar pelo menos uma vez, mesmo se a condição de verificação inicial for falsa, o que o difere das anteriores. Então dizemos que as condições de repetição em Java e em C continuam enquanto a condição for verdadeira o que difere do algoritmo “repita” em portugol.

As estruturas de controle são fundamentais para a criação de programas interativos e adaptativos. Elas possibilitam a implementação de lógica condicional e de repetição, essenciais para resolver problemas de maneira eficiente e prática. Em Java, o uso dessas estruturas permite construir programas que não apenas executam uma sequência fixa de instruções, mas que são capazes de responder a dados de entrada, fazer escolhas e realizar operações de forma repetitiva, atendendo a diferentes requisitos e cenários de uso.

Assim como um código bem estruturado nos guia até a solução, nossas escolhas e decisões moldam o caminho para o sucesso. Com lógica, propósito e perseverança, alcançamos qualquer objetivo.

Referência

Deitel, Paul e Deitel, Harvey. Java: como programar; tradução Edson Furmankiewicz ; revisão técnica Fábio Luis Picelli Lucchini. -- 8. ed. -- São Paulo: Pearson Prentice Hall, 2010.

https://sae.unb.br/cae/conteudo/unbfga/oo/new_fluxoDeControle.html acesso em 30 de outubro de 2024