INE5603 Introdução à POO

Prof. A. G. Silva

28 de agosto de 2017

Comandos de decisão simples e compostas

Objetivos:

- Utilização de controles de decisão em algoritmos simples. Exemplo: levar ou não um guarda-chuva, baseado na verificação do clima?
- Instruções para escolha entre ações alternativas:
  - `if`
  - `if-else`
- Instrução para escolha de uma entre múltiplas opções:
  - `switch-case-default`

Estruturas de controle

- Execução sequencial: as instruções em um programa são executadas uma após a outra na ordem em que são escritas
- Transferência do controle: várias instruções Java permitem especificar que a próxima instrução a executar não seja necessariamente a próxima na sequência
- Bohm e Jacopini demonstraram que todos os programas poderiam ser escritos sem nenhuma instrução *goto*
- Todos os programas podem ser escritos em termos de apenas três tipos de estruturas de controle: estrutura de sequência, estrutura de seleção e estrutura de repetição

Estrutura de sequência

- A menos que instruído de outro modo, o computador executa instruções Java uma após a outra na ordem em que elas são escritas
- O diagrama de atividades na figura a seguir ilustra uma estrutura de sequência típica em que dois cálculos são realizados na ordem
- O Java permite ter o número de ações que você quiser em uma estrutura de sequência
- Em qualquer lugar que uma ação única pode ser colocada, podemos colocar várias ações em sequência

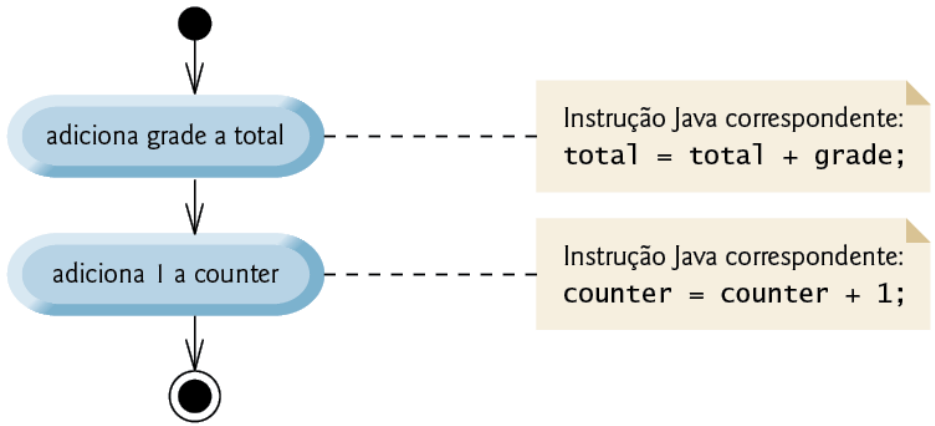

Diagrama de atividades da estrutura de sequência.

Estrutura de seleção

- Instrução `if`
  Executa uma ação, se uma condição é verdadeira; pula-a, se falsa
  Instrução de uma única seleção – seleciona ou ignora uma única ação (ou o grupo de ações).
- Instrução `if...else`
  Realiza uma ação se uma condição for verdadeira e realiza uma ação diferente se a condição for falsa
  Instrução de seleção dupla – seleciona entre duas ações diferentes (ou grupos de ações)
- Instrução `switch`
  Executa um de várias ações, com base no valor de uma expressão
  Instrução de seleção múltipla – seleciona entre muitas ações diferentes (ou grupos de ações)

Estrutura de repetição

- Três instruções de repetição (também chamadas instruções de laço ou de *loop*) executam instruções repetidamente enquanto condição de continuação de loop permanecer verdadeira
- As instruções `while` e `for` realizam a(s) ação(ões) no seu corpo zero ou mais vezes se a condição de continuação de loop for inicialmente falsa, o corpo não será executado
- A instrução `do...while` realiza a(s) ação(ões) no seu corpo uma ou mais vezes
- `if`, `else`, `switch`, `while`, `do` e `for` são palavras-chave

Instrução de seleção única – if

```
if (expressao logica)
    comando ou bloco
```

- Exemplo de pseudocódigo:

```
If (Se) a nota do aluno é maior que ou igual a 60
    Imprime "Aprovado"
```

- Exemplo em Java:

```
if ( grade >= 60 )
    System.out.println( "Aprovado" );
```

- Exemplo em UML:

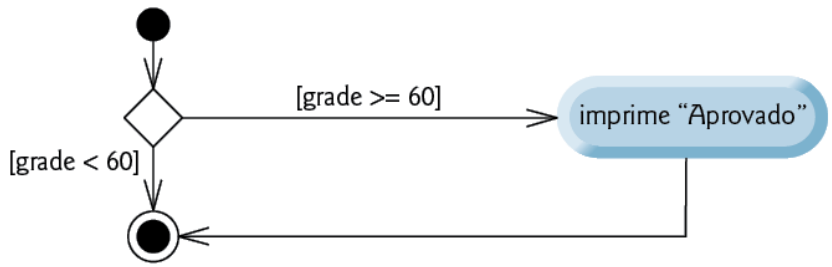

Diagrama de atividades UML de uma instrução de seleção única `if`.

Instrução de seleção dupla – if...else

```
if (expressao logica)
    comando 1 ou bloco 1
else
    comando 2 ou bloco 2
```

- Exemplo de pseudocódigo:

```
Se (if) a nota do aluno for maior que ou igual a 60
    Imprima "Aprovado"
Caso contrário (else)
    Imprima "Reprovado"
```

- Exemplo em Java:

```
if ( grade >= 60 )
    System.out.println( "Aprovado" );
else
    System.out.println( "Reprovado" );
```

Instrução de seleção dupla – if...else

- Exemplo em UML:

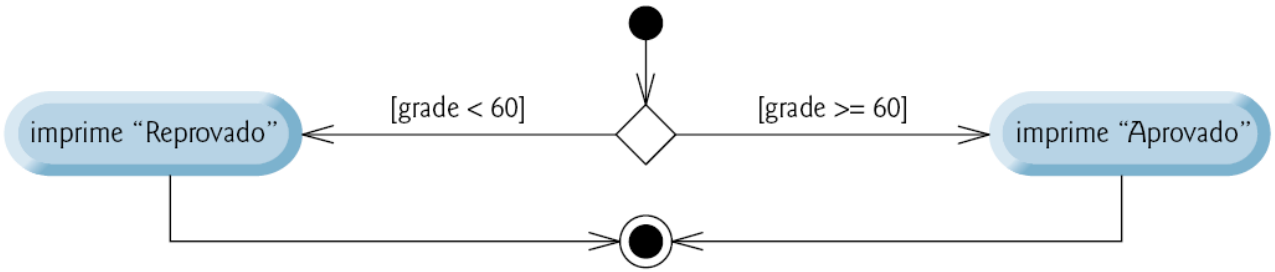

Diagrama de atividades da UML de instrução de seleção dupla `if...else`.

Operador condicional (?:)

- Trata-se de um operador ternário (aceita três operandos) funcionando como abreviatura de `if...else`
- Operandos e `?:` lançam uma expressão condicional
- O operando à esquerda do `?` é uma expressão booleana – que é avaliada como um valor boolean (true ou false)
- O segundo operando (entre o `?` e `:`) é o valor se a expressão booleana for verdadeira
- O terceiro operando (à direita de `:`) é o valor se a a expressão booleana for falsa
- Exemplo:

```
System.out.println(
grade >= 60 ? "Aprovado" : "Reprovado" );
```

Resulta na string "Aprovado" se a expressão booleana `grade >= 60` for verdadeira e na string "Reprovado" se for falsa

Boas práticas de programação

- Recue as duas instruções do corpo de uma instrução `if...else`
- Se existem vários níveis de recuo, cada nível deve ser recuado pela mesma quantidade adicional de espaço
- As instruções condicionais são mais difíceis de ler que as instruções `if...else` e devem ser utilizadas para substituir somente `if...else` simples que escolhem entre dois valores

Aninhamento

- Pode testar múltiplos casos colocando instruções `if...else` dentro de outras instruções `if...else` para criar instruções `if...else` aninhadas
- Pseudocódigo:

```
If (Se) a nota do aluno é maior que ou igual a 90
  Imprima "A"
caso contrário
  If (Se) a nota do aluno é maior que ou igual a 80
    Imprima "B"
  caso contrário
    If (Se) a nota do aluno é maior que ou igual a 70
      Imprima "C"
    caso contrário
      If (Se) a nota do aluno é maior que ou igual a 60
        Imprima "D"
      caso contrário
        Imprima "F"
```

Aninhamento

- Em Java:

```
if ( grade >= 90 )
  System.out.println( "A" );
else
  if ( grade >= 80 )
    System.out.println( "B" );
  else
    if ( grade >= 70 )
      System.out.println( "C" );
    else
      if ( grade >= 60 )
        System.out.println( "D" );
      else
        System.out.println( "F" );
```

Aninhamento

- Em Java (estilo preferido de endentação):

```
if ( grade >= 90 )
    System.out.println( "A" );
else if ( grade >= 80 )
    System.out.println( "B" );
else if ( grade >= 70 )
    System.out.println( "C" );
else if ( grade >= 60 )
    System.out.println( "D" );
else
    System.out.println( "F" );
```

- As duas formas são idênticas, exceto quanto ao espaçamento e recuo, que o compilador ignora

Cuidados

- O compilador Java sempre associa um else à instrução if imediatamente precedente, a menos que instruído de outro modo pela colocação de chaves (and).
- O seguinte código não é o que aparece:

```
if ( x > 5 )
    if ( y > 5 )
        System.out.println( "x and y are > 5" );
else
    System.out.println( "x is <= 5" );
```

- Cuidado! Essa instrução if...else aninhada não é executada como parece. Na verdade, o compilador interpreta a instrução como

```
if ( x > 5 )
    if ( y > 5 )
        System.out.println( "x and y are > 5" );
    else
    System.out.println( "x is <= 5" );
```

Cuidados

- Para forçar a instrução if...else aninhada a executar como foi originalmente concebida, devemos escrevê-la da seguinte maneira:

```
if ( x > 5 )
{
     if ( y > 5 )
          System.out.println( "x and y are > 5" );
}
else
     System.out.println( "x is <= 5" );
```

- As chaves indicam que o segundo if está no corpo do primeiro e que o else está associado com o primeiro if

Blocos

- A instrução if normalmente espera somente uma instrução no seu corpo.
- Para incluir várias instruções no corpo de um if (ou no corpo de um else de uma instrução iif...else), inclua as instruções dentro de chaves.
- As instruções contidas em um par de chaves formam um bloco.
- Um bloco pode ser colocado em qualquer lugar em que uma instrução individual pode ser colocada.
- Exemplo: Um bloco na parte else de uma instrução if...else

```
if ( grade >= 60 )
    System.out.println( "Aprovado" );
else
{
    System.out.println( "Reprovado" );
    System.out.println( "Voce deve fazer este curso
        novamente" );
}
```

Comandos de seleção – switch

- Pode ser efetuar seleção de uma entre múltiplas opções com vários **if**s ou **switch** no caso da expressão ser inteira (exceto tipo *long*) ou *char*
- A instrução switch consiste em um bloco que contém uma sequência de rótulos case e um case default
- O programa avalia a expressão de controle entre os parênteses que se seguem à palavra-chave switch
- O programa compara o valor da expressão controladora (que deve ser avaliada como um valor integral do tipo byte, char, short ou int) com cada rótulo case

Comandos de seleção – switch

- Se ocorrer uma correspondência, o programa executará as instruções para esse case
- A instrução break faz com que o controle do programa prossiga para a primeira instrução depois do switch
- switch não fornece um mecanismo para testar intervalos de valores – cada valor que deve ser testado deve ser listado em um rótulo case separado
- Observe que cada case pode ter múltiplas instruções
- switch difere de outras instruções de controle porque não exige que as múltiplas instruções em um case estejam entre chaves

Comandos de seleção – switch

- Sem um break, as instruções para um caso correspondente e casos subsequentes são executadas até que uma instrução break ou o fim do switch seja encontrado – isso é chamado de "*falling through*"
- Se não ocorrer nenhuma correspondência entre o valor da expressão controladora e um rótulo case, o caso default opcional é executado
- Se não ocorrer nenhuma correspondência e não houver um caso default, o controle de programa simplesmente continua com a primeira instrução depois do switch
- Esquecer uma instrução break, quando esta for necessária em um switch, é um erro de lógica

Comandos de seleção – switch

- A figura a seguir mostra o diagrama de atividades UML para a instrução switch geral
- A maioria das instruções switch usa um break em cada case para terminar a instrução depois de processar o case
- A instrução break não é necessária para o último case (ou o case default opcional, quando ele aparece por último), porque a execução continua com a próxima instrução depois do switch
- É interessante introduzir um case default para processamento de condições excepcionais (este normalmente é listado por último, tornando o break desnecessário)

Comandos de seleção – switch

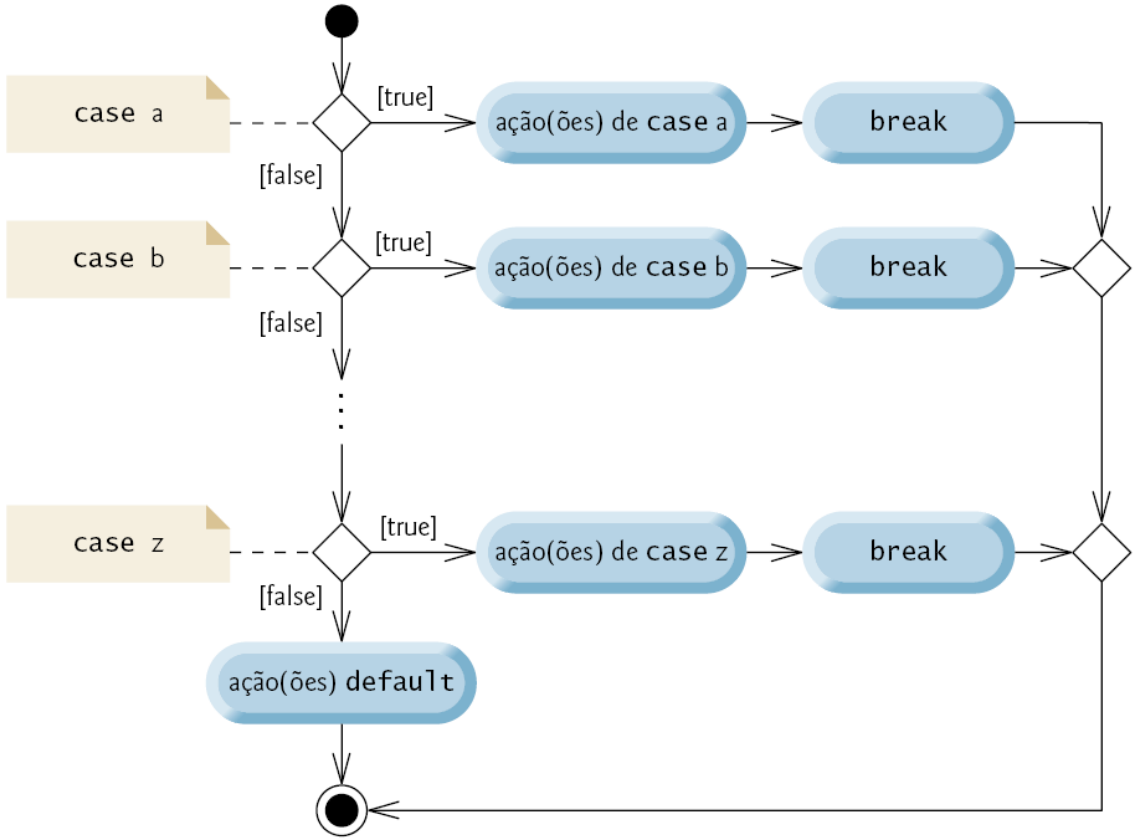

Diagrama de atividades UML de instrução de seleção múltipla `switch` com instruções `break`.

Comandos de seleção – switch

- Exemplo:

```
char conceito = 'B';
int aprov = 0,  reprov = 0;
String mensagem;
switch (conceito) {
    case 'A' : mensagem = "Excelente";
               aprov++;
               break;
    case 'B' : mensagem = "Bom";
               aprov++;
               break;
    case 'C' : mensagem = "Regular";
               aprov++;
               break;
    default  : mensagem = "Insuficiente";
               reprov++;
}
```

Comandos de seleção – switch

- Ao utilizar a instrução switch, lembre-se de que cada case deve conter uma expressão integral constante
- Uma constante integral é simplesmente um valor inteiro
- Além disso, você pode utilizar constantes de caractere – caracteres específicos entre aspas simples, como 'A', '7' or '$' – que representam os valores inteiros dos caracteres
- A expressão em cada case também pode ser uma variável constante – uma variável que contém um valor que não muda no programa inteiro – essa variável é declarada com a palavra-chave final
- O Java tem um recurso chamado enumerações – constantes de enumeração também podem ser utilizadas em rótulos case

Recomendações

- Erros de sintaxe (por exemplo, quando não é colocada uma das chaves em um bloco do programa) são capturados pelo compilador.
- Um erro de lógica (por exemplo, quando não colocadas as duas chaves em um bloco do programa) são capturadas em tempo de execução.
- Um erro fatal de lógica faz com que um programa falhe e finalize prematuramente.
- Um erro não fatal de lógica permite que um programa continue a executar, mas faz com que produza resultados incorretos.
- Assim como um bloco pode ser colocado em qualquer lugar em que uma instrução individual pode ser colocada, também é possível ter uma instrução vazia
- A instrução vazia é representada colocando um ponto e vírgula (;) no qual normalmente estaria uma instrução