UNIP – Universidade Paulista
Ciência da Computação

Aula 4
Estruturas de Controle

Prof. Célio Ricardo Castelano

Tópicos Abordados

1) Instruções de seleção em Java:

2) Instruções de repetição em Java:

Instruções de Seleção

- Instrução `if`:
  - Instrução de uma única seleção.
- Instrução `if...else`:
  - Instrução de seleção dupla.
- Instrução `switch`:
  - Instrução de seleção múltipla.

Instruções de Repetição

- Também conhecidas como *instruções de loop*.
- Realizam repetidamente uma ação enquanto a condição de continuação do loop permanecer verdadeira.
- Instrução `while`:
  - Realiza as ações no seu corpo zero ou mais vezes.
- Instrução `do…while`:
  - Realiza as ações no seu corpo uma ou mais vezes.
- Instrução `for`:
  - Realiza as ações no seu corpo zero ou mais vezes.

A instrução de uma única seleção `if`

- Instruções `if`:
  - Executam uma ação se a condição especificada for verdadeira.
  - Podem ser representadas por um símbolo de decisão (losango) em um diagrama de atividades UML.
    - Setas de transição fora de um símbolo de decisão têm condições de guarda.
      - O fluxo de trabalho segue a seta de transição cuja condição de guarda é verdadeira.

Operadores de Incremento/Decremento

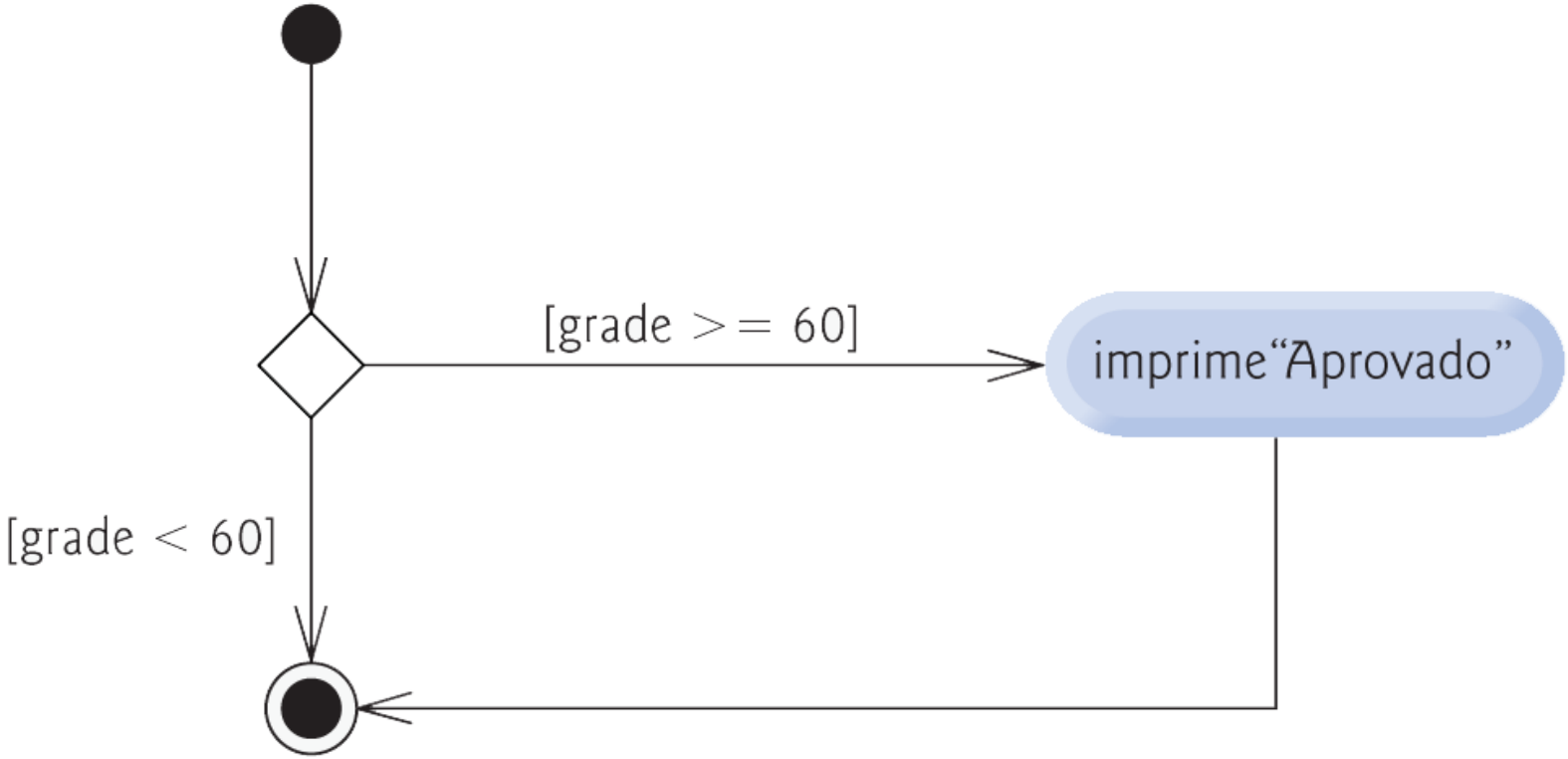

Figura 3.1 - Diagrama de atividades da UML de uma instrução de seleção única `if`.

A instrução de seleção dupla `if...else`

- Instrução `if...else`:
  - Executa uma ação se a condição especificada for `true` ou uma ação diferente se a condição especificada for `false`.
- Operador condicional (`? :`):
  - Único *operador ternário* do Java (recebe três operandos).
  - `? :` e seus três operandos formam uma *expressão condicional*.
    - Toda uma expressão condicional é avaliada para o segundo operando se o primeiro operando for `true`.
    - Toda uma expressão condicional é avaliada para o terceiro operando se o primeiro operando for `false`.

A instrução de seleção dupla `if...else`

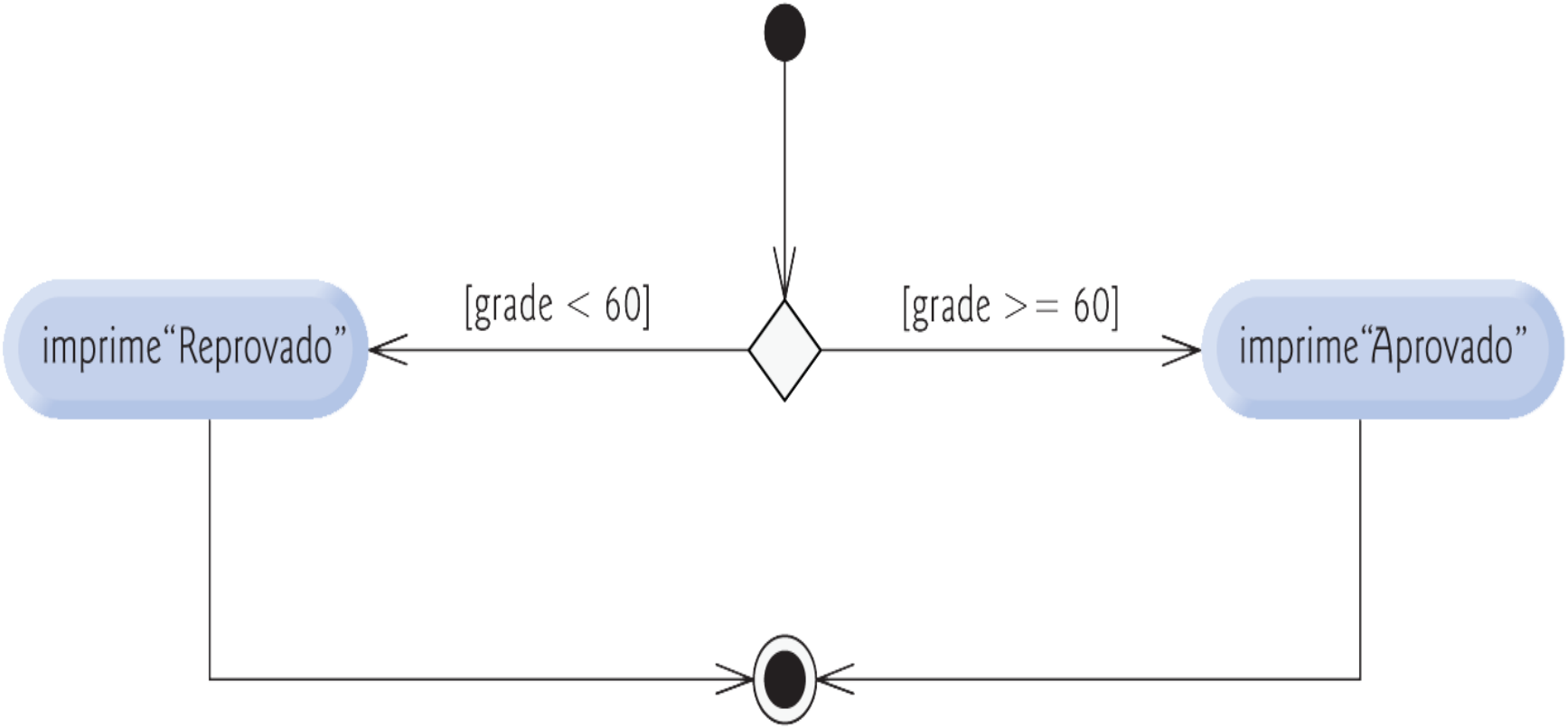

Figura 3.2 - Diagrama de atividades da UML de uma de instrução de seleção dupla `if...`else.

A instrução de seleção dupla `if...else` (*Continuação*)

- Instruções `if...else` aninhadas:
  - Instruções `if...else` podem ser colocadas dentro de outras instruções `if...else`.
- O problema do `else` oscilante:
  - Instruções `else` sempre estão associadas com a `if` imediatamente precedente, a menos que seja especificado o contrário pelas chaves `{ }`.
- Blocos:
  - Chaves `{ }` associam instruções a blocos.
  - Blocos podem substituir instruções individuais, como o corpo de uma `if`.

Boa Prática de programação

- Sempre utilizar as chaves em uma instrução `if...else` (ou outra) ajuda a evitar uma omissão acidental, especialmente ao adicionar instruções à parte `if` ou à parte `else` mais tarde.

```
If (condicao)
{
}
else
{
}
```

- Para evitar omitir uma ou as duas chaves, alguns programadores digitam as chaves de abertura ou fechamento de blocos antes de digitar as instruções individuais dentro das chaves.

A estrutura de seleção múltipla `switch`

- Instrução `switch`:
  - Utilizada para múltiplas seleções.

- O comando switch executa a expressão e compara o valor encontrado com os valores: , , , etc.
- Quando encontra a igualdade ele executa o bloco de comandos daquele valor.
- A execução continuará até o final do switch ou até que ele encontre um break.

A estrutura de seleção múltipla `switch`

- Sintaxe:

```
switch ( <expressão> ) {
    case <valor1>:<comandos 1>
    [break;]
    case <valor2>:<comandos 2>
    [break;]
    case <valor3>:<comandos 3>
    [break;]
    case <valor4>:<comandos 4>
    [break;]
    ...
    default: <comandos default>
}
```

A estrutura de seleção múltipla `switch`

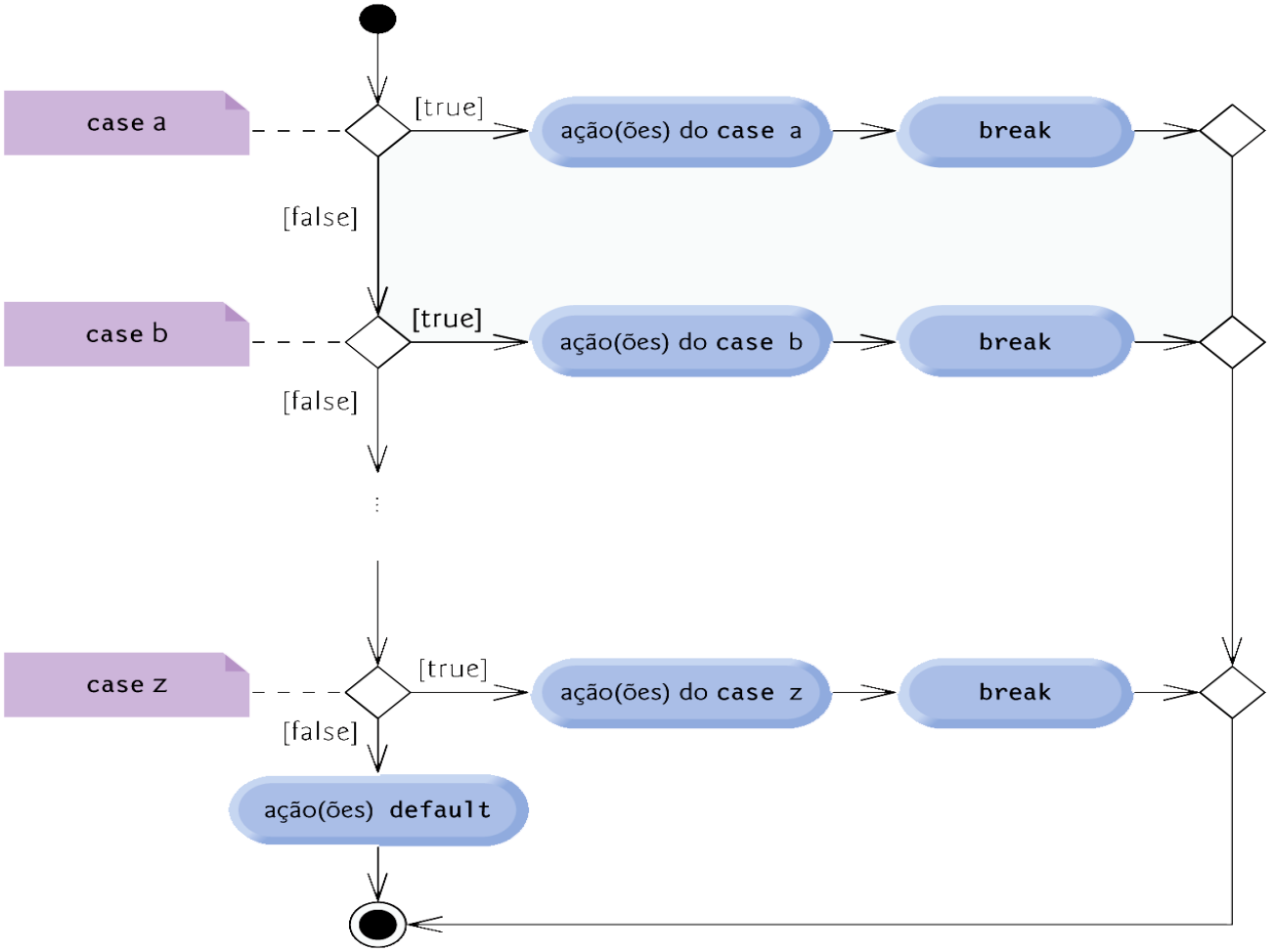

Figura 3.3 - Diagrama de atividade UML de instrução de seleção múltipla `switch` com instruções `break`.

A estrutura de seleção múltipla switch Exemplos

```
switch (i) {
  case 1 : System.out.println("Valor de i é 1");
  break;
  case 2 : System.out.println("Valor de i é 2");
  break;
  case 3 : System.out.println("Valor de i é 3");
  break;
  default: System.out.println("Default");
}

// Switch que irá imprimir na tela o valor "Vitor"
String nome = "Vitor";
switch (nome.hashcode()) {
  case "Vitor".hashCode() :
    System.out.println("Valor de nome é Vitor");
    break;
  case "Paulo".hashCode() :
    System.out.println("Valor de nome é Paulo");
    break;
  default:
    System.out.println("Default");
}
```

A estrutura de seleção múltipla switch Exemplos

- Não é necessário utilizar as chaves ({}) para delimitar um bloco de comandos, pois o interpretador executará todos eles até encontrar um break ou o switch finalizar.
- A única limitação do switch é que a deve retornar um valor numérico inteiro, qualquer um de seus quatro tipos primitivos.
- O switch não funciona com String, float, char e boolean por exemplo. E é por este motivo que no segundo switch do exemplo é utilizado o método hashcode (), que retorna um código inteiro único para cada String diferente.

A estrutura de seleção múltipla switch Boa prática de programação

- Embora cada case e a instrução default em uma switch possam ocorrer em qualquer ordem, coloque a instrução default por último.
- Quando a instrução default é listada por último, o break para essa instrução não é necessário.
- Alguns programadores incluem esse break para clareza e simetria com outros cases.

Operadores lógicos

- Operadores lógicos:
  - Permite formar condições mais complexas.
  - Combina condições simples.
- Operadores lógicos Java:
  - && (E condicional)
  - || (OU condicional)
  - & (E lógico booleano)
  - | (OU inclusivo lógico booleano)
  - ^ (OU exclusivo lógico booleano)
  - ! (NÃO lógico)

Operadores lógicos (*Continuação*)

- Operador E (&&) condicional.
  - Considere a seguinte instrução `if`:
    - `if ( gender == FEMALE && age >= 65 )`
    - `++++seniorFemales;`
    - Condição combinada é `true`:
    - se e somente se ambas as condições simples forem `true`.
    - Condição combinada é `false`:
    - se uma ou ambas as condições simples forem `false`.

Operadores lógicos (*Continuação*)

&& (E condicional)

expressão1	expressão2	expressão1 && expressão2
false	false	False
false	true	False
true	false	False
true	true	True

Figura 3.4 - Tabela-verdade do operador && (E condicional).

Operadores lógicos (*Continuação*)

- Operador OU condicional (`||`):
  - Considere a seguinte instrução `if`:

```
if ( ( semesterAverage >= 90 ) ||
     ( finalExam >= 90 ) )
     System.out.println ( “Aprovado” );
```

    - A condição combinada é `true`:
    - se uma das ou ambas as condições simples forem `true`.
    - A condição combinada é `false`:
    - se ambas as condições simples forem `false`.

Operadores lógicos (*Continuação*)

|| (OU condicional)

expressão1	expressão2	expressão1 || expressão2
false	false	false
false	true	true
true	false	true
true	true	true

Figura 3.5 - Tabela-verdade do operador || (OU condicional).

Operadores lógicos (*Continuação*)

- OU exclusivo lógico booleano (^):
  - Um dos seus operandos é `true` e o outro é `false`:
    - Avaliado como `true`.
  - Ambos os operandos são `true` ou ambos são `false`:
    - Avaliado como `false`.
- Operador de negação lógica (`!`):
  - Operador unário.

Operadores lógicos (*Continuação*)

^ (OU exclusivo lógico booleano)

expressão1	expressão2	expressão1 ^ expressão2
false	false	false
false	true	true
true	false	true
true	true	false

Figura 3.6 - Tabela-verdade do operador ^ (OU exclusivo lógico booleano).

A instrução de repetição `while`

- Instrução `while`:
  - Repete uma ação enquanto a condição de continuação do loop permanecer verdadeira.
  - Utiliza um *símbolo de agregação* no seu diagrama de atividades UML.
    - Mescla dois ou mais fluxos de trabalho.
    - É representado por um losango (como com os símbolos de decisão), mas com:
      - múltiplas setas de transição ‘incoming’;
      - somente uma seta de transição ‘outgoing’; e
      - nenhuma condição de guarda em uma seta de transição.

A instrução de repetição while

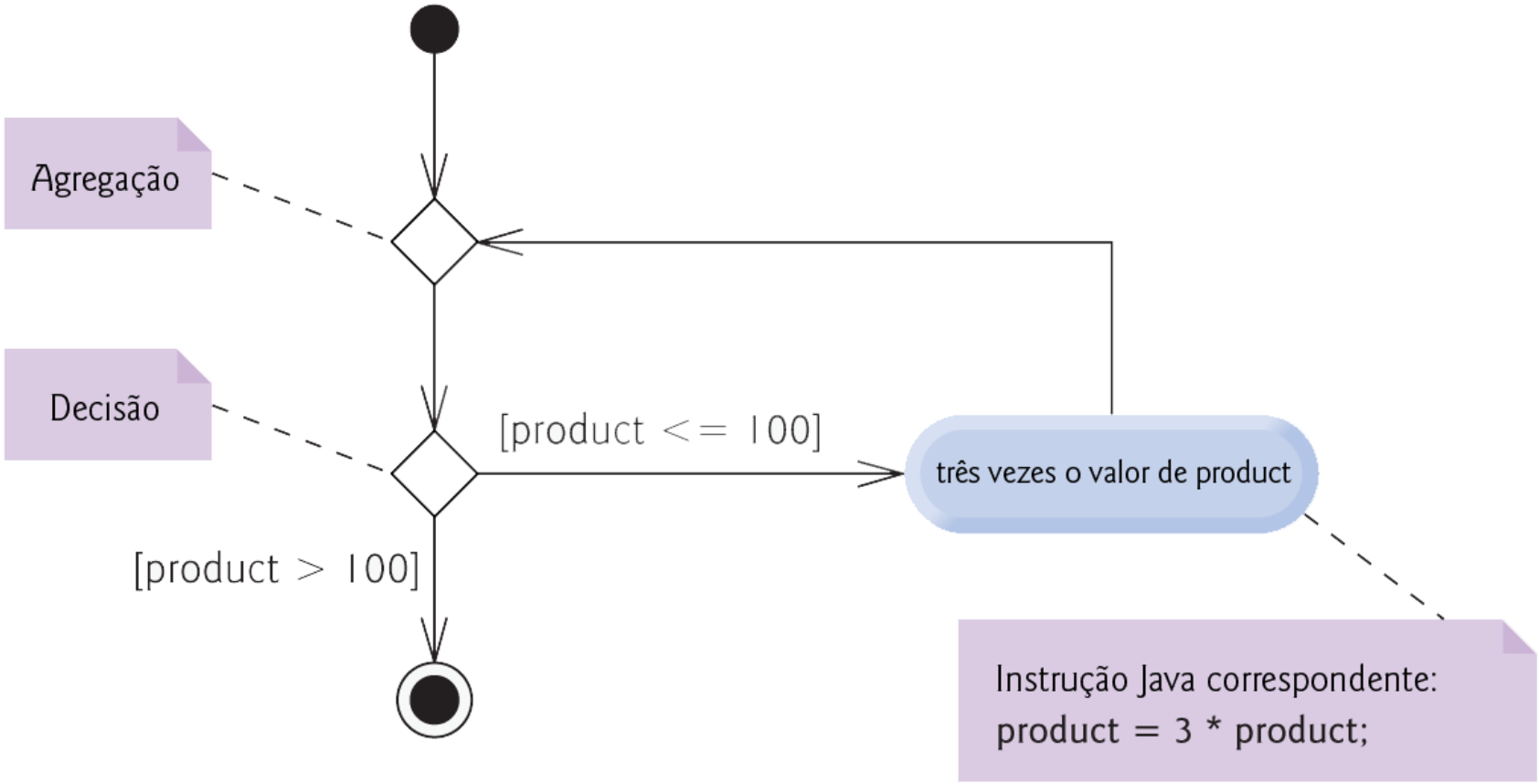

Figura 3.7 - Diagrama de atividades da UML da instrução de repetição while.

A instrução de repetição `while`

Sintaxe:

```
while (<condição>)
{
    <comandos>
}
```

A instrução de repetição while

Exemplo:

```
public class ExemploWhile {
  public static void main(String[] args) {
    int i=0;
    while(i++ < 10)
    {
      System.out.println("Contador é " + i);
    }

    while(true) {
      System.out.println("Laço infinito.");
    }
  }
}
```

A instrução de repetição `for`

• O comando for cria um laço de repetição no fluxo do programa baseado em três parâmetros:

expressão inicial: Onde é executado apenas uma vez, na entrada do laço.

condição: É executado a cada iteração do laço e determina quando o programa deve sair do mesmo. Caso a condição seja verdadeira, repete-se os do laço uma vez, caso seja falsa, o programa pula para a próxima instrução seguinte ao laço.

incremento: É uma operação normal, executada a cada iteração. Geralmente é usada para incrementar contadores ou configurar variáveis.

A instrução de repetição `for`

Sintaxe:

```
for (<expressão inicial>; <condição>; <incremento>)
{

   <comandos>

}
```

A instrução de repetição `for`

Exemplo:

```
public class ExemploFor {
    public static void main(String[] args)
   {
       for (int i=0; i<10; i++)
      {
          System.out.println("Contador é " + i);
      }

      for (;;)
      {
          System.out.println("Laço infinito.");
      }
    }
}
```

Exemplos com a estrutura `for`

- Variando a variável de controle em uma instrução `for`
  - Faz a variável de controle variar de 1 a 100 em incrementos de 1
    - `for ( int i = 1; i <= 100; i++ )`
  - Faz a variável de controle variar de 100 a 1 em incrementos de −1
    - `for ( int i = 100; i >= 1; i-- )`
  - Faz a variável de controle variar de 7 a 77 em incrementos de 7
    - `for ( int i = 7; i <= 77; i += 7 )`
  - Faz a variável de controle variar de 20 a 2 em decrementos de 2
    - `for ( int i = 20; i >= 2; i -= 2 )`
  - Faz a variável de controle variar na seqüência: 2, 5, 8, 11, 14, 17, 20
    - `for ( int i = 2; i <= 20; i += 3 )`
  - Faz a variável de controle variar na seqüência : 99, 88, 77, 66, 55, 44, 33, 22, 11, 0
    - `for ( int i = 99; i >= 0; i -= 11 )`

A instrução de repetição `do while`

• O comando do.. while é utilizada quando se quer que o corpo do laço seja executado pelo menos uma vez.
• A expressão de comparação é avaliada depois que o laço foi executado, enquanto ela for verdadeira os são repetidos.

A instrução de repetição **do while**

Sintaxe:

```
do
{
    <comandos>
} while (<condição>);
```

A instrução de repetição **do while**

Sintaxe:

```
public class ExemploDoWhile
{
   public static void main(String[] args)
   {
      int i=0;
      do
      {
          System.out.println("Contador é " + i);
      } while(++i < 10)

     do
     {
          System.out.println("Laço infinito.");
     } while(true);
  }
}
```

Comando **break**

• É a declaração de desvio usada para sair de um laço antes do normal.
• O tipo determina para onde é transferido o controle.
• O break é transfere o controle para o final de uma construção de laço (for, do, while ou switch).
• O laço vai encerrar independentemente de seu valor de comparação e a declaração após o laço será executada.

Comando **break**

Exemplo:

```
int i = 0;
while (true)
{
    System.out.println(i);
    if ( i++ >= 10 )
       break;
}
```

Este exemplo imprime os valores da variável i de 0 até 9.

Comando **continue**

- A declaração continue faz com que a execução do programa volte imediatamente para o início do laço, porém para a próxima interação.
- O continue faz o interpretador pular para a próxima iteração e obriga-o a testar a condição.

Comando **continue**

```
for (int i =-10; i < 10;  i++)
{
   if  ( i = = 0 )
       continue;
   System.out.println(i);
}
```

- No exemplo, é impresso os valores de - 10 até 9 pulando o número zero.

Exercício 1

- Implemente um programa em Java para a leitura das seguintes informações:
  - codigo, nome, nota1, nota2, nota3, nota4

- Todas as notas do tipo Float;
- Faça o seguinte teste:
  - Caso a média das quatro notas seja maior ou igual a 28, imprima: “Aluno Aprovado”;
  - Caso contrário, imprima: “Aluno Reprovado”.

Exercício 2

- Implemente um programa em Java para a leitura das seguintes informações:
  - codigo, nome, nota1, nota2, nota3, nota4

- Todas as notas do tipo Float (valores entre 0 e 5);
- Calcule a média e imprima de acordo com os seguintes valores: (Utilize o comando ***switch/case***)
  - Média = 0 (“Nota Zero – Péssimo”);
  - Média = 1 (“Nota Um – Ruim”);
  - Média = 2 (“Nota Dois – Regular”);
  - Média = 3 (“Nota Três – Bom”);
  - Média = 4 (“Nota Quatro – Ótimo”);
  - Média = 5 (“Nota Cinco – Excelente”);

Exercício 3

- Implemente um programa em Java para calcular a tabuada.
- O usuário deverá entrar com o número para cálculo.
- Faça duas versões, uma utilizando a instrução de repetição ***for*** e outra com ***while***