Linguagem de Programação

ADS – Análise e Desenvolvimento de Sistemas
Prof. Vagner Macedo

Aula 09
Estruturas de controle de repetição

Controle de Fluxo

- Podemos classificar os comandos aceitos pela linguagem Java em basicamente quatro categorias:
  - Tomada de Decisões
    - If-Else
    - Switch-Case
  - Laços e Repetições
    - For
    - While
    - Do-While

Controle de Fluxo

- Podemos classificar os comandos aceitos pela linguagem Java em basicamente quatro categorias:
  - Tratamento de Exceções
    - Try-Catch-Finally
    - Throw
  - Desvios
    - Break
    - Continue
    - Return

Laços de Repetição

- Laços controlam uma repetição de execução de acordo com uma condição imposta.
- Em Java, assim como tantas outras linguagens, existem três tipos de laço:
  - for
  - while
  - do..while

Comando: For

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

  - expressão inicial: 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.
  - Incremento/decremento: É uma operação normal, executada a cada iteração. Geralmente é usada para incrementar (ou decrementar) contadores ou configurar variáveis.

Comando: For

- Sintaxe:

```
for (<expr inicial>; <condição>; <incremento>){
   <comandos>
 }
```

Comando: For

- Exemplos:

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

Comando: For

- O primeiro **for** do exemplo anterior irá apresentar 10 mensagens indicando o contador de 0 até 9. Quando o valor do inteiro ‘i’ for igual a 10, o interpretador Java alternará o fluxo para o início do outro **for**.

- No segundo **for** do exemplo o código entra num laço infinito, ou seja, o programa não termina e sequer sairá do laço, já que não existe condição para que tal evento ocorra.

Comando: while

- O comando **while** é utilizado quando não se quer que o corpo do laço seja necessariamente executado
- A expressão de comparação é avaliada antes que o laço seja executado
- Enquanto a expressão for verdadeira, os comandos serão repetidos.

Comando: while

- Sintaxe:

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

Comando: while

- Exemplos:

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

Comando: while

- Assim como nos exemplos com o comando **for**, o primeiro exemplo imprime contadores de 0 até 9 e o segundo é um laço infinito, pois o resultado da condição sempre será verdadeiro.

Comando: do..while

- O comando **do**..**while** é utilizado 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, e enquanto ela for verdadeira os comandos do laço serão executados.

Comando: do..while

- Sintaxe:

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

Comando: do..while

- Exemplos:

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

Comando: while x do..while

- Assim como nos exemplos anteriores, o primeiro exemplo do comando **do..while** imprime contadores de 0 até 9 e o segundo é um laço infinito, pois o resultado da condição sempre será verdadeiro.
- Embora a diferença entre os dois comandos **while** sejam mínimas, cada uma é utilizada em uma determinada ocasião.

Comando: break

- É a declaração de desvio usada para sair de um laço antes do normal.
- O **break** transfere o controle para o final de uma construção de **fluxo** (for, do, while ou switch).
- O laço vai encerrar independentemente de seu valor de comparação.

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 em sua próxima iteração.
- O **continue** faz o interpretador pular para a próxima iteração e obriga-o a testar a condição.

Comando: continue

- Exemplo:

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

Obrigado!