Estruturas de repetição (*loops*) – Parte 1

Prof. Bruno Nogueira

Repetições

- Programas frequentemente precisam repetir uma mesma ação um número finito de vezes
  - Aferir nota a uma classe inteira
  - Distribuir peças aos jogadores de um jogo de tabuleiro
  - Enviar email aos assinantes de uma lista
- Tal como vimos até agora, nossa solução seria repetir o mesmo bloco de código ***n*** vezes
  - Considerando ***n*** alunos, jogadores ou assinantes
  - Código pouco elegante
  - Repetição desnecessária do mesmo código

```java
import java.util.Scanner;

public class CalculaAreaVariosCirculos
{
    public static final double PI = 3.14159;

    public static void main (String [ ] args)
    {
        double raio;  // em centímetros
        double area;  // em centímetros quadrados
        Scanner teclado = new Scanner (System.in);

        System.out.println("Entre o raio do primeiro círculo em centímetros: ");
        raio = teclado.nextDouble();

        area = PI * raio * raio;
        System.out.print("O círculo de raio de " + raio + " centímetros");
        System.out.println(" tem uma área de " + area + " centímetros quadrados.");

        System.out.println("Entre o raio do segundo círculo em centímetros: ");
        raio = teclado.nextDouble();

        area = PI * raio * raio;
        System.out.print("O círculo de raio de " + raio + " centímetros");
        System.out.println(" tem uma área de " + area + " centímetros quadrados.");

        System.out.println("Entre o raio do terceiro círculo em centímetros: ");
        raio = teclado.nextDouble();

        area = PI * raio * raio;
        System.out.print("O círculo de raio de " + raio + " centímetros");
        System.out.println(" tem uma área de " + area + " centímetros quadrados.");

    }
}
```

Repetições

- Uma porção de código que se repete é chamado de ***loop***
- Cada repetição do loop é chamado de **iteração**
- Loop é constituído de duas partes básicas
  - Corpo do loop: comando a ser executado em cada iteração
  - Condição de parada: mecanismo de decisão quando a repetição do loop deve parar

Comando *while*

- A primeira maneira que veremos para construir um loop é por meio do comando while
  - Do inglês "enquanto"
  - O corpo do loop é repetido enquanto uma determinada condição for verdadeira
  - Como o teste é baseado em uma condição booleana, é representado por uma expressão booleana
  - Sintaxe:

```
while (EXPRESSÃO_BOOLEANA)
{
    CORPO DO LOOP COMPOSTO
}
```

```
while (EXPRESSÃO_BOOLEANA)
    CORPO DO LOOP SIMPLES
```

Comando *while*

```
1  import java.util.Scanner;
2
3  public class WhileDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int cont, numero;
8          Scanner teclado = new Scanner(System.in);
9
10         System.out.println("Digite um número:");
11         numero = teclado.nextInt();
12
13         cont = 1;
14         while(cont <= numero)
15         {
16             System.out.print(cont + ", ");
17             cont++;
18         }
19         System.out.println("Fim da contagem.");
20     }
21 }
```

Comando *while*

Vamos supor que o usuário digitou o número 2

Iteração 1:
cont = 1
numero = 2
1 <= 2: true

Executado

```
1  import java.util.Scanner;
2
3  public class WhileDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int cont, numero;
8          Scanner teclado = new Scanner(System.in);
9
10         System.out.println("Digite um número:");
11         numero = teclado.nextInt();
12
13         cont = 1;
14         while(cont <= numero)
15         {
16             System.out.print(cont + ", ");
17             cont++;
18         }
19         System.out.println("Fim da contagem.");
20     }
21 }
```

Saída: 1,

Comando *while*

Vamos supor que o usuário digitou o número 2

Iteração 2:
cont = 2
numero = 2
2 <= 2: true

Executado

```
1  import java.util.Scanner;
2
3  public class WhileDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int cont, numero;
8          Scanner teclado = new Scanner(System.in);
9
10         System.out.println("Digite um número:");
11         numero = teclado.nextInt();
12
13         cont = 1;
14         while(cont <= numero)
15         {
16             System.out.print(cont + ", ");
17             cont++;
18         }
19         System.out.println("Fim da contagem.");
20     }
21 }
```

Saída: 1, 2,

Comando *while*

Vamos supor que o usuário digitou o número 2

Iteração 3:
cont = 3
numero = 2
3 <= 2: false

Não executado

```
1  import java.util.Scanner;
2
3  public class WhileDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int cont, numero;
8          Scanner teclado = new Scanner(System.in);
9
10         System.out.println("Digite um número:");
11         numero = teclado.nextInt();
12
13         cont = 1;
14         while(cont <= numero)
15         {
16             System.out.print(cont + ", ");
17             cont++;
18         }
19         System.out.println("Fim da contagem.");
20     }
21 }
```

Saída: 1, 2, Fim da contagem.

Comando *while*

Agora, vamos supor que o usuário digitou o número 0

Iteração 1:
cont = 1
numero = 0
0 <= 2: true

Não executado

```
1  import java.util.Scanner;
2
3  public class WhileDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int cont, numero;
8          Scanner teclado = new Scanner(System.in);
9
10         System.out.println("Digite um número:");
11         numero = teclado.nextInt();
12
13         cont = 1;
14         while(cont <= numero)
15         {
16             System.out.print(cont + ", ");
17             cont++;
18         }
19         System.out.println("Fim da contagem.");
20     }
21 }
```

Saída: Fim da contagem.

Comando *while*

- Tal como no comando if, as chaves são obrigatórias apenas em casos onde o corpo do loop é composto (mais de um comando)
- Corpo do loop pode ser executado zero vezes
  - Se a condição falhar na primeira iteração, o corpo não é executado nenhuma vez
  - Devemos ficar atentos com a escolha do teste
    - Erros na escolha da expressão booleana implicam em resultados inesperados. Um erro muito comum consiste na escolha de < ou > ao invés de <= ou >=

Exemplo

```java
import java.util.Scanner;

public class SomaInteirosPositivos
{
    public static void main(String [ ] args)
    {
        int somaTotal = 0, numero;
        Scanner teclado = new Scanner(System.in);

        System.out.println("Digite um número:");
        numero = teclado.nextInt();

        while(numero > 0)
        {
            somaTotal += numero;
            System.out.println("Digite um número:");
            numero = teclado.nextInt();
        }
        System.out.println("Soma total: " + somaTotal);
    }
}
```

Comando *do ... while*

- Para garantir que o corpo do loop seja executado pelo menos uma vez, podemos usar o comando do ... while
- É bastante similar com o comando while. A diferença é que o teste booleano é feito após o corpo do loop ter sido executado
- Sintaxe:

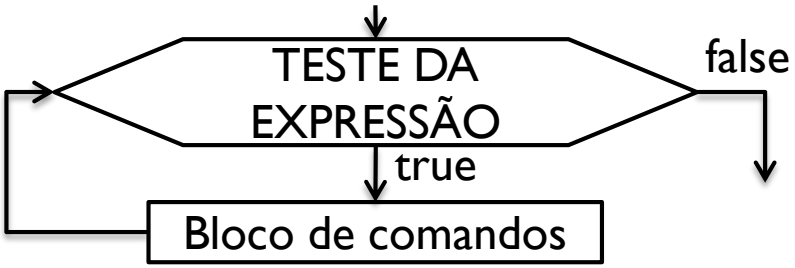
```
do
{
    CORPO DO LOOP COMPOSTO
} while (EXPRESSÃO_BOOLEANA);
```

```
do
    CORPO DO LOOP SIMPLES
while (EXPRESSÃO_BOOLEANA);
```

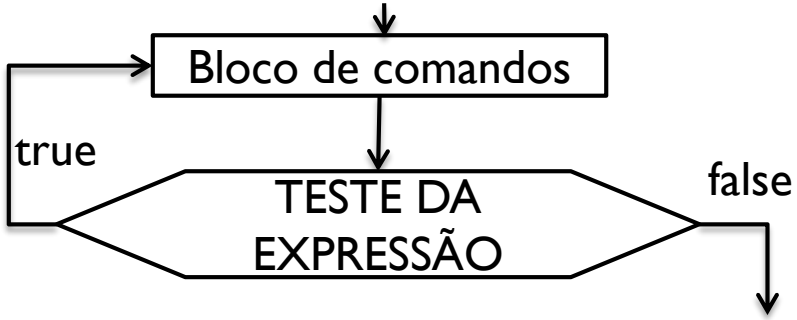
  - Atenção ao **;** após o comando while!

Comando *do ... while*

- While x do-while



while



do-while

Comando *do ... while*

```java
import java.util.Scanner;

public class DoWhileDemo
{
    public static void main(String [ ] args)
    {
        int cont, numero;
        Scanner teclado = new Scanner(System.in);

        System.out.println("Digite um número:");
        numero = teclado.nextInt();

        cont = 1;
        do
        {
            System.out.print(cont + ", ");
            cont++;
        } while(cont <= numero);
        System.out.println("Fim da contagem.");
    }
}
```

Comando *do ... while*

Vamos supor que o usuário digitou o número 2

Iteração 1:
cont = 2
numero = 2
2 <= 2: true

Executado

```
1  import java.util.Scanner;
2
3  public class DoWhileDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int cont, numero;
8          Scanner teclado = new Scanner(System.in);
9
10         System.out.println("Digite um número:");
11         numero = teclado.nextInt();
12
13         cont = 1;
14         do
15         {
16             System.out.print(cont + ", ");
17             cont++;
18         } while(cont <= numero);
19         System.out.println("Fim da contagem.");
20     }
21 }
```

Saída: 1,

Comando *do ... while*

Vamos supor que o usuário digitou o número 2

Iteração 2:
cont = 3
numero = 2
3 <= 2: false

Executado

```
1  import java.util.Scanner;
2
3  public class DoWhileDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int cont, numero;
8          Scanner teclado = new Scanner(System.in);
9
10         System.out.println("Digite um número:");
11         numero = teclado.nextInt();
12
13         cont = 1;
14         do
15         {
16             System.out.print(cont + ", ");
17             cont++;
18         } while(cont <= numero);
19         System.out.println("Fim da contagem.");
20     }
21 }
```

Saída: 1, 2, Fim da contagem.

Comando *do ... while*

Agora, vamos supor que o usuário digitou o número 0

Iteração 1:
cont = 2
numero = 0
2 <= 0: false

Executado

```
1  import java.util.Scanner;
2
3  public class DoWhileDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int cont, numero;
8          Scanner teclado = new Scanner(System.in);
9
10         System.out.println("Digite um número:");
11         numero = teclado.nextInt();
12
13         cont = 1;
14         do
15         {
16             System.out.print(cont + ", ");
17             cont++;
18         } while(cont <= numero);
19         System.out.println("Fim da contagem.");
20     }
21 }
```

Saída: 1, Fim da contagem.

Exercício

- Suponha que você tem um montante de dinheiro e quer aplicá-lo em banco para comprar um carro. Monte um programa para descobrir quanto tempo seu dinheiro precisa ficar aplicado, até atingir o montante desejado. Considere uma taxa de 2% ao mês.
  - Decisões:
    - Variáveis?
    - Constantes?
    - Repetições?
    - Algoritmo final?

```java
import java.util.Scanner;

public class AplicacaoBancaria
{
    public static final double TAXA = 0.2;  // 2% ao mês de rendimentos
    public static void main(String [ ] args)
    {
        double montanteDesejado, montanteInicial, montanteAtual;
        int numeroMeses = 0;
        Scanner teclado = new Scanner(System.in);

        System.out.println("Digite o montante inicial (em R$):");
        montanteInicial = teclado.nextDouble();

        System.out.println("Digite o montante desejado (em R$):");
        montanteDesejado = teclado.nextDouble();

        montanteAtual = montanteInicial;

        while (montanteAtual < montanteDesejado)
        {
            montanteAtual = montanteAtual + montanteAtual * TAXA;
            numeroMeses++;
        }
        System.out.println("Seu dinheiro deve ficar aplicado " + numeroMeses + " meses.");
    }
}
```

Exercício

- Transforme a solução anterior para uma que utilize o comando do-while. Qual das soluções é a mais adequada para o problema? Por quê?

Loops aninhados

- Dentro dos blocos de comando dos loops, é possível inserir qualquer comando – inclusive outros blocos de loop
- Loops aninhados são independentes (quando o mais interno termina sua execução, não necessariamente o mais externo também termina)

```java
import java.util.Scanner;

public class MediasProva
{
    public static void main(String [ ] args)
    {
        double soma, proximaNota;
        int numeroEstudantes;
        String resposta;
        Scanner teclado = new Scanner(System.in);

        do
        {
            System.out.println("Digite todas as notas a serem computadas.");
            System.out.println("Quando terminado, digite um número negativo.");

            soma = 0;
            numeroEstudantes = 0;

            proximaNota = teclado.nextDouble();
            while(proximaNota >= 0)
            {
                soma = soma + proximaNota;
                numeroEstudantes++;
                proximaNota = teclado.nextDouble();
            }
            if(numeroEstudantes > 0)
                System.out.println("A média é " + (soma / numeroEstudantes));
            else
                System.out.println("Não há notas a serem computadas.");
            System.out.println("Deseja calcular outras médias? (sim / nao)");
            resposta = teclado.next();
        } while(resposta.equalsIgnoreCase("sim"));
    }
}
```

Cuidado: loops infinitos

- Um erro comum é construirmos um loop que não termina nunca
  - Decorrente de alguma falha de lógica no nosso programa
  - É repetido "infinitamente"
  - Geralmente, termina quando há estouro de algum recurso (memória, pilha de processos, etc)
  - Quando acontecer, pode-se "matar o processo" (forçar o encerramento do programa) utilizando o comando ctrl + c

```java
import java.util.Scanner;

public class AplicacaoBancariaErrado
{
    public static final double TAXA = -0.2;  // -2% ao mês de rendimentos
    public static void main(String [ ] args)
    {
        double montanteDesejado, montanteInicial, montanteAtual;
        int numeroMeses = 0;
        Scanner teclado = new Scanner(System.in);

        System.out.println("Digite o montante inicial (em R$):");
        montanteInicial = teclado.nextDouble();

        System.out.println("Digite o montante desejado (em R$):");
        montanteDesejado = teclado.nextDouble();

        montanteAtual = montanteInicial;

        while (montanteAtual < montanteDesejado)
        {
            montanteAtual = montanteAtual + montanteAtual * TAXA;
            numeroMeses++;
        }
        System.out.println("Seu dinheiro deve ficar aplicado " + numeroMeses + " meses.");
    }
}
```

Exercício

- Qual é a saída dos seguintes códigos?

```
int count = 0;
while (count < 5)
    count++;
System.out.println("Valor: " + count);
```

```
int count = 0;
do
    count++;
while(count < 0);
System.out.println("Valor: " + count);
```

Exercício

- Reescreva o seguinte código, a fim de que use um while ao invés de um do-while. O comportamento deve ser exatamente o mesmo do código original.

```
int numero;
do
{
    System.out.println("Entre um número inteiro:");
    numero = teclado.nextInt();
    System.out.println("Você digitou o numero " + numero);
} while (numero > 0);
```

Comando *for*

- O comando for permite a construção de loops que podem ser controlados por um contador:

```
for(Ação_Inicialização; Expressão_Booleana; Ação_Atualização)
    COMANDO_SIMPLES
```

```
for(Ação_Inicialização; Expressão_Booleana; Ação_Atualização)
{
    BLOCO_DE_COMANDOS
}
```

  - Exemplo: um programa que imprime os valores de um contador inteiro de 1 a 3

```
int count;
for(count = 1; count <= 3; count ++)
    System.out.println(count);
```

Comando *for*

- Pode ser escrito como um comando while

```
for(Ação_Inicialização; Expressão_Booleana; Ação_Atualização)
{
    BLOCO_DE_COMANDOS
}
```

é equivalente a:

```
Ação_Inicialização;
while (Expressão_Booleana)
{
    BLOCO_DE_COMANDOS;
    Ação_Atualização;
}
```

Comando *for*

```
int count;
for(count = 1; count <= 3; count ++)
    System.out.println(count);
```

```
int count;
count = 1;
while(count <= 3)
{
    System.out.println(count);
    count++;
}
```

Comando *for*

```
1  import java.util.Scanner;
2
3  public class ForDemo
4  {
5      public static void main(String [ ] args)
6      {
7          int contagemRegressiva;
8          for(contagemRegressiva = 3; contagemRegressiva >= 0; contagemRegressiva--)
9          {
10             System.out.println(contagemRegressiva);
11             System.out.println("e contando...");
12         }
13         System.out.println("BOOM!");
14     }
15 }
```

Declarando o contador no próprio for

- É possível declarar o contador no próprio comando for

```
int count;
for(count = 1; count <= 3; count ++)
    System.out.println(count);
```

```
for(int count = 1; count <= 3; count ++)
    System.out.println(count);
```

Declarando o contador no próprio for

```
for(int count = 1; count <= 3; count ++)
    System.out.println(count);
```

- Neste caso, a variável count é local para o próprio loop
  - Não pode ser usada fora do loop
  - Seu uso fora do loop acarreta em erro de compilação

```
for(int count = 1; count <= 3; count ++)
    System.out.println(count);
System.out.println(count); // GERA ERRO!
```

while, *do-while* ou *for*?

- Antes de escolher, deve-se levar em consideração alguns aspectos
  - *do-while* pode ser escolhido se você tiver certeza de que seu código deve ser percorrido pelo menos uma vez
  - Na maioria das vezes, entretanto, deve-se considerar a possibilidade de um loop ser percorrido zero vezes. Para estes casos, deve-se usar *while* ou *for*
    - *for* mais adequado para situações controladas numericamente (por um contador, por exemplo)
    - Se a situação não é clara, opte pelo *while*
    - *for*, entretanto, deixa a lógica mais simples de ser entendida

Exercício

- Escreva um programa em Java que receba do usuário **n** números inteiros positivos e imprima, ao final:
  - A quantidade de valores digitados pelo usuário;
  - O menor valor digitado pelo usuário;
  - A média de todos os valores digitados pelo usuário.