FIAP

Lógica de Programação - Java

Aula 07 – Estruturas de repetição

Alexandre Russi Jr.

profalexandre.junior@fiap.com.br

Necessidade – Atividade 1

- Imagine só... Se tivesse que repetir algum comando 10 vezes
- Teria que escrever essa linha 10x ???

- Atividade 1: Dados uma sequência de 3 números inteiros. Calcule a soma de todos os números da sequência.

Atividade 1 – Resolução

```java
import java.util.Scanner;

public class Atividade01 {

     public static void main(String[] args) {
          Scanner tec = new Scanner(System.in);
          int num, soma = 0;

          System.out.print("Digite um número: ");
          num = tec.nextInt();
          soma = soma + num;

          System.out.print("Digite um número: ");
          num = tec.nextInt();
          soma = soma + num;

          System.out.print("Digite um número: ");
          num = tec.nextInt();
          soma += num;

          System.out.println("O resultado é: " + soma);
     }
}
```

Atividade 2 – Tabuada

- Dado um número inteiro positivo n, escreva um algoritmo que imprime a tabuada de n até o valor 5.
- Supondo que o valor digitado seja n = 4, seu programa deverá imprimir:

4 x 1 = 4

4 x 2 = 8

4 x 3 = 12

4 x 4 = 16

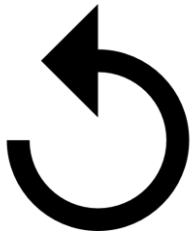
4 x 5 = 20

Atividade 2 – Resolução

```java
import java.util.Scanner;

public class Atividade02 {

    public static void main(String[] args) {
        Scanner tec = new Scanner(System.in);
        int n;
        System.out.print("Digite um número: ");
        n = tec.nextInt();

        System.out.println(n + " x 1 = " + (n * 1));
        System.out.println(n + " x 2 = " + (n * 2));
        System.out.println(n + " x 3 = " + (n * 3));
        System.out.println(n + " x 4 = " + (n * 4));
        System.out.println(n + " x 5 = " + (n * 5));
    }

}
```

Considerações

- Na Atividade 1 e 2 conseguimos resolver pois a quantidade de **repetições é pequena**.
- Temos padrões que se repetem!
- Para alguns problemas é necessário outra estrutura para nossos algoritmos:
  os comandos de repetição

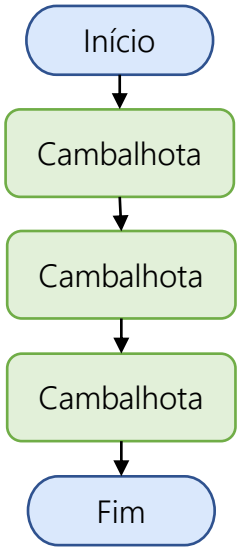


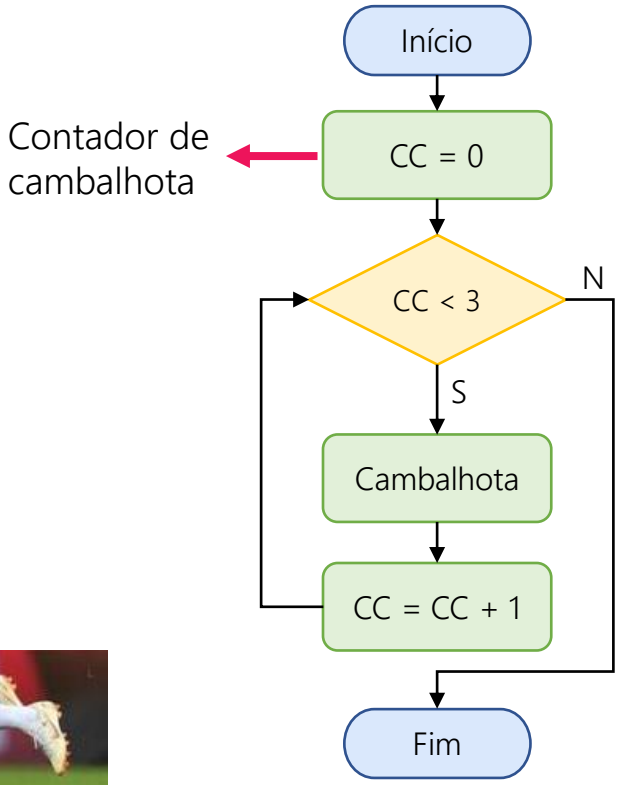
Repetição: While – Enquanto

- Os comandos de repetição são elementos fundamentais dentro de linguagens de programação pois permite a execução de um mesmo conjunto de instruções até que a condição não seja mais satisfeita.
- Ou seja, repete sempre se a **condição** for **verdadeira**.
- Uma execução desse conjunto de instruções é chamado de iteração.
- Iniciaremos nosso estudo com o comando de repetição while

```
while (condicao) {
    // bloco contendo instruções que
    // serão executadas repetidamente
}
```

Repetição: While – Enquanto

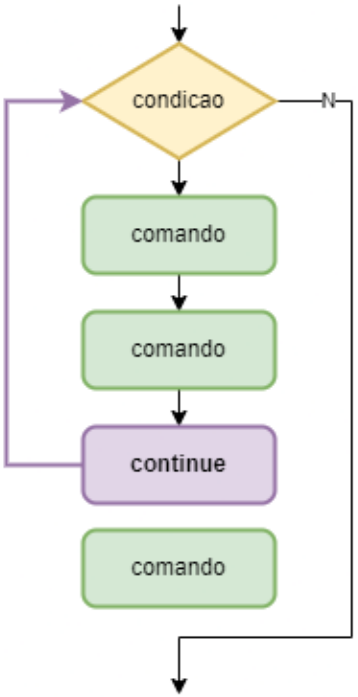





```
int cc = 0;

while (cc < 3) {
     System.out.println("Cambalhota");
     cc++;
}
```

Modificações do fluxo de um laço: continue

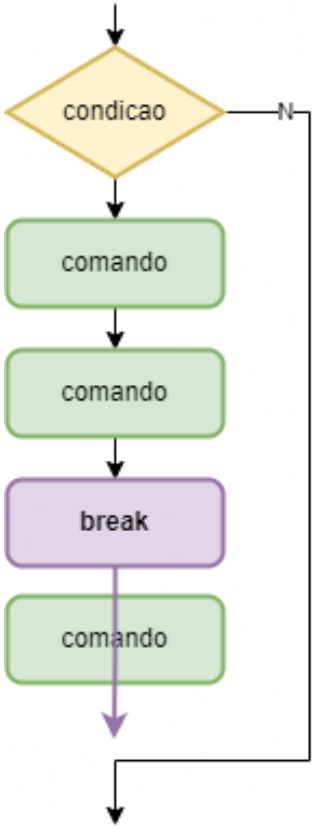
- O comando continue modifica o fluxo de um laço de repetição.
- Quando o continue é executado, ele **ignora** os comandos abaixo e executa o loop, ou a próxima **iteração**.




```
int cc = 0;
while (cc < 10) {
    cc++;

    if (cc == 3) {
        continue;
    }

    System.out.println("Cambalhota " + cc);

}
```

Modificações do fluxo de um laço: break

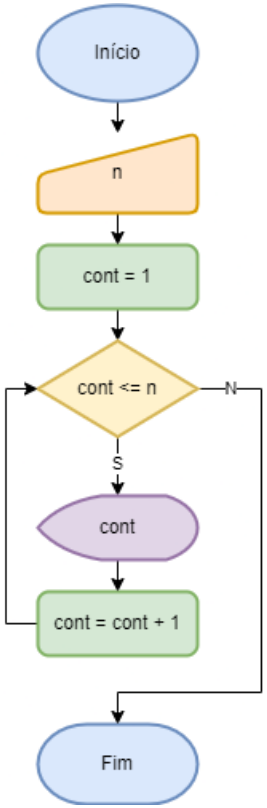
- Quando o break é executado, ele faz com que o programa saia do laço.




```
int cc = 0;
while (cc < 10) {
    cc++;

    if (cc == 7) {
        break;
    }

    System.out.println("Cambalhota " + cc);

}
```

Atividade 3: While – Enquanto

- Algoritmo que recebe um número inteiro positivo n, imprime na tela todos os números de 1 a n.



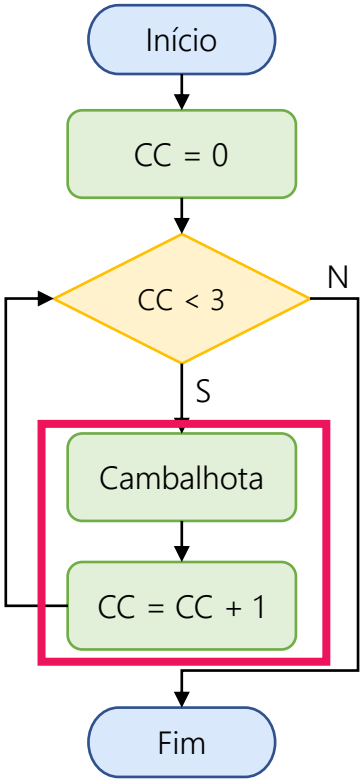

```
import java.util.Scanner;

public class Atividade03 {

	public static void main(String[] args) {
	Scanner tec = new Scanner(System.in);

	System.out.print("Digite um número n: ");
	int n = tec.nextInt();
	int cont = 1;

		while (cont <= n) {
			System.out.println(cont);
			cont++;
		}
	}
}
```

Repetição: Do/While – Faça Enquanto

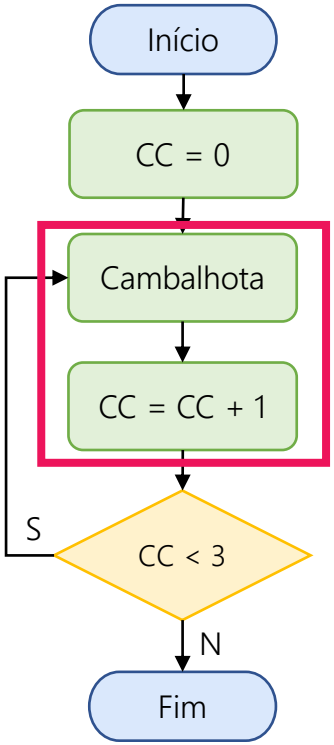
- O do/while executa primeiro e verifica a condição apenas no fim.
- Interpretação: **execute as instruções do bloco enquanto a expressão condicional for verdadeira**
- Ao término da execução das instruções do bloco, testa-se a expressão condicional
- Se a condição for falsa a repetição termina, senão volta ao início do bloco de instruções

```
do {
    // bloco contendo instruções que
    // serão executadas repetidamente
} while (condicao);
```

Repetição: Do/While – Faça Enquanto



while



do/while


```
int cc = 0;

do {
     System.out.println("Cambalhota");
     cc++;
} while (cc < 3);
```

Atividade 4: Problema da validação

- Em muitos dos nossos algoritmos fizemos validações dos dados de entrada. Contudo, apenas exibíamos mensagens e encerrávamos nossos algoritmos. Com os comandos de repetição temos condições de garantir que a informação esteja correta antes do algoritmo continuar.
- **Atividade 4**: Escreva um programa que dadas duas notas de 0 a 10 calcula a média aritmética entre elas.

Atividade 4: Problema da validação – Resolução

```java
import java.util.Scanner;

public class Atividade04 {

     public static void main(String[] args) {
          Scanner tec = new Scanner(System.in);
          double nota1, nota2;

          do {
               System.out.print("Digite a primeira nota: ");
               nota1 = tec.nextDouble();
          } while (nota1 < 0 || nota1 > 10);

          do {
               System.out.print("Digite a segunda nota: ");
               nota2 = tec.nextDouble();
          } while (nota2 < 0 || nota2 > 10);

          double media = (nota1 + nota2) / 2;
          System.out.println("A média é: " + media);
     }
}
```

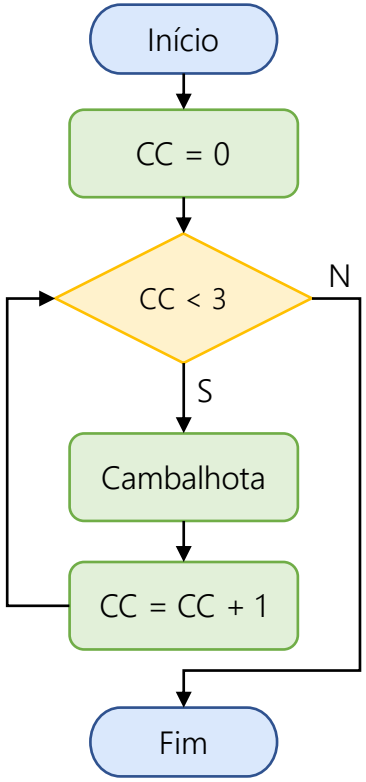
Repetição: For

- O comando de repetição for é adequado quando **sabemos exatamente a quantidade** de vezes que um conjunto de instruções irá repetir no algoritmo.
- Sintaxe do for pode ser apresentada como:

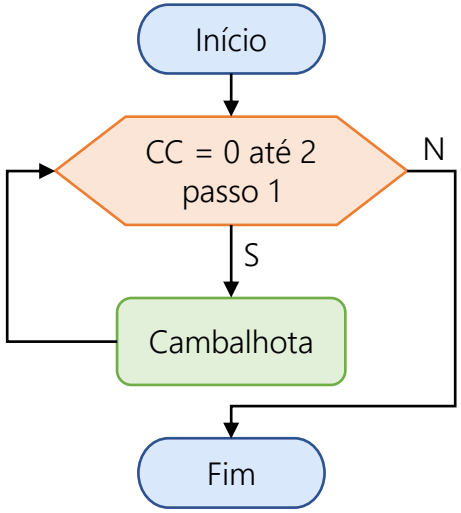
```
for (<inicial>; <condição>; <atualização>) {
    // bloco contendo instruções que
    // serão executadas repetidamente
}
```

- <inicial> é a declaração de um contador e sua inicialização
- <condição> é a condição de parada, ou seja, até onde seu contador deve ir
- <atualização> de quantas unidades seu contador atualiza
- Note que i++ é equivalente a i = i + 1;

Repetição: For



while



for

```
for (int cc = 0; cc < 3; cc++) {
      System.out.println("Cambalhota");
}
```

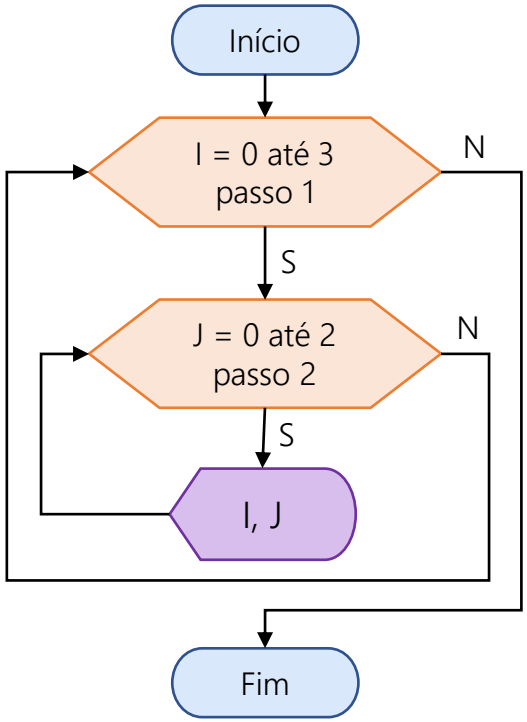
Atividade 5: Recebendo a qtd de cambalhotas

- Faça um programa que receba a quantidade de cambalhotas que o usuário deseja
- A seguir, seu programa deve exibir a mensagem “Cambalhota” a quantidade de vezes que o usuário solicitou.
- Utilize o laço for.

Atividade 5 – Resolução

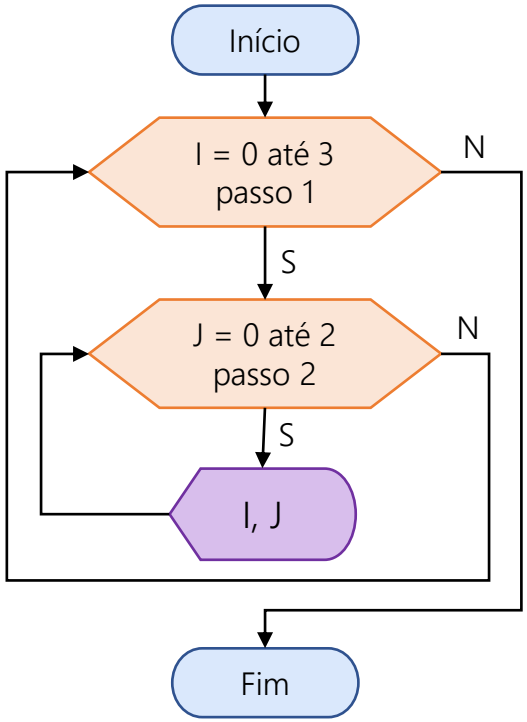
```
import java.util.Scanner;

public class Atividade05 {

    public static void main(String[] args) {
        Scanner tec = new Scanner(System.in);

        System.out.print("Digite a qtd de cambalhotas: ");
        int qtdCamb = tec.nextInt();

        for (int cc = 0; cc < qtdCamb; cc++) {
            System.out.println("Cambalhota " + cc);
        }

    }

}
```

Repetição encadeada ou Laços Aninhados



I	J
0	0
0	2
1	0
1	2
2	0
2	2
3	0
3	2

Repetição encadeada ou Laços Aninhados




```
for (int i = 0; i <=3; i++) {

     for (int j = 0; j <= 2; j+=2) {

          System.out.printf("%d, %d \n", i, j);

     }

}
```

Desafios e exercícios

Estruturas de repetição e condição

Exercício 1

- Faça um programa que exiba a mensagem “Olá, Mundo”.
- Essa mensagem deverá ser exibida repetidamente.
- Ao final de toda iteração da repetição, você deve perguntar ao usuário se ele deseja exibir a mensagem novamente.
- Se sim, exiba novamente. Senão, saia do loop e exiba a mensagem “Fim”.

Exercício 2

- Contagem de 0 a 100 pulando de 10 em 10.
- O terminal deve ficar assim:

```
0
10
20
30
40
50
60
70
80
90
100
```

Exercício 3

- Faça um programa que receba um número n
- Exiba a tabuada deste número do 0 ao 25.
- Utilize laços de repetição.

Exercício 4

- Faça um programa que receba 10 valores digitados pelo usuário e, ao final, informe qual é a soma deles.

Exercício 5

- Faça um programa que receba 12 valores digitados pelo usuário e, ao final, informe qual é o maior deles.

Exercício 6

- Faça um programa capaz de exibir todos os valores pares entre 2 e um valor fornecido pelo usuário.

Exercício 7

- Um professor quer saber quantos alunos de uma sala de 20 tiveram nota maior do que a média. Faça um programa onde o professor informe a média da turma e as notas de cada um dos 20 alunos e, ao final, seja exibido quantos alunos tiveram nota superior à média e quantos tiveram nota inferior à média.

Exercício 8

FIAP

- Escreva um programa que dado um inteiro n **positivo** calcula e imprime a soma de todos os números inteiros entre 1 e n.
- Valide a entrada do usuário, só aceite números positivos!!

- Dica: use **do/while** para a validação e **for** para a soma.

- Por exemplo, se n = **10** então deverá ser calculado:
  - 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = **55**
- E a impressão final seria:
  - A soma de 1 até **10** é: **55**

Exercício 9

FIAP

- Escreva um algoritmo que recebe um inteiro positivo **n** e imprime todos os divisores positivos de n.
- Utilize o laço for.

- Exemplo:

  Suponha que n = 28, nessa situação devemos imprimir os números

  1, 2, 4, 7, 14 e 28, que são todos os divisores do 28.

- Dica: para o número ser divisor de n, a divisão precisa ter **resto nulo**.

Exercício 10 – Desafio

- Determine e mostre todos os números primos no intervalo de 2 a 2000.

Dicas:

- Para resolver esse problema, primeiro faça um algoritmo que verifica se um número inteiro qualquer é primo ou não.
- A seguir, com esse código em mãos, faça os ajustes necessários para mostrar todos os números primos no intervalo solicitado.

- Você precisará colocar uma estrutura de repetição dentro da outra.
  - Laços aninhados!!!!

Alguma Dúvida?

Entre em contato por e-mail ou via LinkedIn

/alexandrerussi