

Laços de Repetição

while
do – while
for

Programação Orientada a Objetos – Aula 04
Professores: Hamilton Machiti da Costa

Laços de Repetição

- Servem para mudar o fluxo de execução de um algoritmo de modo a repetir um mesmo trecho de código um número limitado de vezes.
- Para que a execução do laço pare é necessária uma condição de parada, que é expressa por uma expressão booleana (lógica).
- Se a estrutura laço de repetição, também conhecida por **loop**, não existisse, seria necessário escrever várias vezes o mesmo trecho de código, como no slide a seguir.
- Considere a seguinte situação:
 - Deseja-se escrever na tela os números de 1 a 10. Sem o uso de laços de repetição, o código ficaria assim:

Se os loops não existissem...

```
public class SemLoop{  
    public static void main(String[] args){  
        System.out.println("1");  
        System.out.println("2");  
        System.out.println("3");  
        System.out.println("4");  
        System.out.println("5");  
        System.out.println("6");  
        System.out.println("7");  
        System.out.println("8");  
        System.out.println("9");  
        System.out.println("10");  
    }  
}
```

while

- O loop while (enquanto) primeiro testa condição lógica; se ela for verdadeira, executa o código que está dentro do loop e testa a condição lógica de novo; se for verdadeira, executa de novo; e assim sucessivamente até que a condição lógica se torne falsa.
- Usado principalmente quando o número de execuções do loop é desconhecido.
Ex: leia inteiros digitados pelo usuário até que ele digite -1

```
int x = 0;
while (x != -1) {
    x = Integer.parseInt(
        JOptionPane.showInputDialog("Digite um número"));
}
```

Imprimindo de 1 a 10 com while...

```
public class LoopWhile{  
    public static void main(String[] args) {  
        int i = 1;  
        while (i <= 10) {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

do - while

- O loop do while (faz enquanto), primeiro executa o código que está dentro do loop e depois testa condição lógica; se for verdadeira, executa novamente e testa de novo; se for verdadeira, executa de novo; e assim sucessivamente até que a condição se torne falsa.
- Usado quando se quer que o código que está no loop seja executado pelo menos uma vez. Ex: para validar uma entrada de dados, ler valores até que o usuário digite um número maior ou igual a 0 para evitar uma raiz quadrada de número negativo.

```
int x;  
do {  
    x = Integer.parseInt(JOptionPane.showInputDialog(  
        "Digite um número maior que zero"));  
} while(x < 0);
```

Imprimindo de 1 a 10 com do while...

```
public class LoopDoWhile{  
    public static void main(String[] args) {  
        int i = 1;  
        do {  
            System.out.println(i);  
            i++;  
        } while (i <= 10);  
    }  
}
```


for

- for (para): é dividido em 3 partes - **for (A; B; C)**
 - A) declaração da variável contadora
 - B) condição lógica de parada
 - C) incremento da variável contadora.
- De modo análogo ao while, primeiro testa a condição de parada (B) e, se for verdadeira, executa o código interno ao loop; quando chega no final, incrementa a variável contadora declarada em (A) com o incremento definido em (C). Então testa a condição novamente; se for verdadeira, volta a executar e incrementar; caso contrário, sai do loop.
- Usado quando se conhece o número de voltas que será dado no loop.

Imprimindo de 1 a 10 com for...

```
public class LoopFor{  
    public static void main(String[] args) {  
        for(int i = 1; i <= 10; i++){  
            System.out.println(i);  
        }  
    }  
}
```

Imprimindo de 10 a 1 com for...

```
public class LoopForReverso{  
    public static void main(String[] args) {  
        for(int i = 10; i > 0; i--){  
            System.out.println(i);  
        }  
    }  
}
```