**01Laço com while**

**Transcrição**

Finalmente chegamos nos laços, a última estrutura de controle de fluxo, básica e primordial em todas as linguagens! Criaremos uma classe específica para aprendermos sobre a estrutura de laço de repetição, o TestaWhile.

O while é uma palavra chave e, dentro dos parênteses, obrigatoriamente recebe uma expressão booleana, assim como o if. Por isso, precisaremos incluir algo lá dentro, que nos devolva true ou false.

public class TestaWhile {

public static void main(String[] args) {

int contador = 0;

while(contador <= 10) {

System.out.println(contador);

contador = contador + 1;

}

}

}Copiar código

Ao salvarmos e rodarmos o código, serão impressos os números de 0 a 10, como gostaríamos!

O while é uma instrução muito simples - lembrando que é preciso sempre inicializar e declarar a variável a ser utilizada, neste caso, em contador. Para reforçarmos algo que já foi visto, o escopo, poderemos imprimir contador novamente após o while:

public class TestaWhile {

public static void main(String[] args) {

int contador = 0;

while(contador <= 10) {

System.out.println(contador);

contador = contador + 1;

}

System.out.println(contador);

}

}Copiar código

A partir do qual se obtém a impressão de 0 a 11!

É claro que cabem outras condições booleanas no lugar de contador <= 10). Não é muito comum utilizarmos o formato contador = contador + 1; quando operamos sobre a própria variável, uma vez que existe uma forma mais sucinta, herdada do C:

contador += 1;Copiar código

Não é que seja "igual a mais um"! Queremos somar 1 nele mesmo. É uma sintaxe estranha, mas indica exatamente o mesmo que contador = contador + 1;. Para o mesmo efeito, existe ainda o **++**:

contador++;Copiar código

Esta, na verdade, é a forma mais comum de se somar o valor de si mesmo mais uma vez, e usar ++contador; (o pré-incremento) também traria o mesmo resultado. Há casos em que existem diferenças, mas por ora não nos preocuparemos com isso.

É muito mais importante entendermos o escopo, que a variável precisa ser inicializada antes de se fazer qualquer ação com ela, pois isso não acontece automaticamente em condições temporárias, e que o while é o sistema de laço mais simples de todos.

**02Enquanto isso, o while...**

Fernando decidiu praticar seu conhecimento sobre a estrutura de repetição while. Ele escreveu o seguinte programa:

package projeto;

public class Programa {

public static void main(String[] args) {

int contador = 1;

while(contador <= 10) {

System.out.println(contador);

}

}

}Copiar código

Todavia, seu programa só imprime no console 1 infinitamente.

Consegue enxergar o problema no código de Fernando? Depois de formar uma opinião compare seu achado com a resposta do instrutor logo em seguida.

**Opinião do instrutor**

O while aceita receber dois valores, true e false. O valor true mantém o loop em execução, já o false o finaliza. Esses valores podem ser resultados de uma expressão como contador <= 10. Todavia, no código de Fernando, ele não incrementou a variável contador a cada iteração do while. Devido a esse esquecimento, a variável contador sempre mantinha o valor 1 fazendo com que o programa caísse em um loop infinito.

**03Fixando o laço while**

Quais as afirmações abaixo são verdadeiras em relação ao while?

Parte superior do formulário

* O while é executado enquanto a expressão condicional definida for falsa. Caso ela seja verdadeira o while deixa de ser executado.
* Alternativa correta

Dado o seguinte código:

package projeto;

public class Programa {

public static void main(String args[]) {

while(false) {

System.out.println(1);

}

}

}

Ele imprimirá no console do Eclipse o valor 1 infinitas vezes.

* Alternativa correta

Na expressão condicional do while é possível utilizar qualquer operador de comparação (< [menor], > [maior], <= [menor ou igual], >= [maior ou igual], == [igual a] e != [diferente de]) e qualquer operador lógico (&& [and], || [ou]).

**Isso aí, aluno!** Todos os operadores de comparação e lógicos são válidos na expressão condicional do while! Use-os com sabedoria!

* Alternativa correta

O while sempre precisará de uma expressão condicional que definirá quando o laço deve ser interrompido.

**Muito bem!** Lembre-se, essa expressão condicional precisará ser informada dentro dos parênteses do while e pode ainda utilizar qualquer um dos operadores de comparação e operadores lógicos aprendidos no capítulo 6.

Parte inferior do formulário

Parabéns, você acertou!

**04Escopo nos laços**

**Transcrição**

Para vermos o laço de forma mais estruturada e desafiadora, faremos uma somatória com os números de 0 a 10, criando a classe TestaSomatoria:

public class TestaSomatoria {

public static void main(String[] args) {

int contador = 0;

while(contador <= 10) {

int total = 0;

total = total + contador;

System.out.println(total);

contador++;

}

}

}Copiar código

Vamos imprimir as somatórias parciais para ver o que está acontecendo?

Queremos que se mostre 0, seguido de 1, e então 2, 3, por causa de 1 + 2, e então 6, de 1 + 2 + 3. No entanto, obteremos:

0

1

2

3

4

5

6

7

8

9

10Copiar código

Ué! Não funcionou! Isto porque toda vez que se entra no while, é criada uma nova variável total por causa do escopo e, ao voltarmos ao próximo laço, quando ocorre a **iteração**, ele zera de novo, pois a velha total já deixou de existir.

Falta acertarmos o escopo declarando e inicializando a variável total após a linha que contém contador:

public class TestaSomatoria {

public static void main(String[] args) {

int contador = 0;

int total = 0;

while(contador <= 10) {

total = total + contador;

System.out.println(total);

contador++;

}

}

}Copiar código

Se salvarmos e rodarmos novamente, desta vez veremos o seguinte no Console:

0

1

3

6

10

15

21

28

36

45

55Copiar código

Ou seja, as somatórias parciais, incluindo a última, 55, que é o número desejado. Se quisermos apenas este resultado final, poderemos deixar o código assim:

public class TestaSomatoria {

public static void main(String[] args) {

int contador = 0;

int total = 0;

while(contador <= 10) {

total = total + contador;

contador++;

}

System.out.println(total);

}

}Copiar código

Salvando e rodando o código novamente, obtém-se a impressão de 55.

É possível deixar este código mais enxuto, porém focaremos em total = total + contador;, que já vimos que pode ser escrito assim: total += contador;, o qual traz exatamente o mesmo resultado.

**05Um erro de compilação...**

Clarice, amiga de Fernando, percebeu que ele está com dificuldade na implementação de um loop feito com while. Seu objetivo é imprimir no console os números de 1 a 10. No entanto, o código não compilava. Vejamos seu código:

package projeto;

public class Programa {

public static void main(String[] args) {

while (contador <= 10) {

int contador = 1;

System.out.println(contador);

contador++;

}

}

}Copiar código

Se você estivesse com Clarice agora, como explicaria o problema do código para ela? Depois de pensar em uma explicação, compare-a com a do instrutor a seguir.

**Opinião do instrutor**

Fernando sabiamente se preocupou em incrementar a variável contador a cada iteração do while para que o loop chegasse ao fim e não caísse em um loop infinito. Todavia, ele declarou a variável contador dentro do bloco do while. Por uma questão de escopo, essa variável só existirá dentro do bloco while. Para piorar as contas, a expressão contador <= 10 tenta acessar uma variável que ainda não foi declarada, razão principal do erro de compilação que esta enfrentado.

Corrigindo o código temos:

package projeto;

public class Programa {

public static void main(String args[]) {

int contador = 1;

while (contador <= 10) {

System.out.println(contador);

contador++;

}

}

}Copiar código

**06Laço com for**

**Transcrição**

O for tem a sintaxe um pouco mais estranha. O while é uma estrutura de laço, e o for realiza a mesma tarefa, porém possui algumas vantagens em relação à legibilidade, mesmo que o resultado final - o *bytecode* - seja o mesmo. Criaremos TestaFor, em que incluiremos algo equivalente ao laço feito anteriormente, que conta de 0 a 10 imprimindo todos os números.

Diferentemente do while, não é preciso declararmos contador fora dele, pois o for, palavra chave do Java, tem uma sintaxe muito diferente. Até então, utilizamos apenas ponto e vírgula no fim dos *statements*, isto é, das linhas. Neste caso, usaremos o ponto e vírgula **dentro dos parênteses** (isto também herança do C).

Dentro dos parênteses, então, serão criados três "espaços" intercalados por ponto e vírgula, e então abriremos e fecharemos as chaves normalmente. O primeiro espaço é opcional e costuma ter a declaração e inicialização da variável, sendo executado **apenas uma vez**.

O segundo espaço é executado **todas as vezes** e contém a condição booleana para saber se ele deve ou não entrar no laço, ou seja, executar a próxima iteração. No nosso caso, queremos saber se contador é menor ou igual a 10, como no while.

O terceiro espaço geralmente é ocupado por aquilo a ser executado ao fim de cada iteração, o que acaba sendo um tanto estranho para quem não está bem ambientado com isto. O código ficará desta maneira:

public class TestaFor {

public static void main(String[] args) {

for(int contador = 0; contador <= 10; contador++) {

System.out.println(contador);

}

}

}Copiar código

Salvaremos e rodaremos o código, e obteremos o esperado, como em while:

0

1

2

3

4

5

6

7

8

9

10Copiar código

Diferentemente do while, apesar de int contador = 0 valer no escopo do for inteiro em todas as iterações, ele não é zerado, sendo executado apenas uma vez, e por isto sua sintaxe não é muito intuitiva. Se quisermos imprimir o último valor que o contador estava lendo, não conseguiremos, por conta do escopo.

O for oferece a possibilidade de haver uma variável que participa de todas as iterações, que é o que precisamos, mas depois do for, ela deixa de valer.

Não é melhor usarmos o while, então? Depende. Muitas vezes queremos utilizar a variável temporariamente, somente dentro do laço, e é por isso que o for é mais atrativo, e se adequa melhor a este tipo de caso.

No entanto, while e for são intercambiáveis, e inclusive existe outro laço, denominado *do-while*, que não veremos neste curso, mas que também poderia ser utilizado.

**07Transformando while em for**

Dessa vez, Clarice resolveu desafiar Fernando a transformar o código a seguir que utiliza um laço while em um código que utilize o for. Você consegue ajudar Fernando a chegar a resposta do desafio? Compare a sua resposta com a do instrutor e verifique se acertou!

package projeto;

public class Programa {

public static void main(String[] args) {

int contador = 0;

while(contador <= 10) {

System.out.println(contador);

contador++;

}

}

}Copiar código

**Opinião do instrutor**

Como visto na aula, o for oferece uma opção de declaração mais enxuta, onde precisamos informar dentro dos parênteses do for a inicialização da variável que irá servir de auxiliar para a condição de parada, a condição de parada que deve ser verificada como verdadeira para que o laço seja executado e, por fim, a atualização do valor da variável auxiliar. Lembrando que essas informações precisam ser separadas umas das outras por ponto e vírgula (;). A resposta ao desafio ficaria assim:

package projeto;

public class Programa {

public static void main(String[] args) {

for(int contador = 0; contador <= 10; contador++) {

System.out.println(contador);

}

}

}Copiar código

**08Mão na massa: Laços**

Finalmente, chegou a hora de praticarmos os laços!

**While**

1) Crie uma classe e seu método main, no curso demos o nome de "**TestaWhile**".

2) Escreva o **while** que imprima os números de 0 a 10, lembre-se de criar sua variável **contador** com o valor 0, ficando dessa forma,

public class TestaWhile {

public static void main(String[] args){

int contador = 0;

while(contador <= 10) {

System.out.println(contador);

contador = contador + 1;

}

}

}Copiar código

Podemos trocar a linha que incrementa a variável **contador** por:

contador += 1;Copiar código

Ou

contador++; //mais usado!Copiar código

**For**

1) Crie uma classe e seu método main, no curso escolhemos "**TestaFor**"

2) Escreva um laço **for** que imprima os números de 0 a 10, desta forma:

public class TestaFor {

public static void main(String[] args){

for(int contador = 0; contador <= 10; contador++) {

System.out.println(contador);

}

}

}Copiar código

Repare que não podemos usar a variável **contador** fora do escopo do **for**.

**Opinião do instrutor**

Qualquer dúvida, não hesite! Nos envie uma mensagem no fórum!

**09Laços encadeados**

**Transcrição**

Já vimos todos os comandos básicos da sintaxe. Vamos praticar o uso dos laços, com o if, para sedimentarmos este conhecimento adquirido no curso! Criaremos uma classe para testarmos **laços encadeados**, aninhados uns aos outros: TestaLacos, com um main para imprimirmos de 0 a 10 dez vezes em linhas distintas, com a tabuada de cada número.

Usaremos o int multiplicador, começando pela tabuada do 1, indo à do 10. Dentro deste laço, queremos fazer outro, com valor diverso, como em um contador, também começando do 0 e indo a 10.

public class TestaLacos {

public static void main(String[] args) {

for(int multiplicador = 1; multiplicador <= 10; multiplicador++) {

for(int contador = 0; contador <= 10; contador++) {

System.out.println(multiplicador \* contador);

}

}

}

}Copiar código

Se pedirmos para que seja impresso multiplicador \* contador, obteremos algo gigantesco, como verificaremos salvando e rodando o código. Serão impressos os resultados contendo as tabuadas, mas queremos algo um pouco mais organizado. Para isso, em vez de utilizarmos o System.out.println();, usaremos System.out.print();, seguido de System.out.print(" ");, que nos trará os números todos alinhados horizontalmente.

Ainda não é isto que queremos! Queremos um "Enter" a cada tabuada, quer dizer, cada tabuada em uma linha. Vamos, então, incluir outro System.out.println(); após o escopo do segundo for, assim:

public class TestaLacos {

public static void main(String[] args) {

for(int multiplicador = 1; multiplicador <= 10; multiplicador++) {

for(int contador = 0; contador <= 10; contador++) {

System.out.print(multiplicador \* contador);

System.out.print(" ");

}

System.out.println();

}

}

}Copiar código

Salvando e rodando o código acima, obteremos, como gostaríamos:

0 1 2 3 4 5 6 7 8 9 10

0 2 4 6 8 10 12 14 16 18 20

0 3 6 9 12 15 18 21 24 27 30

0 4 8 12 16 20 24 28 32 36 40

0 5 10 15 20 25 30 35 40 45 50

0 6 12 18 24 30 36 42 48 54 60

0 7 14 21 28 35 42 49 56 63 70

0 8 16 24 32 40 48 56 64 72 80

0 9 18 27 36 45 54 63 72 81 90

0 10 20 30 40 50 60 70 80 90 100Copiar código

**10Mais laços com break**

**Transcrição**

Feitas as tabuadas do vídeo anterior, vamos testar mais laços encadeados e ver como eles podem se comunicar? Criaremos para isto a classe TestaLacos2, para a qual copiaremos e colaremos o conteúdo de TestaLacos. Desta vez, substituiremos multiplicador por linha, enquanto contador passará a ser coluna. E não faremos mais multiplicações, e sim com que apareçam 10 linhas e 10 colunas. A partir do código abaixo, o que vocês acham que acontecerá?

public class TestaLacos2 {

public static void main(String[] args) {

for(int linha = 0; linha < 10; linha++) {

for(int coluna = 0; coluna < 10; coluna++) {

System.out.print("\*");

}

System.out.println();

}

}

}Copiar código

Na aba "Console", será mostrado algo não muito interessante:

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***Copiar código

Uma grande quantidade de laços encadeados acaba não sendo esteticamente aprazível e, às vezes, queremos que um laço se comunique com outro. Para que os asteriscos formem uma matriz triangular, por exemplo, acrescentaríamos ao código um if para quando coluna for maior que linha, fazendo com que o laço pare de ser executado e saia dali para ir à próxima linha do for, externo.

Bem como em outras linguagens, existe um comando no Java, a palavra chave break, que "corta" a execução do laço mais interno, isto é, mais próximo de onde ela mesma se encontra, resultando exatamente no efeito que buscamos:

public class TestaLacos2 {

public static void main(String[] args) {

for(int linha = 0; linha < 10; linha++) {

for(int coluna = 0; coluna < 10; coluna++) {

if(coluna > linha) {

break;

}

System.out.print("\*");

}

System.out.println();

}

}

}Copiar código

Ao salvarmos e rodarmos o código, teremos:

\*

**\*\***

**\*\****\**

***\*\*\*\****

***\*\*\*\*****\**

**\*\*\*\*\*\***

**\*\*\*\*\*\****\**

***\*\*\*\*\*\*\*\****

***\*\*\*\*\*\*\*\*****\**

**\*\*\*\*\*\*\*\*\*\***Copiar código

No exemplo acima, poderíamos obter o mesmo efeito usando a condicional if sem as chaves, pois o break ocupa apenas uma linha, como seria possível também com for e while. No entanto, por boa prática, e visando à legibilidade e convenção, optaremos por usar as chaves sempre que possível.

E no segundo for, poderíamos ter substituído coluna < 10 por coluna <= linha, modificando-se a instrução para não usarmos o break. Assim, o código completo ficaria da seguinte maneira:

public class TestaLacos2 {

public static void main(String[] args) {

for(int linha = 0; linha < 10; linha++) {

for(int coluna = 0; coluna <= linha; coluna++) {

System.out.print("\*");

}

System.out.println();

}

}

}Copiar código

Há muitos exercícios a serem feitos e, mesmo que isso seja trivial para você, que já conhece outra linguagem de programação, ou esteja revendo comandos mais básicos, eles são interessantes para fixar erros de compilação. Senão, quando o conteúdo ficar mais complexo, as chances de se debater por aquilo que já deveria estar bem sedimentado serão maiores.

Portanto, não menospreze a sintaxe básica do Java! Se tiver dúvidas, use nosso fórum, com participação de instrutores e alunos, veja as dúvidas, busque se aprofundar cada vez mais.

Pratique bastante, pois no próximo curso encararemos os desafios de migrarmos da melhor forma de uma programação procedural, imperativa, para a tal da Orientação a Objetos. Muito obrigado!

**11Fixando o comando break**

Clarice está em dúvida sobre o modo como o break funciona quando utilizado dentro de laços de repetição alinhados. Escolha a opção que descreve exatamente o funcionamento desse comando nessas situações.

Parte superior do formulário

* Finaliza a execução do programa no momento em que é chamado.
* Alternativa correta

Para a execução do laço mais interno que contém o comando break e continua executando os laços mais externos.

**Muito bem!** O break irá interromper apenas o laço de repetição mais interno que o contém.

* Alternativa correta

Para a execução de todos os laços de repetição aninhados que o contém e continua normalmente o programa.

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**12Exercitando laços aninhados e break**

Após assistir as aulas de aninhamento de laços de repetição e sobre o comando break, Fernando decidiu criar um código como forma de treinamento que imprime na tela o seguinte:

1

12

123

1234

12345Copiar código

Para isso, ele criou o seguinte código.

public class ProgramaComBreak {

public static void main(String args[]) {

for(int linha = 0; linha < 5; linha++) {

for (int coluna = 0; coluna < 5; coluna++) {

if ( ??? ) {

break;

}

System.out.print( ??? );

}

System.out.println();

}

}

}Copiar código

No entanto, ele está em dúvida sobre como deve preencher os espaços com ???. Selecione a opção que mostra **respectivamente** as formas corretas de preenchimento do código. Se tiver com dúvida, faça o teste e execute o programa.

Parte superior do formulário

* coluna > linha e coluna
* Alternativa correta

coluna > linha e coluna+1

**Excelente, aluno!** Conseguimos assim obter a resposta correta! Observe que precisamos adicionar 1 ao valor da coluna pois o for está inicializando essa variável com o valor 0.

* Alternativa correta

coluna >= linha e coluna

* Alternativa correta

coluna >= linha e coluna+1

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**13Mão na massa: Aprofundando laços**

Vamos um pouco mais a fundo agora.

1) Crie uma classe e seu método main, escolhemos o nome "**TestaLacos**"

2) Escreva um **for** encadeado que imprima a tabuada de cada número, nosso código ficará assim:

public class TestaLacos {

public static void main(String[] args){

for (int multiplicador = 1; multiplicador <= 10; multiplicador++) {

for(int contador = 0; contador <= 10; contador++) {

System.out.print(multiplicador \* contador);

System.out.print(" ");

}

System.out.println();

}

}

}

Copiar código

Repare que usamos o System.out.println(); para organizar melhor a impressão.

3) Vamos criar outra classe e implementar o método main, no curso escolhemos o nome "**TestaLaco2**"

4) Vamos praticar o comando **break** imprimindo uma matriz triangular, esse é o nosso código:

public class TestaLaco2{

public static void main(String[] args){

for (int linha = 0; linha < 10; linha++) {

for(int coluna = 0; coluna < 10; coluna++) {

if (coluna > linha) {

break;

}

System.out.print("\*");

}

System.out.println();

}

}

}Copiar código

Repare que o **break** interrompe a execução do laço, fazendo com que pule para o laço mais externo.

5) Podemos melhorar nosso código, para isso faça as modificações:

public class TestaLaco2 {

public static void main(String[] args){

for (int linha = 0; linha < 10; linha++) {

for(int coluna = 0; coluna <= linha; coluna++) {

System.out.print("\*");

}

System.out.println();

}

}

}Copiar código

Removemos o **if** e mudamos o critério de repetição do **for** interno:

coluna <= linha //restante do forCopiar código

**Opinião do instrutor**

Boa! Não se esqueça de visitar nosso fórum caso tenha dúvidas ou queira ajudar outros alunos!

**14Desafio Opcional: Múltiplos de 3**

Segue mais um exercício para praticar o laço for!

**Utilize um laço do tipo for para imprimir todos os múltiplos de 3, entre 1 e 100.**

Dica: Existem duas maneiras tradicionais de resolver este problema.

Uma delas consiste em fazer o for, e utilizar numero % 3 para descobrir o resto da divisão de um número por 3 (o operador % se chama de *módulo*). Se o resto for zero, ele é divisível por 3. Como em:

if(numero % 3 == 0) {

// faça algo

}Copiar código

Outra abordagem é fazer um laço um pouco diferente, que pula já direto pelos múltiplos de três. Existem outras abordagens, escolha a sua e faça a implementação em uma nova classe!

**Opinião do instrutor**

A abordagem que utiliza o resto da divisão é:

class MultiplosDeTresAteCem {

public static void main (String[] args) {

for (int i = 1; i < 100; i++ ){

if (i % 3 == 0) {

System.out.println(i);

}

}

}

}Copiar código

ou, entre outras tantas opções, a mais simples:

class MultiplosDeTresAteCem {

public static void main (String[] args) {

for (int i = 3; i < 100; i += 3 ){

System.out.println(i);

}

}

}Copiar código

**14Desafio Opcional: Múltiplos de 3**

Nesse exercício opcional, o seu desafio é imprimir os fatoriais de 1 a 10!

Você se lembra do fatorial? Não? Sem problemas, seguem as regras:

* O fatorial de 0 é 1.
* O fatorial de 1 é (0!) \* 1 = 1.
* O fatorial de 2 é (1!) \* 2 = 2
* O fatorial de 3 é (2!) \* 3 = 6
* O fatorial de 4 é (3!) \* 4 = 24
* O fatorial de um número n é n \* n-1 \* n-2 ... até n = 1.

Ou seja:

* O fatorial de 4! = 1 x 2 x 3 x 4 = 24
* O fatorial de 6! = 1 x 2 x 3 x 4 x 5 x 6 = 720

Agora crie uma nova classe, escreva um for que inicie uma variável n (número atual) como 1 e fatorial (resultado total) como 1. Faça seu laço entre 1 a 10 e calcule o resultado!

**Opinião do instrutor**

Segue uma possível implementação:

class Fatorial {

public static void main(String[] args) {

int fatorial = 1;

for (int i = 1; i < 11; i++) {

fatorial \*= i;

System.out.println("Fatorial de " + i + " = " + fatorial);

}

}

}Copiar código

O fatorial é sempre o produto de números consecutivos inteiros de 1 até o próprio número. Ex: Fatorial de 4 = 4 x 3 x 2 x 1 = 24 ou fatorial de 4 = 4 x 3! = 24. Com essa última lógica é que essa questão é feita. Pegando sempre o número que a gente quer e multiplicando pelo fatorial do número anterior.

**16O que aprendemos?**

Neste capítulo aprendemos:

* A sintaxe do while e for
* O operador +=
* O operador ++
* Laços aninhados
* A funcionalidade do break

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**17E agora?**

Nesse curso apresentamos os primeiros passos para você escrever e executar um programa simples com Java e Eclipse IDE, mas claro que o Java oferece muito mais recursos principalmente pensando na **Orientação a Objetos**.

Convido você assistir o próximo curso que fala justamente sobre a Orientação a Objetos com Java. Você aprenderá os principais recursos desse paradigma poderoso sempre aplicado com Java e Eclipse.

Espero que você tenha gostado desse curso e peço avaliar para dar um feedback para gente.

Muito obrigado, Paulo :)

**18Arquivos do projeto atual**

No link abaixo, você encontra o projeto até o momento atual do curso.

<https://github.com/alura-cursos/Curso-Java-parte-1-Primeiros-passos/archive/capitulo7.zip>

**Opinião do instrutor**

Lembre-se de usar o fórum em caso de dúvidas.