# Aula 05-Estruturas de Repetição while e do-while

## Introdução

Olá, meus estudantes, sejam bem-vindos à nossa aula de Algoritmo e Programação em Java do Instituto da Oportunidade Social. Nessa aula, você vai conhecer como implementar estruturas de repetição while e do-while.

## Estruturas de repetição

Estruturas de repetição são necessárias para quando queremos repetir a execução de uma ou mais instruções múltiplas vezes. Desde modo, evitamos o trabalho manual de reescrever o trecho de código repetidas vezes. Por exemplo, para criar um programa para ler 10 notas de alunos sem estrutura de repetição seria assim:

|  |
| --- |
| 1. **import** java.util.Scanner; 3. **public** **class** Main { 4. **public** **static** **void** main(String[] args) { 5. Scanner entrada = **new** Scanner(System.in); 7. **int** n1, n2, n3, n4, n5, n6, n7, n8, n9, n10; 9. System.out.println(" Informe a nota do aluno 1"); 10. n1 = entrada.nextInt(); 12. System.out.println(" Informe a nota do aluno 2"); 13. n2 = entrada.nextInt(); 15. System.out.println(" Informe a nota do aluno 3"); 16. n3 = entrada.nextInt(); 18. System.out.println(" Informe a nota do aluno 4"); 19. n4 = entrada.nextInt(); 21. System.out.println(" Informe a nota do aluno 5"); 22. n5 = entrada.nextInt(); 24. System.out.println(" Informe a nota do aluno 6"); 25. n6 = entrada.nextInt(); 27. System.out.println(" Informe a nota do aluno 7"); 28. n7 = entrada.nextInt(); 30. System.out.println(" Informe a nota do aluno 8"); 31. n8 = entrada.nextInt(); 33. System.out.println(" Informe a nota do aluno 9"); 34. n9 = entrada.nextInt(); 36. System.out.println(" Informe a nota do aluno 10"); 37. n10 = entrada.nextInt(); 38. } 39. } |

Essa abordagem não é eficiente, pois, além de ser muito trabalhosa, torna-se difícil a manutenibilidade do código. Por exemplo, se ao invés 10 alunos tivéssemos 100 ou 1000?

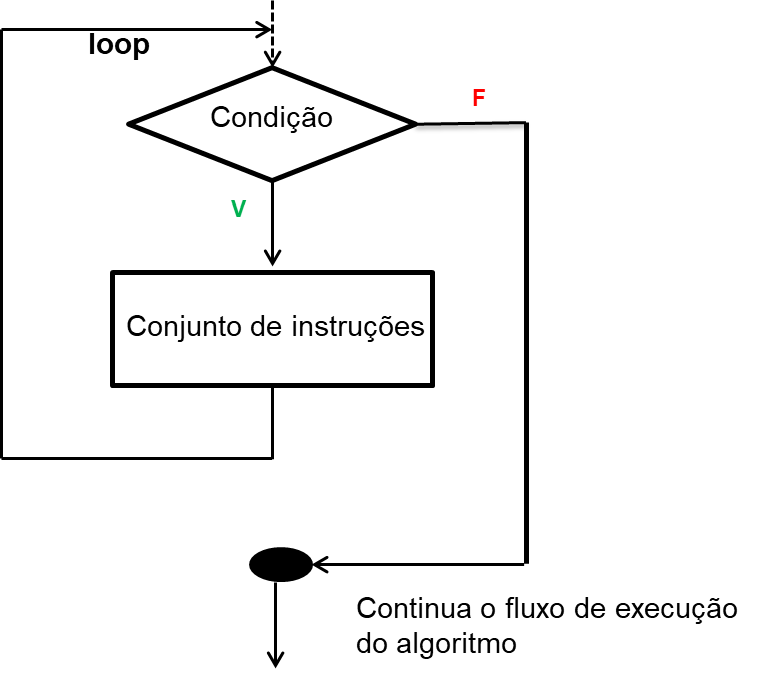
Por isso, existem formas mais eficientes de faz isso, como o uso do laço de repetição while, por exemplo. As vantagens de usar estruturas de repetição são o algoritmo passa a ter um tamanho (número de linha) menor e se torna mais legível e é possível aumentar a escalabilidade do projeto sem fazer grandes alterações no código.

Em programação, podemos fazer dois tipos de repetições:

* Aquela em que sabemos o número de vezes que as instruções serão executadas.
* Aquela em que não sabemos o número vezes que devemos repetir as instruções, pois de alguma condição específica, por exemplo, o usuário digitar uma determinada tecla.

### Comando while

O while (que significa enquanto) é uma repetição com teste no início do comando. A condição é testada a iteração, ou seja, antes de executar novamente as instruções o while verificar se a condição ainda continua verdadeira. Desse modo, o bloco de instruções será repetidamente executado enquanto a condição for verdadeira e, quando a condição for falsa, a execução do bloco de comandos será interrompida.



A sintaxe do comando while é:

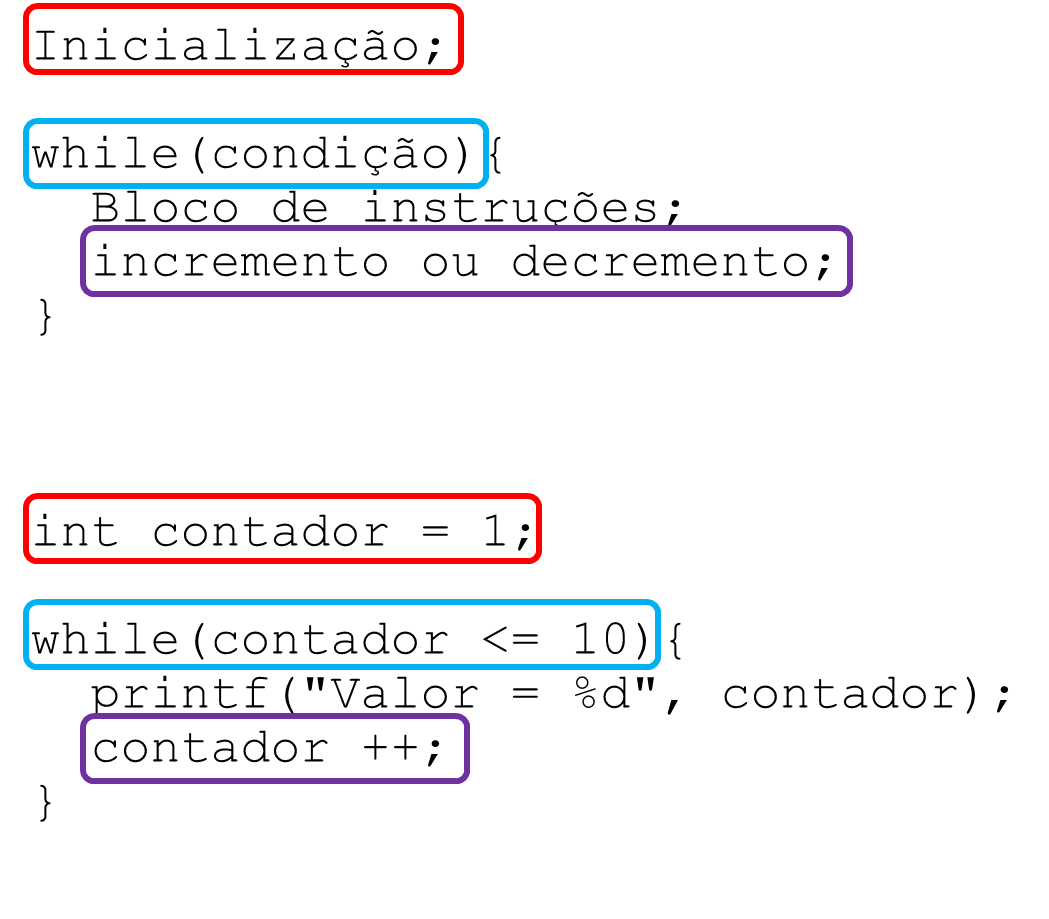
Graphical user interface, text

Description automatically generated

### Contador

Quando queremos repetir a execução de um programa por um número finito de vezes, precisamos criar um contador. O contador nada mais é que uma variável inteira, que é usada para contar a quantidade de vezes que um bloco de instruções é executado repetidamente.

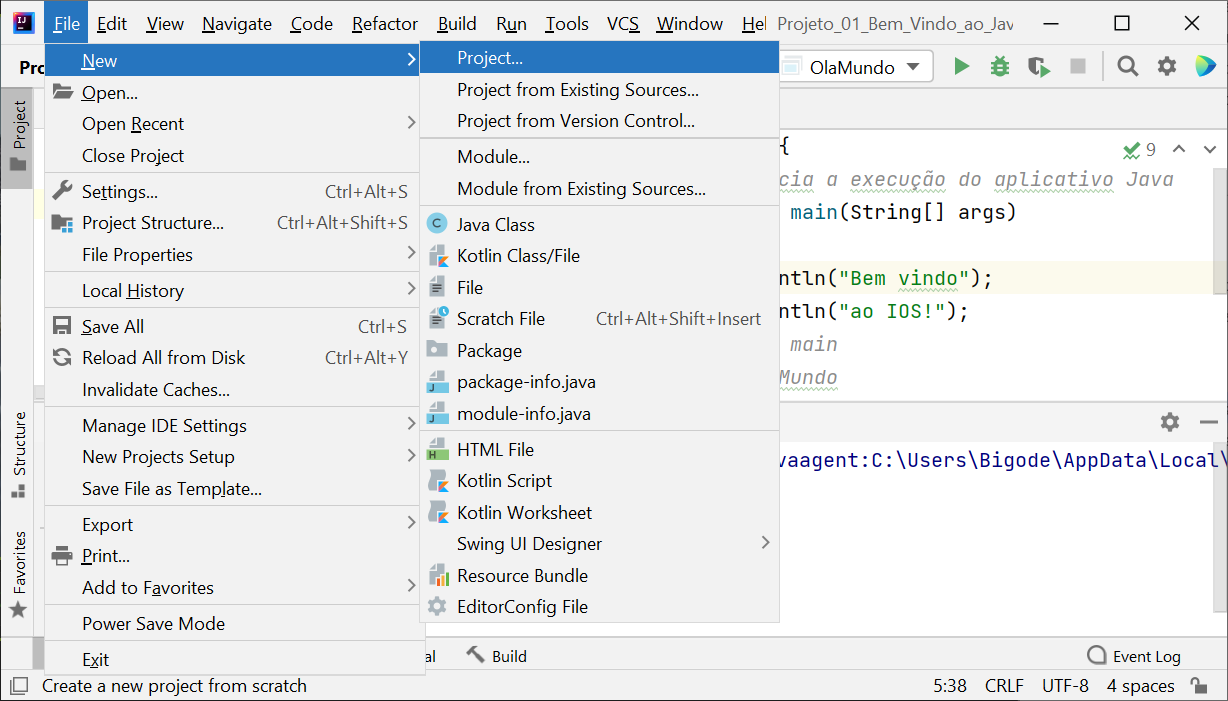
Antes de utilizar o contador, você deve garantir um valor inicial (**fazer a inicialização do contador**) fora do bloco de instruções do **while**. Você também deve garantir que a condição do **while** será falsa em algum momento, para isso, **dentro do bloco de instruções do while**, o contador deverá ser incrementado (seu valor irá aumentar a cada execução do loop) ou decrementado (seu valor irá diminuir a cada execução do loop) de acordo com a lógica implementada.



#### Vamos praticar

Vamos fazer um programa que imprima a sequência dos números inteiros de 1 a 100. Siga os passos para criar o projeto:

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.



1. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_12\_imprimir\_sequencia
2. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Contador e insira o código mostrado na Figura 1.
3. **public** **class** Contador {
4. **public** **static** **void** main(String[] args) {
5. **int** contador; // declarando o contado
6. contador = 1; // inicializando o contador
8. **while** (contador <= 100) {
9. System.out.print(contador + " ");
10. **if**(contador%30 == 0) System.out.println(""); // Quebra de linha para ver todos os números
11. contador++; // mesma coisa que utilizar contador = contador + 1
12. }
13. } // fim do método main
14. } // fim da classe Contador

Figura 1 - Código do programa utilizando while.

1. Ao executar o programa, ele irá imprimir os números de 1 a 100 no terminal.

Text

Description automatically generated with medium confidence

A instrução na linha 4 da Figura 1

contador = 1; // inicializando o contador

é onde inicializamos o nosso contador, ou seja, definimos um valor inicial para ele (no caso o valor inicial é 1).

A condição do comando while

**while** (contador <= 100)

é falsa quando o contador assumir valores acima de 100.

Dentro do bloco de instruções do **while**, na linha 9, temos a instrução de incremento do contador, ou seja,

contador++; // mesma coisa que utilizar contador = contador + 1

Aumenta o valor da variável contador a cada vez que o bloco de instruções é executado. Com isso, garantimos que em algum momento a condição do **while** vai se tornar falsa.

#### Vamos praticar II

Vamos implementar agora um laço de repetição com um contador que tenha seu valor decrementado.

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.

Graphical user interface, text, application, email

Description automatically generated

1. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_13\_while\_decremento
2. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por exemplo: AdivRandom e insira o código mostrado na Figura 2.
3. **public** **class** Decremento {
4. **public** **static** **void** main(String[] args) {
5. **int** contador = 100; // Declara e inicializa o contador
7. **while** (contador > 0) {
8. System.out.print(contador + " ");
9. **if**(contador%30 == 0) System.out.println(""); // Quebra de linha para ver todos os números
10. contador--; // mesma coisa que utilizar contador = contador - 1
11. }
12. } // fim do método main
13. } // fim da classe Decremento

Figura 2 - Código do programa while com o contador sendo decrementado.

1. Ao executar o programa ele irá imprimir os números de 100 a 1 no terminal.

A picture containing calendar

Description automatically generated

A instrução na linha 4 da Figura 1

**int** contador = 100; //  Declara e inicializa o contador

é onde declaramos e inicializamos o nosso contador, ou seja, definimos um valor inicial para ele (no caso o valor inicial é 100).

A condição do comando while

**while** (contador > 0)

é falsa quando o contador assumir valores menores ou iguais a 0 (zero).

Dentro do bloco de instruções do while, na linha 8, temos a instrução de decremento do contador, ou seja,

Contador--; // mesma coisa que utilizar contador = contador - 1

Diminui o valor da variável contador a cada vez que o bloco de instruções é executado. Com isso, garantimos que em algum momento a condição do **while** vai se tornar falsa.

### Acumulador

Podemos também usar laços de repetição para acumular valores dentro de uma variável. Nesse caso, a variável tem o papel de acumuladora, pois ela vai armazenando os valores a cada repetir do loop. Por exemplo, se quisermos calcular o somatório dos números pares de 1 a 100.

#### Vamos praticar

Vamos fazer um programa que imprimir a soma acumulada dos números pares de 1 a 100. Siga os passos para criar o projeto:

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.

Graphical user interface, text, application, email

Description automatically generated

1. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_14\_somatorio
2. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Somatorio e insira o código mostrado na Figura 3.
3. **public** **class** Somatorio {
4. **public** **static** **void** main(String[] args) {
5. **int** contador = 1;
6. **int** acumulador = 0;
8. **while** (contador <= 100) {
9. **if** (contador%2 == 0)
10. acumulador = acumulador + contador; // adiciona o contador par ao acumulador
11. contador++; // incrementa o contador
12. }
13. System.out.println("A soma dos números pares de 1 a 100 é " + acumulador);
14. } // fim do método main
15. } // fim da classe Somatorio

Figura 3 - Código do programa somatório.

1. Ao executar o programa, ele irá imprimir a soma dos números pares de 1 a 100 no terminal.

Graphical user interface, text, application

Description automatically generated

A instrução na linha 4 da Figura 1

**int** acumulador = 0; // Declara e inicializa o acumulador

é onde declaramos e inicializamos o nosso acumulador.

Dica: se nosso acumulador for usado para fazer um somatório, devemos inicializá-lo com o valor neutro da operação de adição, que é 0 (zero). Se nosso acumulador for usado para fazer um produtório, devemos inicializá-lo com o valor neutro da operação de multiplicação, que é 1 (um).

A condição do comando while

**while** (contador <= 100)

é falsa quando o contador assumir valores acima de 100.

Dentro do bloco de instruções do **while**, na linha 7, temos o teste lógico para verificar se o número e par em, na linha 8, a instrução:

acumulador = acumulador + contador; // adiciona o contador par ao acumulador

realiza a soma acumulativa.

### Variável de controle

Variável de controle é usado para controlar as repetições, quando não sabemos o número exato de repetições do laço. Nesse caso, o loop é finalizado assim que a variável tem um determinado valor.

#### Vamos praticar

Vamos fazer um programa que leia o nome e a idade de um grupo de pessoas e mostre o nome da pessoa novamente se ela for maior de idade. Além disso, como não sabemos o total de pessoas vamos perguntar se o usuário deseja continuar ou não a digitar as idades. Assim, a cada iteração pergunte se o usuário se deseja continuar:

* mostre as opções “s” ou “n”
  + caso digite “s” continue o loop;
  + caso digite “n” encerre o loop.

Siga os passos para criar o projeto:

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.

Graphical user interface, text, application, email

Description automatically generated

1. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_15\_var\_control
2. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Controle e insira o código mostrado na Figura 4.
3. **import** java.util.Scanner;
5. **public** **class** Controle {
7. **public** **static** **void** main(String[] args) {
9. Scanner entrada = **new** Scanner(System.in);
11. **char** resp; // declara variavel de controle
12. String nome;
13. **int** idade;
14. resp = 's'; // inicializa a variavel de controle
16. **while** (resp == 's') {
17. System.out.println("Digite o seu nome:");
18. nome = entrada.next();
19. System.out.println("Digite sua idade:");
20. idade = entrada.nextInt();
22. **if** (idade >= 18) {
23. System.out.println("Seu nome eh: " + nome);
24. }
26. System.out.println("Deseja continuar ? (s/n):");
27. resp = entrada.next().charAt(0);
28. }
29. entrada.close();
30. } // fim do método main
31. } // fim da classe Controle

Figura 4 - Código do programa Controle.

1. Ao executar o programa, você pode digitar quantos nomes e idades quiser até selecionar n para parar o laço de repetição.

Graphical user interface, text, application

Description automatically generated

A instrução na linha 7 da Figura 1

**char** resp; // declara variavel de controle

é onde declaramos a variável de controle. E na linha 10, a instrução:

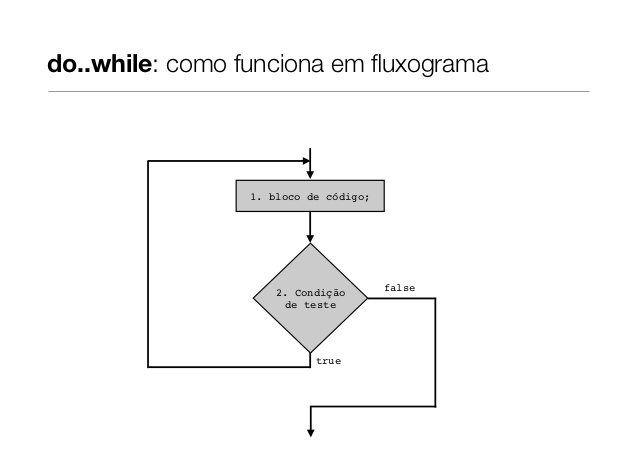
resp = 's'; // inicializa a variavel de controle

é onde inicializamos com o valor ‘s’, que torna a condição do while (linha 14) verdadeira.

Na linha 25, lemos a resposta do usuário se ele deseja continuar ou não, enquanto ele digitar ‘s’ o loop continua repetindo as instruções e quando o usuário digitar ‘n’ o loop é encerado.

### Comando do ... while

Esse tipo de estrutura de repetição é caracterizado por fazer o teste de controle no final do bloco de comandos. Os comandos repetidos são executados pelos menos uma vez antes da condição ser testada. A condição é testada no final e sempre após a execução do bloco do laço.



A sintaxe do comando do-while é:

A picture containing line chart

Description automatically generated

#### Vamos praticar

Vamos criar um programa para ler diversas notas e faça a soma acumulada das notas digitadas. O loop deverá se repetir enquanto a nota digitada for maior ou igual a zero e menor ou igual a dez, caso contrário o loop deverá ser encerrado e a soma das notas deverá ser exibida. Siga os passos para escrever o programa utilizando o switch-case:

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.
2. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_16\_do\_while
3. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por ex.: Notas e insira o código mostrado na Figura 5.
4. **import** java.util.Scanner;
6. **public** **class** Notas {
7. **public** **static** **void** main(String[] args) {
8. Scanner entrada = **new** Scanner(System.in);
9. **double** nota, soma = 0;
10. **do** {
11. System.out.println(" \*\*\*\* Digite uma nota \*\*\*\*\* ");
12. nota = entrada.nextDouble();
14. **if** (nota >= 0 && nota <= 10)
15. soma += nota;
17. } **while** (nota >= 0 && nota <= 10);
18. System.out.println("A soma das notas é:  " + soma);
19. entrada.close();
20. } // fim do método main
21. } // fim da classe notas

Figura 5. Código utilizando o do-while.

1. O resultado do código pode ser visto ao compilar e executar o código, clicando com o botão direito na classe e selecionando a opção Run.

Graphical user interface, text, application

Description automatically generated

Na linha 14, o do-while testa a condição para ver se repete o bloco de instruções ou se encerra o loop, ou seja, o do-while executa pelo menos uma vez o bloco de instruções antes de testar a condição de repetição.

## Misturando o aprendizado

Vamos dar um upgrade no programa de adivinhações com valor randômico. Agora o programa só irá parar de executar quando você acertar o valor sorteado. E no final ele irá imprimir o número total de tentativas. Siga os passos para criar o projeto:

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.

Graphical user interface, text, application, email

Description automatically generated

1. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_17\_adivinha\_up
2. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por exemplo: AdivinhaUP e insira o código mostrado na Figura 6.
3. **import** java.util.Scanner; // Import necessário para utilizar a classe Scanner
4. **import** java.util.Random; // Import necessário para utilizar a classe Random
6. **public** **class** AdivinhaUP {
7. **public** **static** **void** main(String[] args) {
8. // Instância (buffer) utilizando a classe Scanner
9. Scanner entrada = **new** Scanner(System.in);
10. // Instância utilizando a classe Random
11. Random rand = **new** Random();
12. **int** palpite, num = 0;
13. num = rand.nextInt(10) + 1;
15. **do**{
16. System.out.println("Digite seu palpite:");
17. palpite = entrada.nextInt();
18. **if** (palpite == num) {
19. System.out.println("Você acertou!!!");
20. } **else** {
21. **if** (palpite < num) {
22. System.out.println("Seu palpite está abaixo.");
23. } **else** {
24. System.out.println("Seu palpite está acima.");
25. }
26. }
27. }**while**(palpite != num);
29. entrada.close();
30. } // fim do método main
31. }

Figura 6 - Código do upgrade do programa adivinhação.

1. Ao executar o programa, você pode dar um palpite até acertar o número sorteado.

Graphical user interface, text, application

Description automatically generated

## Exercícios de revisão

1. Escreva um algoritmo que imprima a sequência do número 100 ao 200.
2. Faça um algoritmo que leia 10 números inteiros e imprima:
   * quantos são pares;
   * e quantos são ímpares.
3. Construir um algoritmo que leia um número inteiro e imprime a sequência:
   * 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024. . .
   * enquanto o valor da sequência for menor ou igual ao número lido.
4. Faça um algoritmo que imprima a metade de cada número de 10 a 20.
   * Exemplo:
     + a metade de 10 é 5.0
     + a metade de 11 é 5.5
     + a metade de 12 é 6.0

. . .

* + - a metade de 18 é 9.0
    - a metade de 19 é 9.5
    - a metade de 20 é 10.0

1. Faça um algoritmo que imprima a tabuada do 5.
2. Faça um algoritmo que leia dez números inteiros e mostre o menor entre eles.
3. O IMC (índice de Massa Corporal) é uma medida do grau de obesidade de uma pessoa.
   * Faça um algoritmo que leia a altura e o peso de 10 pessoas.
   * Calcular o IMC de cada pessoa e vericar quantas pessoas estão com o IMC entre 18,5 e 24,9 que é considerado sem obesidade.

Text

Description automatically generated