Domain Driven Design with Java

Iteration Structures

Dr. Gustavo Molina

Outline for Section 1

Estruturas de Repetição

- Um Loop qualquer é uma instrução que faz o programa executar um bloco ou pedaço de código diversas vezes, enquanto uma condição de saída não for satisfeita.
- Servem instruir o programa a executar diversas vezes uma tarefa qualquer.
- Em java, é possível utilizar os loops for, while e do-while.
- Hoje aprenderemos os loops *while* e *do-while.*

Outline for Section 2

WHILE

- O loop *while* executa o bloco de código entre chaves apenas enquanto a sua condição de satisfação seja verdadeira (true);
- ex: "Enquanto não houver obstáculos a sua frente, ande um passo de cada vez".
- "Enquanto houver comida no prato, coma".
- "Enquanto não chegar ao seu destino, dirija".
- "Enquanto estiver debaixo da água, não respire".

While, exemplo 1

- Exemplo:

```
public class Main {
  public static void main(String[] args) {
    int i = 0;
    while (i < 5) {
      System.out.println(i);
      i++;
    }
  }
}
```

While, exemplo 1

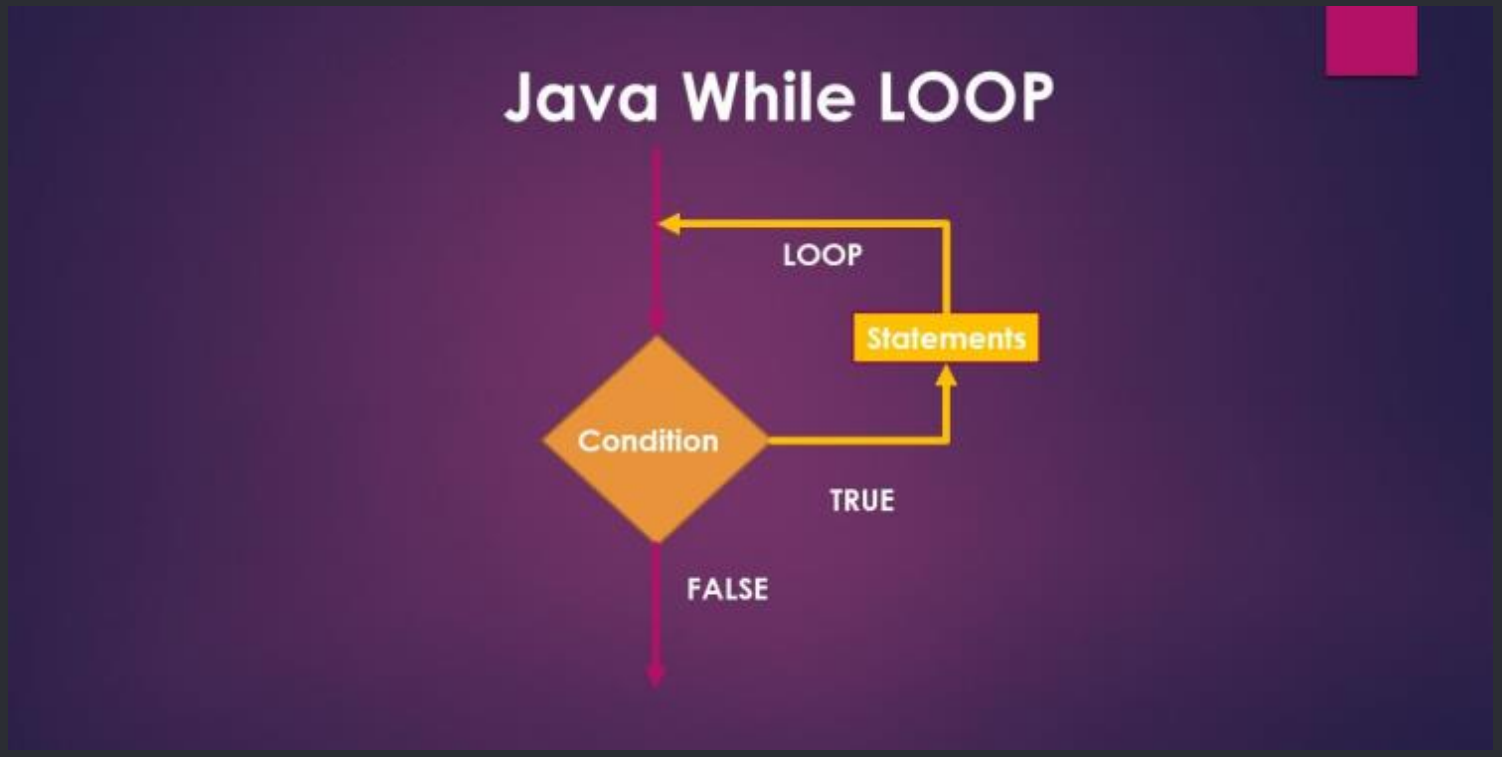

Figure: Instrução while

Loop Do-While

- O loop do/while é uma variante do loop while.
- Este loop executará o bloco de código uma vez, antes de verificar se a condição é verdadeira, então repetirá o loop enquanto a condição for verdadeira.

do-while: Exemplo

```
int i = 0;
do {
   System.out.println(i);
   i++;
}
while (i < 5);
```

do-while

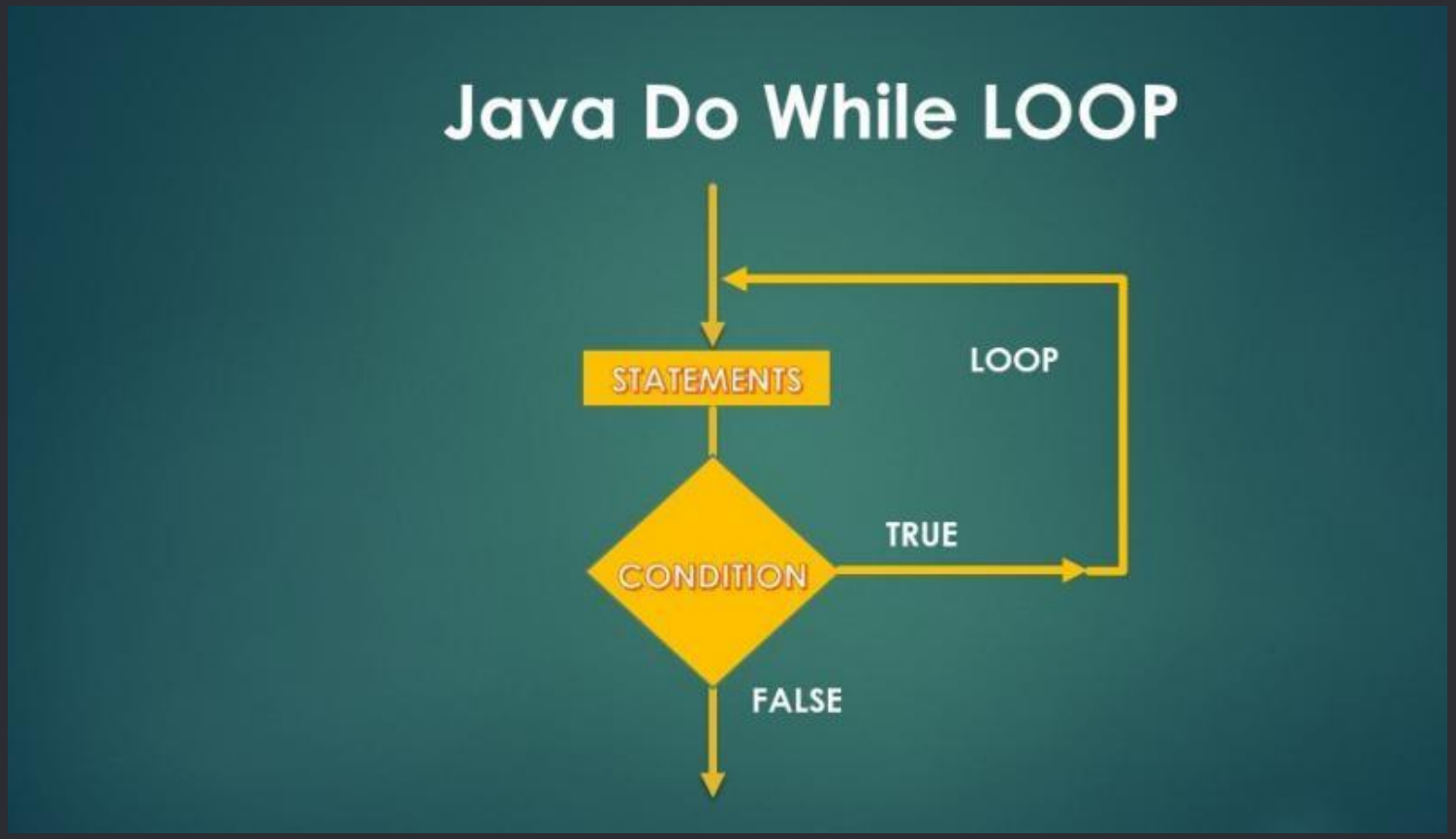

Figure: Instrução do-while

while vs do-while

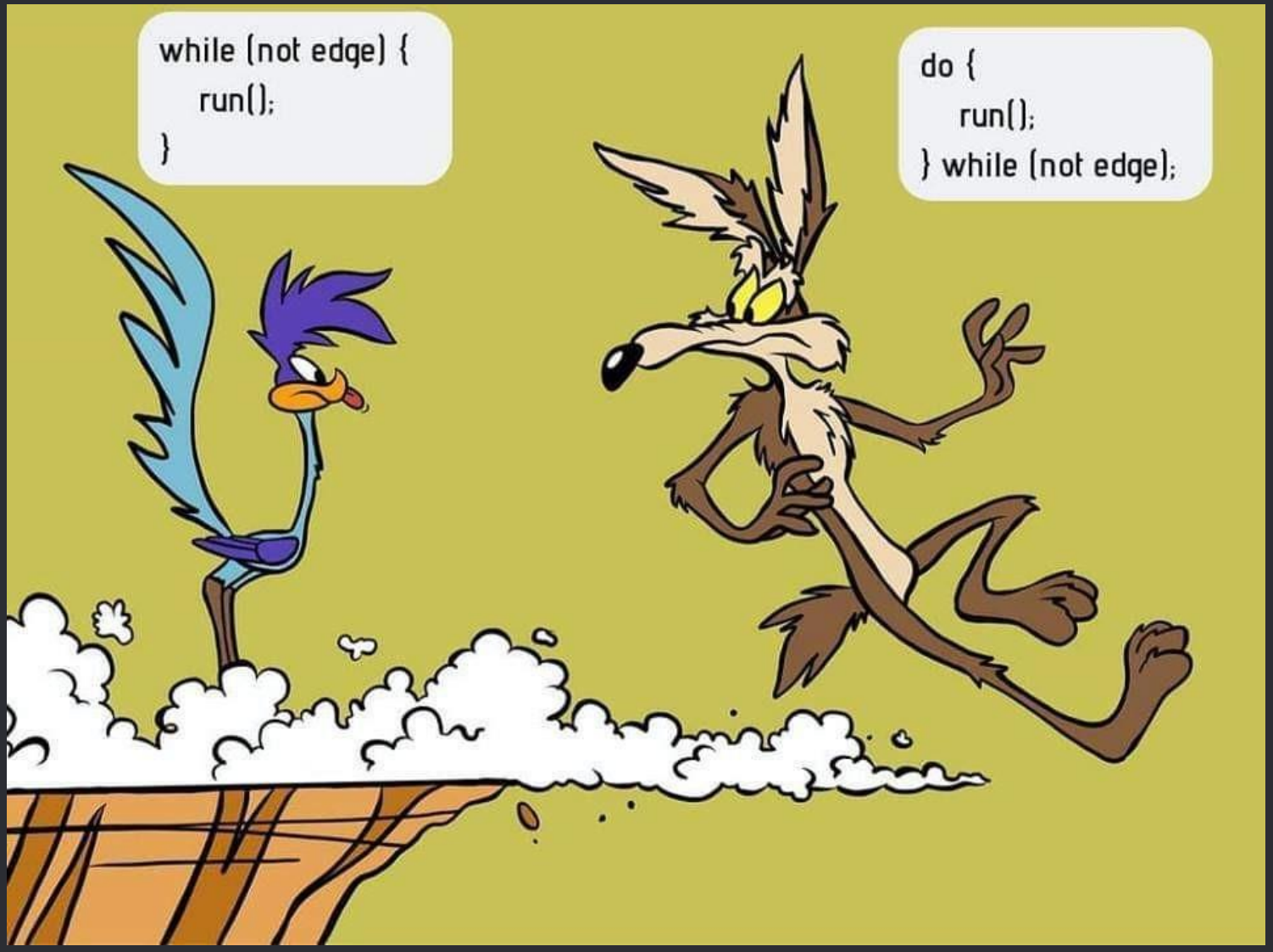

Figure: while vs do-while

Instrução Break

- A instrução *break* faz com que o programa saia imediatamente do loop;
- Bastante útil para evitar loops infinitos.

break, exemplo

- Exemplo:

```
public class Main {
  public static void main(String[] args) {
    int i = 0;
    while (i < 5) {
      if( i == 2 )
        break;
      System.out.println(i);
      i++;
    }
  }
}
```

Instrução Continue

- A instrução *continue* faz com que o programa interrompa o loop por um instante, continuando imediatamente para a próxima Iteração.
- Bastante útil para evitar controlar quais sequencias serão analisadas / descartadas.

continue, exemplo

- Exemplo:

```
public class Main {
  public static void main(String[] args) {
    int i = 0;
    while (i < 5) {
      if( i == 2 )
        continue;
      System.out.println(i);
      i++;
    }
  }
}
```

Continue / break

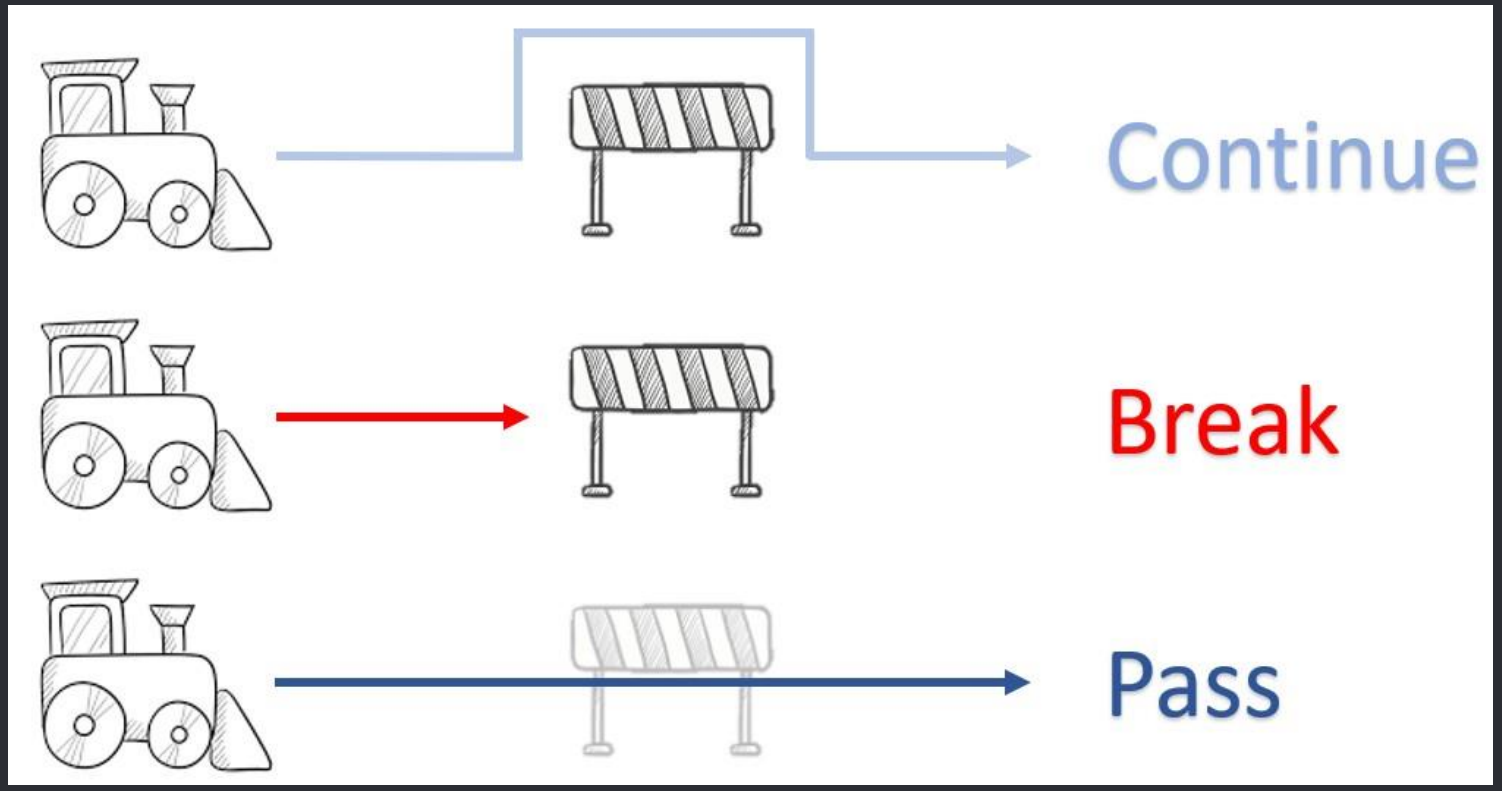

Figure: continue vs break

Continue / break

Figure: continue vs break

Outline for Section 3

1. Introdução
2. O loop *while*
3. Exercícios
4. O loop *For*
5. Exercícios
6. Referencias - Para se aprofundar mais

Estruturas de Repetição

- Quando Você sabe exatamente quantas vezes você quer executar o loop ou iteração de um bloco de código, use o loop FOR ao invés de um loop While:

Outline for Section 4

Loop For

- O loop *for* executa o bloco de código entre chaves baseada no número de iterações desejada podendo inclusive, variar a quantidade de passos;
- ex: "Imprima do numero 0 ao 1 milhão de dois em dois"
- "Imprima todos os números de uma coleção de números"
- "Execute determinado bloco de código 10 vezes"

For, exemplo 1

- Exemplo:

```
public class Main {
  public static void main(String[] args) {
    for (statement 1; statement 2; statement 3) {
        // code block to be executed
    }
  }
}
```

For, explicado

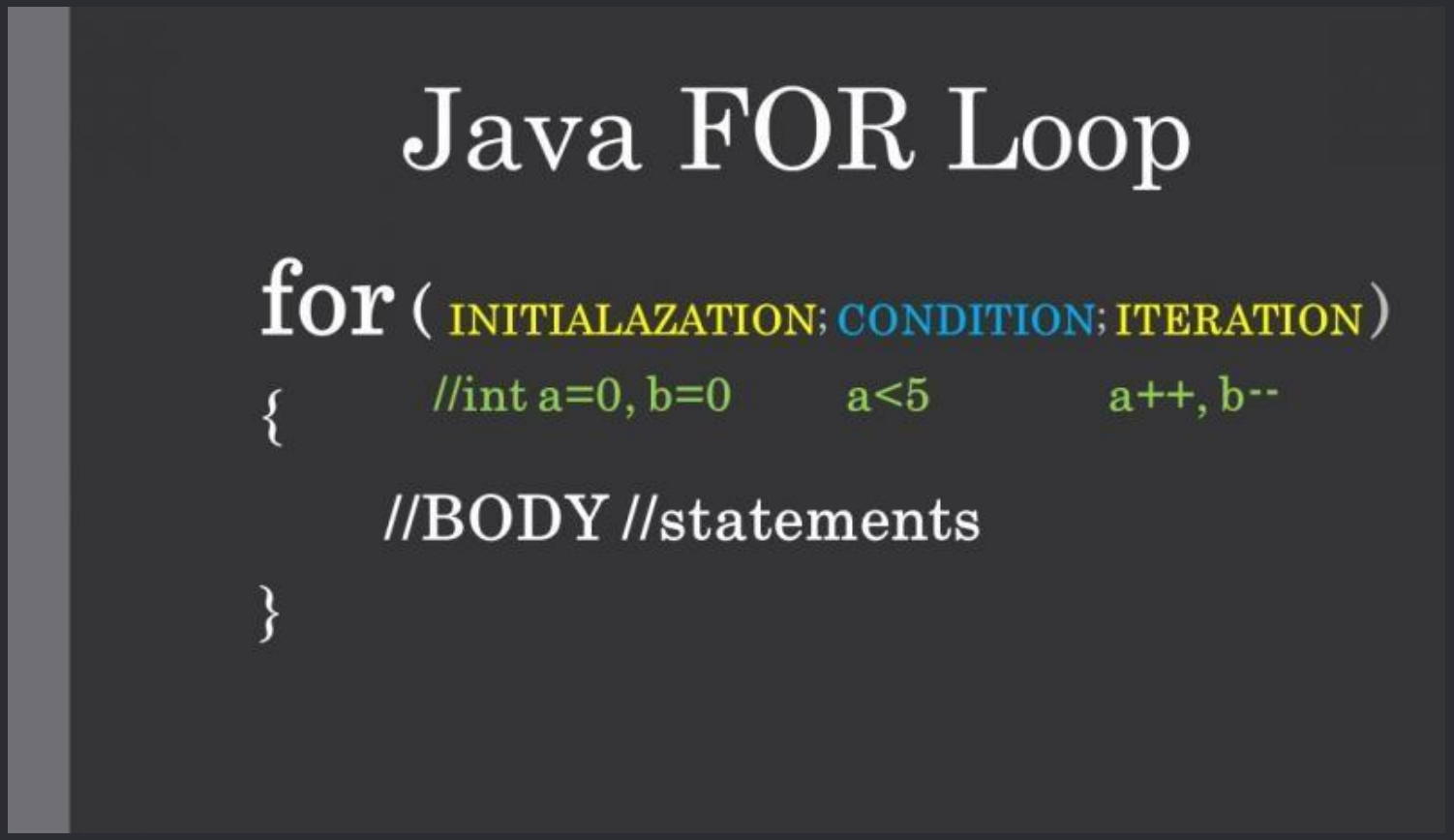

Figure: O Loop For

For: Exemplo1

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

Cuidados com For: Exemplo 2

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

Instrução Break

- A instrução *break* faz com que o programa saia imediatamente do loop;
- Bastante útil para evitar loops infinitos

break, exemplo

- Exemplo:

```
public class Main {
  public static void main(String[] args) {
   for (int i = 0; i <= 10; i++) {
         System.out.println(i);
          if( i == 2 )
             break;
    }
  }
}
```

Instrução Continue

- A instrução *continue* faz com que o programa interrompa o loop por um instante, continuando imediatamente para a próxima iteração
- Bastante útil para evitar controlar quais sequencias serão analisadas / descartadas;

continue, exemplo

- Exemplo:

```
public class Main {
  public static void main(String[] args) {
    for (int i = 0; i <= 10; i++) {
      if( i == 2 )
        continue;
      System.out.println(i);
    }
  }
}
```

Continue / break

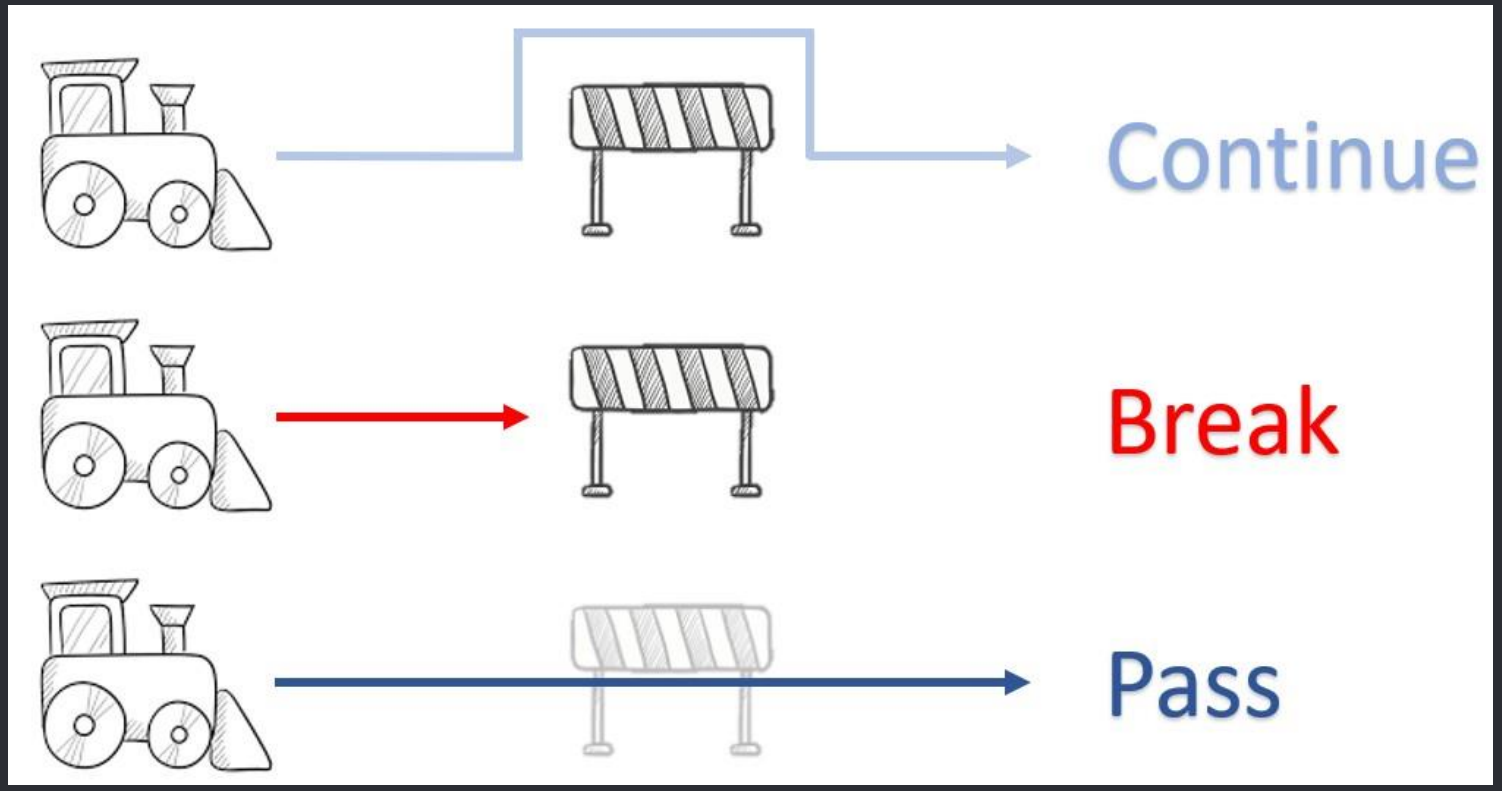

Figure: continue vs break

Continue / break

Figure: continue vs break

Outline for Section 5

Exercícios de fixação

1. Escreva um programa que leia várias frases que o usuário possa escrever no console ou terminal, quando o usuário escrever "PARE", interrompa o programa.
2. Escreva um programa que receba dois valores: um valor de início e um valor de final, inteiros. Escreva um programa que imprima na saída padrão todos os números no intervalo inicial até o final, inclusive.

Exercícios de fixação

3. Escreva um programa que imprima na saída padrão todos os números de 0 a 1000 que são pares;

4. Escreva um programa que imprima na saída padrão todos os números de 0 a 1000 que são divisíveis por 5;

5. Escreva um programa que permita que o usuário digite vários números e imprima na saída padrão somente o primeiro e o ultimo números digitados pelo usuário

Exercícios de fixação

6. Escreva um programa que imprima de um intervalo de números de 0 a 1000, apenas um a cada 6: ex: 0,6, 12;
7. Escreva um programa que que imprima os números de um intervalo inserido pelo usuário, de maneira que cada número seja multiplicado pelo número anterior: Ex: usuário entra com 1 e 3: programa calcula 1, 2*1, 3*2, etc.
8. Escreva um programa que gere a tabuada de qualquer número de zero a 10;

Outline for Section 6

1. Introdução
2. O loop *while*
3. Exercícios
4. O loop *For*
5. Exercícios
6. Referencias - Para se aprofundar mais

Para se aprofundar...

- DEITEL, Harvey M. et al. Java: como programar. 2016.
- HORSTMANN, Cay S. Big Java: early objects. John Wiley & Sons, 2016.
- GOLDMAN, Alfredo; KON, Fabio; SILVA, Paulo JS. Introdução à Ciência da Computação com Java e Orientação a Objetos. Editado e Revisado por Raphael Y. de Camargo, v. 1.

Perguntas?

Bora! Perguntas?