



ORACLE

Academy



Java Foundations

2-2

O que meu Programa Está Fazendo?

ORACLE
Academy



Objetivos

- Esta lição abrange os seguintes objetivos:
 - Entender como o Java é lido linha por linha
 - Configurar e usar pontos de interrupção
 - Terminar instruções com pontos e vírgulas (;)
 - Organizar o código usando espaço em branco e outras convenções
 - Criar comentários



Lendo um Programa Linha por Linha

- Cada linha em um programa é lida uma vez

```
1 System.out.println("Linha 1");  
2 System.out.println("Linha 2");  
3 System.out.println("Linha 3");  
4 System.out.println("Linha 4");  
5 System.out.println("Linha 5");
```

- Neste exemplo...
 - A linha 1 é lida...
 - Depois a linha 2...
 - Depois a linha 3...
 - Depois a linha 4...
 - Depois a linha 5...

Lendo Linha por Linha

- O Java é lido principalmente linha por linha
- Mas existem alguns pontos adicionais a serem considerados
- Vamos investigar usando...
 - Um ponto de interrupção
 - Outros recursos de um IDE Java





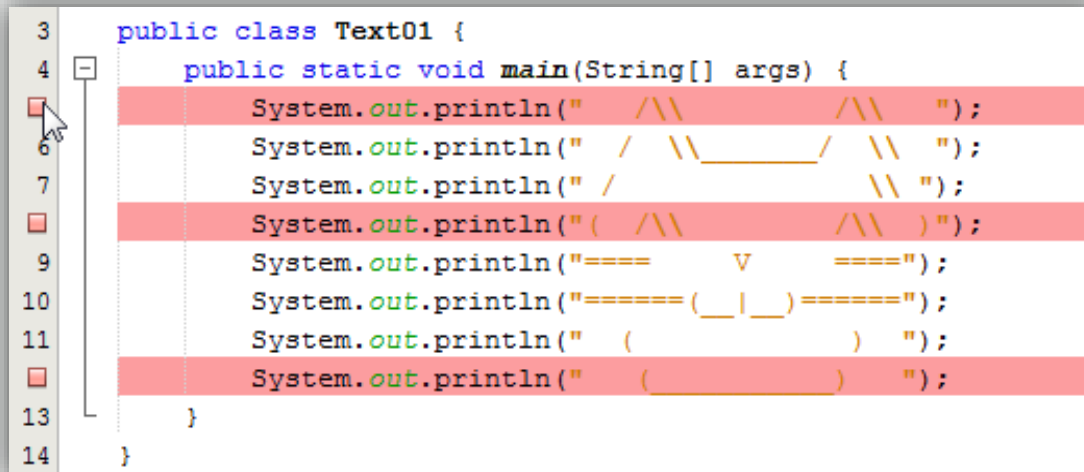
Pontos de interrupção

- Definir um ponto de interrupção no seu código para
 - Pausar a execução do código
 - Verificar o estado atual do programa
 - Ajudar na depuração
- Os pontos de interrupção afetam a execução do código...
 - Quando o código é executado com o depurador
- Os pontos de interrupção não podem afetar a execução do código...
 - Quando o código é executado normalmente



Definindo a Animação de um Ponto de Interrupção

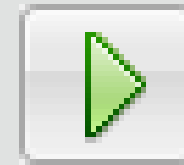
- Para definir um ponto de interrupção...
 - Posicione o cursor sobre um número na margem esquerda
 - Clique em ... e você tem um ponto de interrupção!
 - Clique novamente para remover um ponto de interrupção
 - Você pode definir vários pontos de interrupção



```
3 public class Text01 {  
4     public static void main(String[] args) {  
5         System.out.println("  /\\"  
6         System.out.println("  /  \\"  
7         System.out.println("  /      \\"  
8         System.out.println(" (  /\\"  
9         System.out.println("====  V  =====");  
10        System.out.println("===== ( _ | _ ) =====");  
11        System.out.println("  (              )  ");  
12        System.out.println("  (              )  ");  
13    }  
14 }
```

Exercício 1, Parte 1

- Crie um novo projeto e adicione o arquivo `Text01.java` a ele
- Defina um ponto de interrupção na Linha 5 (a linha com a primeira instrução de impressão)
- Execute o programa normalmente
 - Os pontos de interrupção não devem interferir no programa

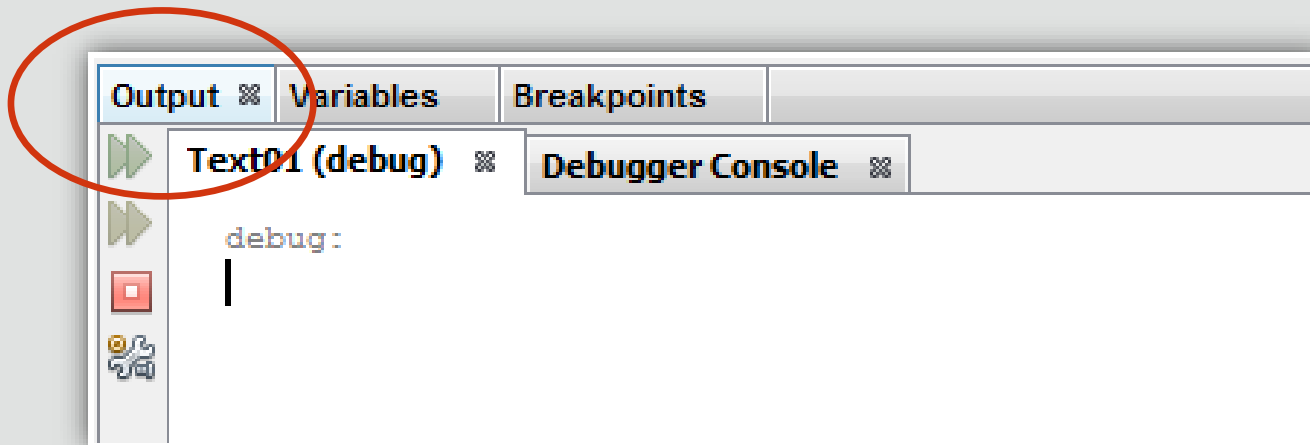


Exercício 1, Parte 2

- Execute o programa com o depurador:
 - Certifique-se de que a janela Output esteja em exibição
 - Pressione Step Over para passar para a linha seguinte
- Observe o gato que aparece em uma linha por vez



Step Over



Exercício 1, Parte 3

- Modifique o código para que as primeiras três instruções de impressão apareçam na Linha 5 (Isso denomina-se remover espaço em branco)
- Execute o programa com o depurador:
 - Certifique-se de que a janela Output esteja em exibição
 - Pressione Step Over Expression para passar para a linha seguinte
 - Ignore o código complicado no fim da depuração
- Observe o gato que aparece em uma linha por vez
- Tente remover um ponto e vírgula ao depurar o programa



Step Over
Expression

E se você fizer isso antes de depurar?



Resultados da Investigação, Parte 1

- Você poderia dizer que o Java lê linha por linha...
- Mas, se houver várias instruções em uma única linha, seria mais correto dizer que o Java lê instrução por instrução
- É necessário um ponto e vírgula (;) para terminar uma instrução
 - Um erro bem comum é esquecer o ponto e vírgula
 - Em outras linguagens, como Python, o ponto e vírgula não é tão importante

```
System.out.println( "Miau" ) ;
```

- A edição do código não tem efeito enquanto o programa está em execução
- Você precisa recompilar para que as alterações sejam implementadas



Resultados da Investigação, Parte 2

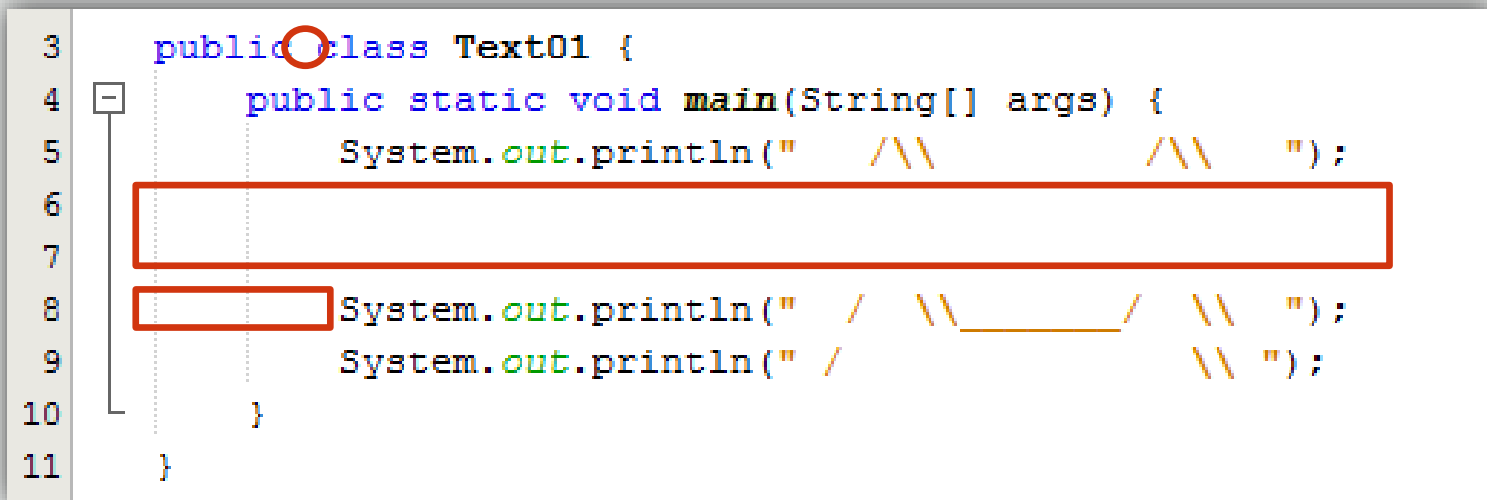
- O Java não é exato em relação ao espaço em branco
- Outras linguagens como Python podem ser extremamente precisas
- Você poderia escrever um programa inteiro em uma única linha
 - Mas seria confuso e quase impossível usá-lo
 - Use o espaço em branco para manter o código organizado

```
3 public class Text01 {
4     public static void main(String[] args) {
5         System.out.println("    /\ \    /\ \    "); System.out.println(" /  \ \_____/  \ \    ");
6         System.out.println(" /
7         System.out.println("( /\ \    /\ \    )");
8         System.out.println("==== V      ===");
9         System.out.println("===== ( _ | _ ) =====");
10        System.out.println(" (              ) ");
11
12        System.out.println(" (              ) ");
13    }
14 }
```

*Esse código funciona...
mas é muito confuso*

Espaço em branco

- O espaço em branco é qualquer espaço sem código:
 - Espaço entre palavras
 - Linhas em branco
 - Recuo antes de uma linha de código



```
3 public class Text01 {
4     public static void main(String[] args) {
5         System.out.println("    /\\"      /\\"    ");
6
7
8         System.out.println(" /  \\"_____/  \\" ");
9         System.out.println(" /                      \\" ");
10    }
11 }
```

Efeitos do Espaço em Branco

- O espaço em branco ajuda a manter o código organizado
- O espaço em branco não afeta o modo como o código é executado
- Você pode usar o espaço em branco da maneira que preferir
- Mas o recuo apropriado é bastante recomendado porque ele...
 - Facilita a leitura
 - Evita erros durante a programação

Ahh! Código desorganizado!



Recuo e Chaves

- Insira um recuo de mais uma tabulação (4 espaços) após uma chave de abertura ({)
- Pare o recuo de mais uma tabulação (4 espaços) antes de uma chave de fechamento (})
- O código dentro de chaves denomina-se bloco de código
 - Quando adicionar uma chave de abertura ({) ...
 - Você acabará precisando de uma chave de fechamento (})
 - Um erro comum é incluir uma chave sem correspondência ou esquecer de incluir uma chave

Exemplo de Animação em Bloco

```
public class Example
{
    public static void main(String[] args){
        System.out.println("Inner code");
        System.out.println("Inner code");
        {
            System.out.println("Inner-inner code");
        }
    }
}
```

Essas chaves também criam um bloco dentro de outro...

cujo código é ainda mais recuado

Ajuda de Recuo do IDE

- Um IDE pode...
 - Codificar com cores o escopo de um bloco (Greenfoot, BlueJ)
 - Definir automaticamente um recuo logo após uma chave
 - Realçar uma chave correspondente (como mostrado abaixo)
- Alguns comandos Java requerem chaves, embora você possa sempre adicionar mais

```
public class Example
{
    public static void main(String[] args){
        System.out.println("Inner code");
        System.out.println("Inner code");
        {
            System.out.println("Inner-inner code");
        }
    }
}
```

Exercício 2

- Crie um novo projeto e adicione o arquivo `Text02.java` a ele
- Você consegue corrigir esse programa e produzir a saída a seguir?

```
1
2
3
4
```

- Dicas:
 - Seu IDE sublinha o código que está com problema
 - Seu IDE pode realçar chaves correspondentes
 - Seu IDE tem um atalho para recuo correto

Comentários

- Os programas com espaços bem organizados podem ficar muito grandes e tornarem-se difíceis de ler
- Você pode adicionar comentários ao código para...
 - Fornecer uma explicação ou informações adicionais ao programador (Código de comentário)
 - Desativar o código e impedir que ele seja executado sem apagá-lo (Código marcado para ser ignorado)

Ahh! O que todo esse código está fazendo?



Adicionando Comentários ao Código

- Comentários em uma única linha...
 - Começar com duas barras //
 - Terminar quando a linha termina

```
//Um comentário em uma única linha termina automaticamente  
//quando a linha termina
```

```
System.out.println("Esta linha é impressa");
```

Adicionando Comentários ao Código

- Comentários em várias linhas...
 - Começar com uma barra e um asterisco /*
 - Terminar com um asterisco e uma barra */

```
/*Um comentário em várias linhas...  
continua por muitas linhas...  
System.out.println("Esta linha não é impressa");  
até aparecer um asterisco com uma barra*/  
System.out.println("Esta linha é impressa");
```

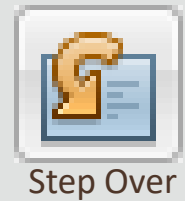
Lendo Linha por Linha

- Podemos analisar um pouco mais o código
- Vamos investigar usando...
 - Blocos de código
 - Comentários
 - Pontos de interrupção
 - Outros recursos de seu IDE



Exercício 3

- Crie um novo projeto e adicione o arquivo `Text03.java` a ele
- Defina um ponto de interrupção na Linha 11
- Execute o programa com o depurador:
 - Certifique-se de que a janela Output esteja selecionada
 - Pressione Step Over para passar para a linha seguinte
- Observe que o rosto do gato aparece, mas as patas não
- Digite `drawLegs();` na Linha 19 e depure o programa
 - Onde você poderia adicionar um ponto de interrupção para ver as patas desenhadas uma linha por vez?
 - O que acontece com a saída quando comentários são adicionados às linhas?





Resultados da Investigação, Parte 3

- Quando o Java lê linha por linha...
- Ele começa dentro do bloco de código especial conhecido como método main

```
public static void main(String[] args){  
  
} //método final main
```

- Nenhum outro código é executado a menos que seja chamado
 - Neste exercício, o método main deve chamar especificamente o bloco de código que imprime as patas
- O código comentado é ignorado
 - Os comentários são removidos no código de bytes

O Fluxo do Programa

1. Todos os programas Java começam no método main
2. Nenhum outro código é executado a menos que seja chamado

2) E depois passe para cá

1) Comece aqui

```
public class Text03 {  
    public static void drawLegs() {  
        System.out.println("    ||    ||    ");  
        System.out.println("    ||    ||    ");  
        System.out.println("   (||)   (||)   ");  
  
        public static void main(String[] args) {  
            System.out.println("  /\\"  
            System.out.println(" /  \\"  
            System.out.println(" /      \\"  
            System.out.println("(  /\\"  
            System.out.println("====  V  ===");  
            System.out.println("===== ( _|_ ) =====");  
            System.out.println(" (              ) ");  
            System.out.println(" (              ) ");  
            drawLegs();  
        }  
    }  
}
```

O Método Main

- O método main é um bloco de código especial
- Todos os programas Java começam no método main
- Seus programas só devem ter um método main
- Os métodos serão analisados mais detalhadamente na próxima lição
 - drawLegs() é um exemplo de método

```
public static void main(String[] args){  
    //Seu programa começa aqui  
} //fim do método main
```

Resumo

- Erros comuns:

- Ponto e vírgula ausente (;)

```
System.out.println("Miau")
```

- {Chaves} sem correspondência

```
{  
    System.out.println("Meow");
```

- Mantenha o código organizado usando:

- Espaço em branco
- Chaves ({ })
- Comentários

Resumo

- Nesta lição, você deverá ter aprendido a:
 - Entender como o Java é lido linha por linha
 - Configurar e usar pontos de interrupção
 - Terminar instruções com pontos e vírgulas (;)
 - Organizar o código usando espaço em branco e outras convenções
 - Criar comentários





ORACLE

Academy

