

# PROGRAMAÇÃO ORIENTADA A OBJETOS

---

Introdução a Aplicativos Java

Prof. Emanuel Barreiros



# Java

- O Java foi concebido como uma linguagem multiplataforma
- Inicialmente, foi projetada para ser uma linguagem que executaria em dispositivos eletrodomésticos (geladeiras, cafeteiras, torradeiras, etc.)
- Ganhou popularidade e se difundiu em outros domínios

# Bibliotecas (APIs) Java

- Você pode criar todas as classes e métodos que precisar, mas nem sempre é necessário ou indicado
- Existe uma vasta biblioteca de classes que já vem incluída no Java
- Tais bibliotecas são conhecidas como APIs (Application Programming Interfaces)
- Dica: sempre que possível use classes e métodos fornecidos pelo Java.
- API do Java: <http://docs.oracle.com/javase/7/docs/api/>

# Criando um programa em Java

- Passos:

1. Crie um arquivo com código Java (qualquer editor de texto padrão serve)
2. Salve-o no seu disco com a extensão **.java**
3. Compile o código usando o **javac**
  1. Ex: `javac HelloWorld.java`
  2. O sistema criará um arquivo com a extensão **.class** com o código compilado (bytecodes)
4. Execute o código compilado
  1. Ex: `java HelloWorld` (sem o **.class** da extensão)

# Problemas que podem acontecer

- Erros de compilação
  - Erros sintáticos, que impedem que o código seja compilado
  - Erros sintáticos significam que você fez alguma coisa que feriu a sintaxe da linguagem, ou seja, sua estrutura e/ou regras de escrita
  - Erros de compilação são reportados pelo compilador, que tenta ajudar o programador dando alguma mensagem que o ajude a corrigir a falha
- Erros de execução
  - Erros que não podem ser previstos em tempo de execução
    - Divisão por zero
    - Conexão negada com um servidor
    - Abrir um arquivo que não existe
    - Etc.

# Nosso primeiro programa Java

```
1
2 ▼ public class HelloWorld {
3
4 ▼   public static void main(String[] args) {
5
6       //Isso é um comentário de linha simples.
7       //O compilador despreza tudo que comece com //
8       System.out.println("Olá mundo!");
9
10      /*
11      Esse é um comentário de linha múltipla.
12
13      Tudo que você colocar aqui será ignorado.
14      */
15
16   }
17
18 }
```

Vamos criá-lo passo a passo...

# Comentários

```
1
2 ▼ public class HelloWorld {
3
4 ▼     public static void main(String[] args) {
5
6         //Isso é um comentário de linha simples.
7         //O compilador despreza tudo que comece com //
8         System.out.println("Olá mundo!");
9
10        /*
11        Esse é um comentário de linha múltipla.
12
13        Tudo que você colocar aqui será ignorado.
14        */
15
16    }
17
18 }
```

# Declarando uma classe

```
1
2 public class HelloWorld {
3
4     public static void main(String[] args) {
5
6         //Isso é um comentário de linha simples.
7         //O compilador despreza tudo que comece com //
8         System.out.pr
9
10        /*
11        Esse é um come
12
13        Tudo que você
14        */
15
16    }
17
18 }
```

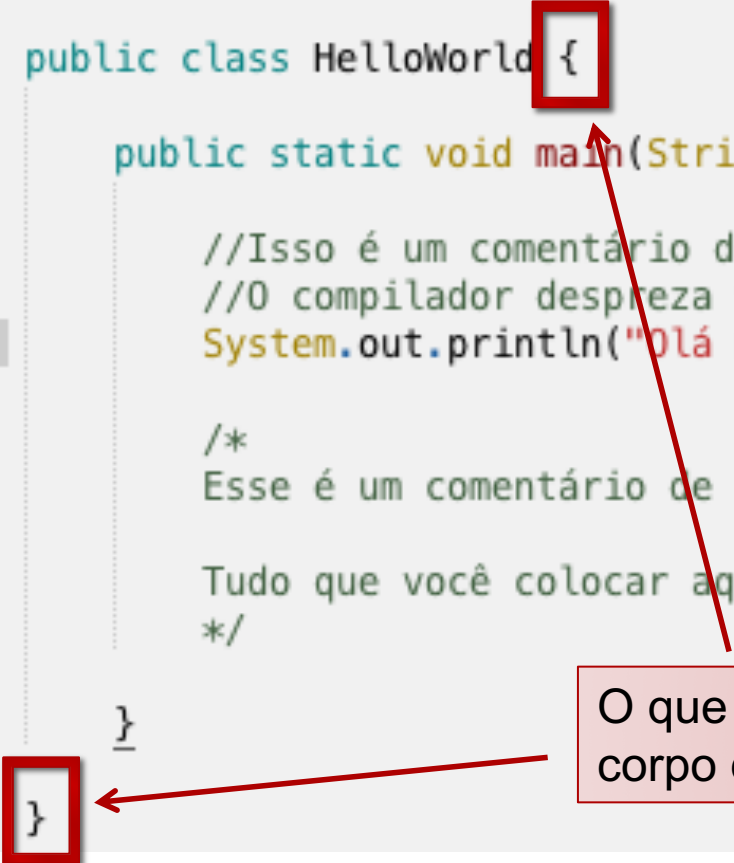
A declaração de uma classe se inicia com a utilização da palavra chave **class** e segue logo depois com o nome da classe. Nomes de classe 'devem' iniciar com letra maiúscula.

O nome do arquivo **.java** deter o mesmo nome da classe.



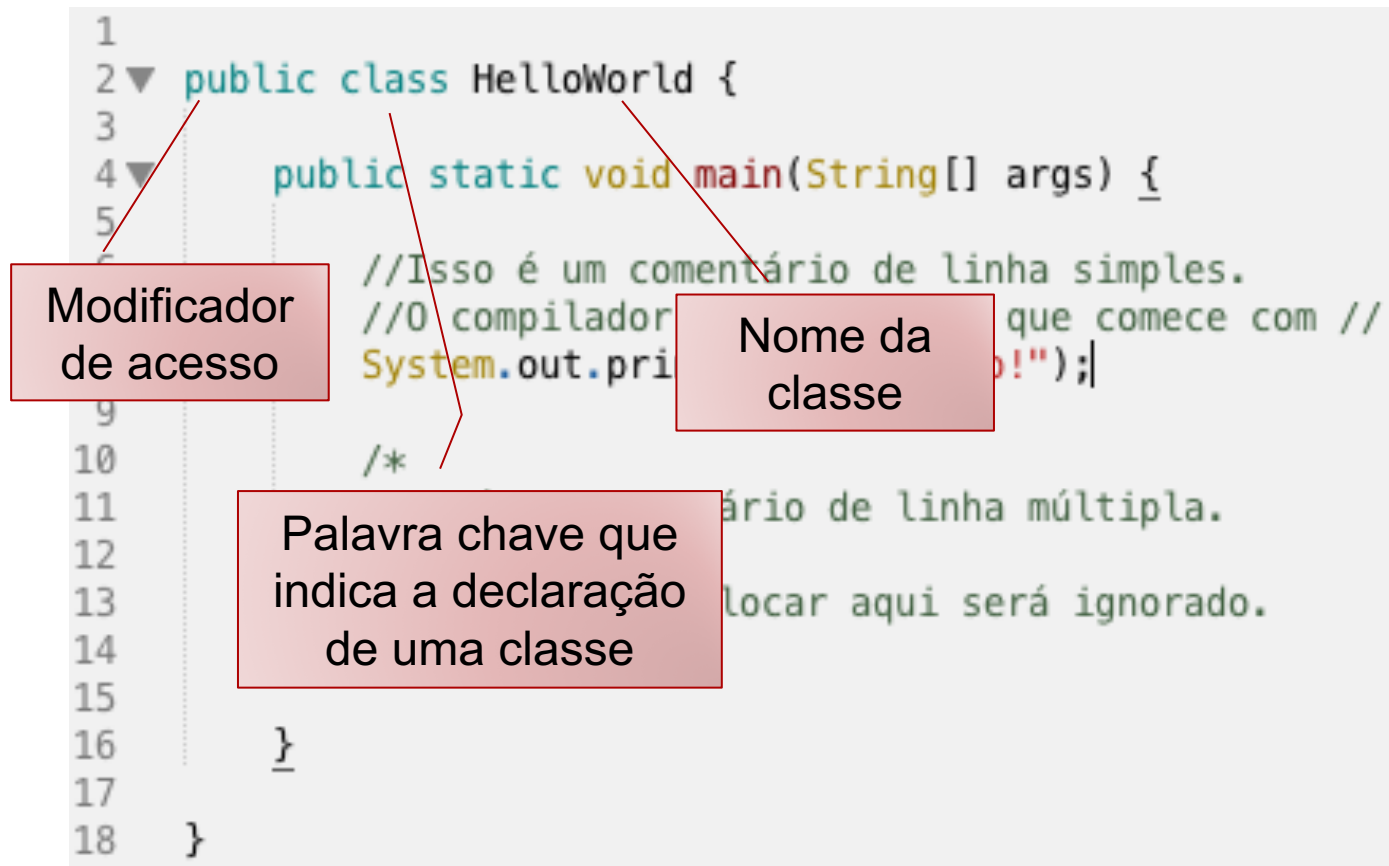
# Declarando uma classe

```
1
2 ▼ public class HelloWorld {
3
4 ▼   public static void main(String[] args) {
5
6       //Isso é um comentário de linha simples.
7       //O compilador despreza tudo que comece com //
8       System.out.println("Olá mundo!");
9
10      /*
11      Esse é um comentário de linha múltipla.
12
13      Tudo que você colocar aqui será ignorado.
14      */
15
16   }
17 }
18
```



O que aparecer entre as chaves compõe o corpo da classe.

# Anatomia de uma declaração de classe



# Identificadores

- Por convenção, nomes de classe devem iniciar com letra maiúscula, bem como a primeira letra de cada 'palavra' que a compõe, caso seja um nome composto.
  - Exemplos de nomes de classes não permitidos:
    - 1carro
    - \*carro
    - &Carro
  - Exemplo de nomes de classes que seguem a convenção:
    - Carro
    - CarroFamilia
- Nomes de classe podem começar com letra minúscula e com o caracter *underscore* (`_`) mas não se aconselha fazer isso pois dificulta o entendimento

# Declarando um método

```
1
2 ▼ public class HelloWorld {
3
4 ▼ public static void main(String[] args) {
5
6     //Isso é um comentário de linha simples.
7     //O compilador despreza tudo que comece com //
8     System.out.println("Hello World!");
9
10    /*
11     * Esse é um comentário de bloco
12     * Tudo que você escrever aqui será ignorado
13     */
14
15 }
16 }
17
18 }
```

Por convenção, nomes de métodos iniciam com letra minúscula. No mais, seguem as mesmas regras para definição de nomes de classes.

Todo programa Java DEVE possuir um método *main*.

Os parênteses após o identificador do método indicam que esta declaração trata-se de um método.

# Anatomia da declaração de um método

```
1
2 ▼ public class HelloWorld {
3
4 ▼ public static void main(String[] args) {
5
6     //Isso é um comentário de linha simples.
7     //O compilador despreza tudo que comece com //
8     System.out.println("Olá mundo!");
9
10    *
11    Esse é um comentário de linha múltipla.
12
13    * colocar aqui será ignorado.
14
15
16 }
17
18 }
```

Nome do método

Modificador de acesso

Tipo de retorno

Parâmetros

Indica se o método é de instância ou de classe

# Declarando um método

```
1
2 ▼ public class HelloWorld {
3
4 ▼   public static void main(String[] args) {
5
6       //Isso é um comentário de linha simples.
7       //O compilador despreza tudo que comece com //
8       System.out.println("Olá mundo!");
9
10      /*
11      Esse é um comentário de linha múltipla.
12
13      Tudo que você colo
14      */
15
16   }
17
18 }
```

Semelhante ao que acontece com as classes, as chaves delimitam o corpo do método.

# Alguns exemplos práticos

- Mostrar a diferença entre o `print` e `println`
- Mostrar o uso de caracteres de escape
- Mostrar o uso do `printf`
  - `%d`
  - `%x`
  - `%s`
  - `%f`
- OBS: O apêndice G do livro Java Como Programar mostra mais exemplos e outros tipos de formatação para o *printf*