Modularidade e Métodos estáticos

UA.DETI.POO

Modularidade

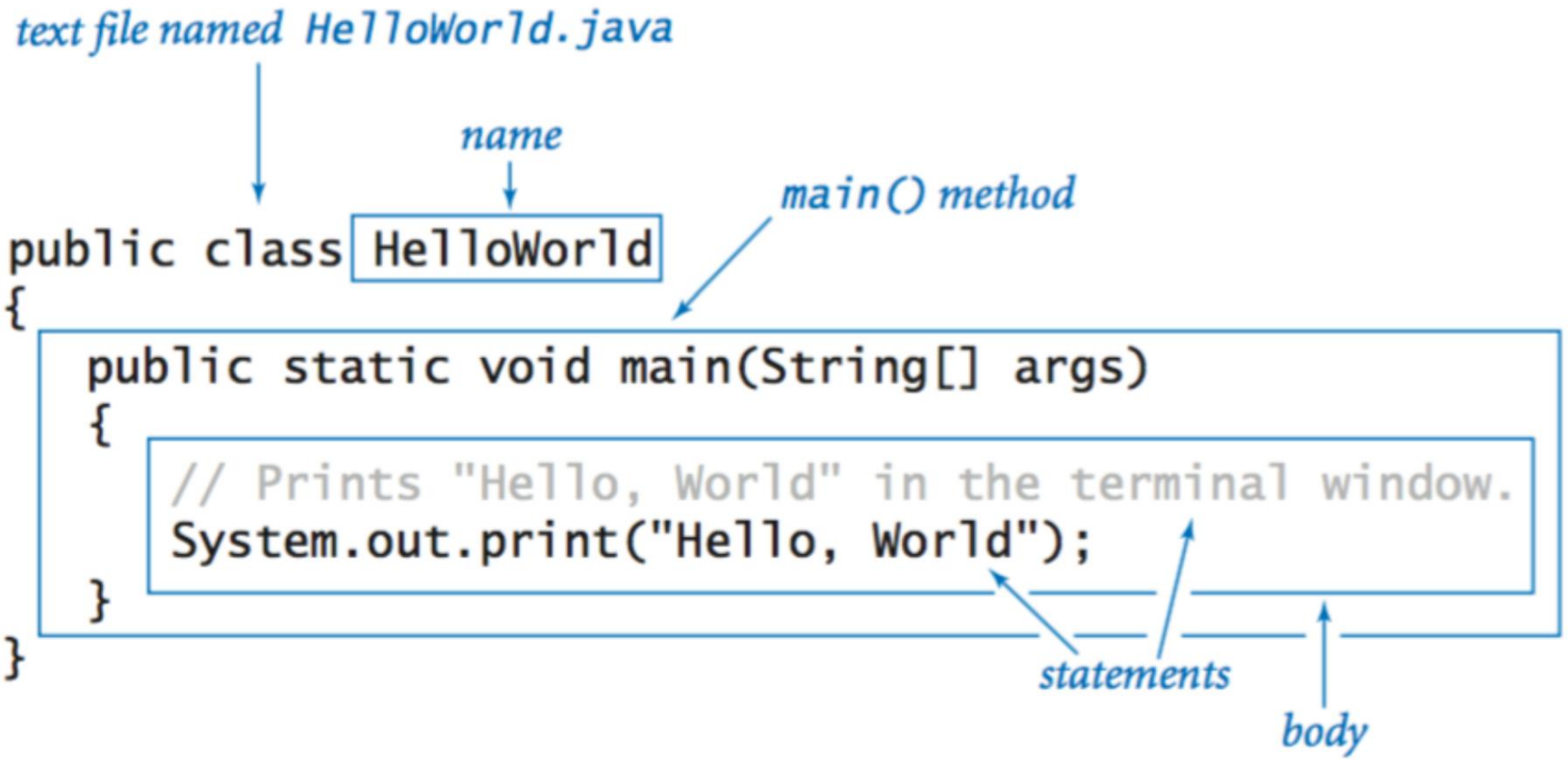
Conceito básico de classe

- Definição duma classe (ficheiro Exemplo.java):

```
public class Exemplo {
    // dados
    // métodos
}
```

- O ficheiro Exemplo.java deve conter uma classe pública denominada Exemplo.
  - Devemos usar uma nomenclatura do tipo Person, SomeClass, SomeLongNameForClass, …
  - Java é uma linguagem *case-sensitive* (i.e. Exemplo != exemplo)
- Esta classe deve ser declarada como public

Classe principal e método main



Funções/métodos estáticos

- Uma função
  - Realiza uma tarefa.
  - Tem zero ou mais argumentos de entrada.
  - Retorna zero ou um valor de saída.
- Aplicações
  - Os cientistas usam funções matemáticas para calcular fórmulas.
  - Os programadores usam funções para construir programas modulares.
  - Vamos usá-las para ambos os objetivos.
- Exemplos

```
Math.random(), Math.abs(), Integer.parseInt()
System.out.println(), main()
```

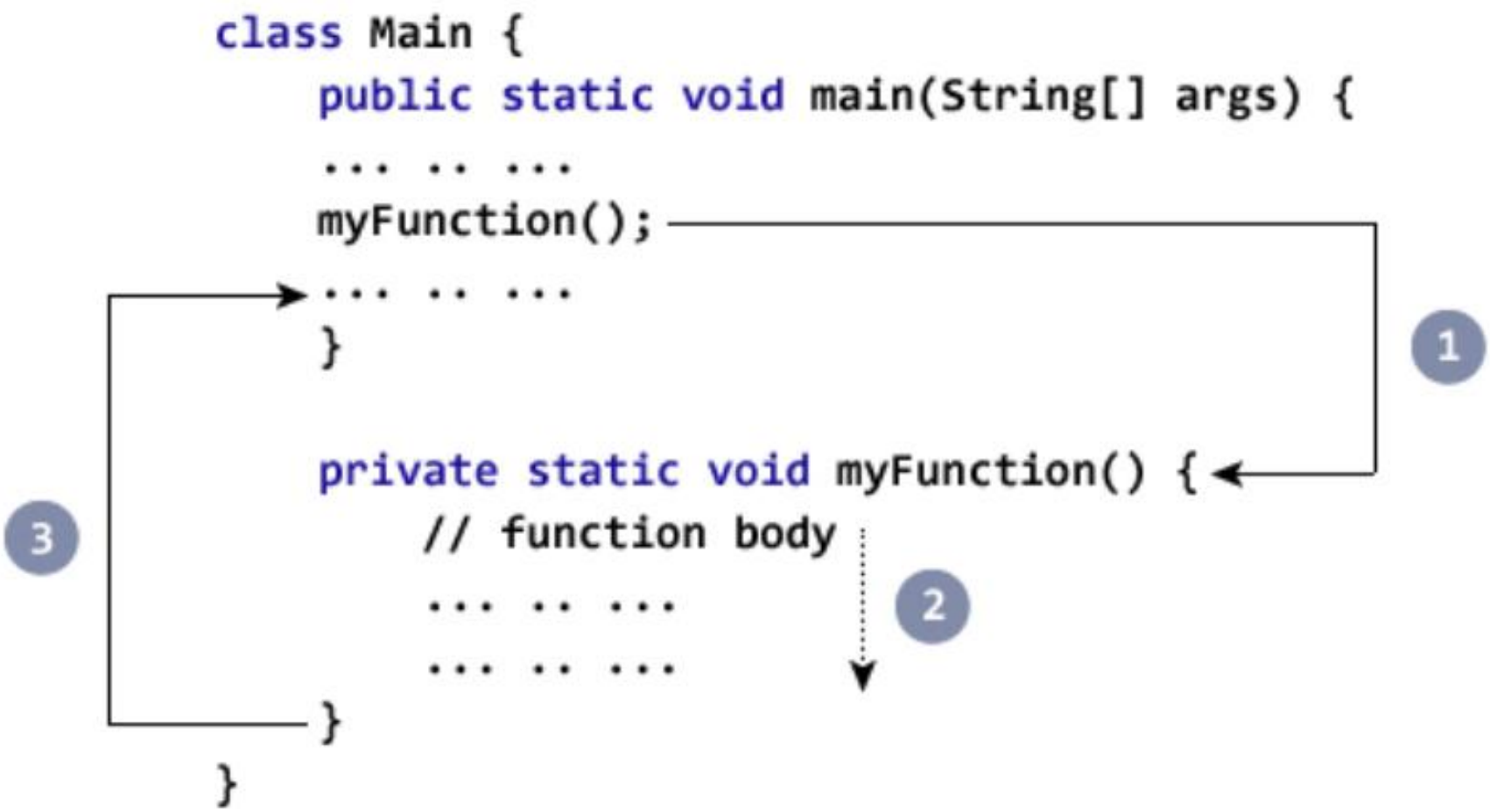
Métodos estáticos

- Para implementar uma função (método estático), precisamos de
  - Criar um nome
  - Declarar o tipo e o nome do(s) argumento(s)
  - Especificar o tipo para o valor de retorno
  - Implementar o corpo do método
  - Terminar com a instrução de retorno

```
public static void myFunction() {
    System.out.println("My Function called");
}

public static double doisXQuadrado(double x) {
    return 2*x*x;
}
```

Execução



https://www.programiz.com/java-programming/methods

Exemplos

```
public class Testes {

    public static void main(String[] args) {
        System.out.println("About to encounter a method.");
        // method call
        myMethod();
        System.out.println("Method was executed successfully!");
    }

    // method definition
    private static void myMethod() {
        System.out.println("Printing from inside myMethod()!");
    }

}
```

Exemplos

```
public class Testes {

    public static int getIntegerSum(int i, int j) {
        return i + j;
    }

    public static int multiplyInteger(int x, int y) {
        return x * y;
    }

    public static void main(String[] args) {
        System.out.println("10 + 20 = " + getIntegerSum(10, 20));
        System.out.println("20 x 40 = " + multiplyInteger(20, 40));
    }
}
```

```
10 + 20 = 30
20 x 40 = 800
```

Exemplos

```
public class Testes {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            // method call
            int result = getSquare(i);
            System.out.println("Square of " + i + " is : " + result);
        }
    }

    private static int getSquare(int x) {
        return x * x;
    }
}
```

```
Square of 1 is : 1
Square of 2 is : 4
Square of 3 is : 9
Square of 4 is : 16
Square of 5 is : 25
```

Packages

Java tem pacotes

- Java fornece classes existentes em pacotes
- Podem definir novos packages

https://docs.oracle.com/en/java/javase/11/docs/api/

All Modules | Java SE | JDK | Other Modules

Module	Description
java.base	Defines the foundational APIs of the Java SE Plat
java.compiler	Defines the Language Model, Annotation Process
java.datatransfer	Defines the API for transferring data between an
java.desktop	Defines the AWT and Swing user interface toolkit
java.instrument	Defines services that allow agents to instrument
java.logging	Defines the Java Logging API.
java.management	Defines the Java Management Extensions (JMX) A
java.management.rmi	Defines the RMI connector for the Java Managem
java.naming	Defines the Java Naming and Directory Interface
java.net.http	Defines the HTTP Client and WebSocket APIs.
java.prefs	Defines the Preferences API.
java.rmi	Defines the Remote Method Invocation (RMI) API
java.scripting	Defines the Scripting API.
java.se	Defines the API of the Java SE Platform.
java.security.jgss	Defines the Java binding of the IETF Generic Sec
java.security.sasl	Defines Java support for the IETF Simple Authent
java.smartcardio	Defines the Java Smart Card I/O API.
java.sql	Defines the JDBC API.
java.sql.rowset	Defines the JDBC RowSet API.

Espaço de Nomes - Package

- Em Java a gestão do espaço de nomes (*namespace*) é efetuado através do conceito de package.

- Porque gestão de espaço de nomes?

- → Evita conflitos de nomes de classes
  - Não temos geralmente problemas em distinguir os nomes das classes que construímos.
  - Mas como garantimos que a nossa classe Book não colide com outra que eventualmente possa já existir?

Package e import

- Utilização
  - As classes são referenciadas através dos seus nomes absolutos ou utilizando a primitiva import.

```
import java.util.ArrayList
import java.util.*
```

  - A cláusula import deve aparecer sempre nas primeiras linhas de um programa.
- Quando escrevemos,

```
import java.util.*;
```

  - estamos a indicar um caminho para um pacote de classes permitindo usá-las através de nomes simples:

```
ArrayList<String> al = new ArrayList<>();
```

- De outra forma teríamos de escrever:

```
java.util.ArrayList<String> al = new java.util.ArrayList<>();
```

Criar um package

- Na primeira linha de código:

```
package poo;
```

  - garante que a classe pública dessa unidade de compilação fará parte do package poo.
- O espaço de nomes é baseado numa estrutura de sub-directórios
  - Este package vai corresponder a uma entrada de directório: **$CLASSPATH/poo**
  - Boa prática usar DNS invertido: **pt.ua.deti.poo**
- A sua utilização será na forma:

```
poo.Book sr = new poo.Book();
```

  - ou

```
import poo.*
Book sr = new Book();
```

Espaço de Nomes - Package

```
package aula01;
```

- Em Java a gestão do espaço de nomes (*namespace*) é efetuado através do conceito de package.
  - Evita conflitos de nomes de classes

- O espaço de nomes é baseado numa estrutura de sub-diretórios
  - O package 'aula01' do exemplo anterior vai corresponder a uma entrada de diretório
  - O "*Fully Qualified Name*" da classe será aula01.MyFirstClass
  - Este FQN corresponde ao caminho aula01/MyFirstClass.class

- Voltaremos a isto mais tarde

O meu primeiro "package"

```
└── poo
    └── test
        └── mypack
            └── MyPackageClass.java
```

```
package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

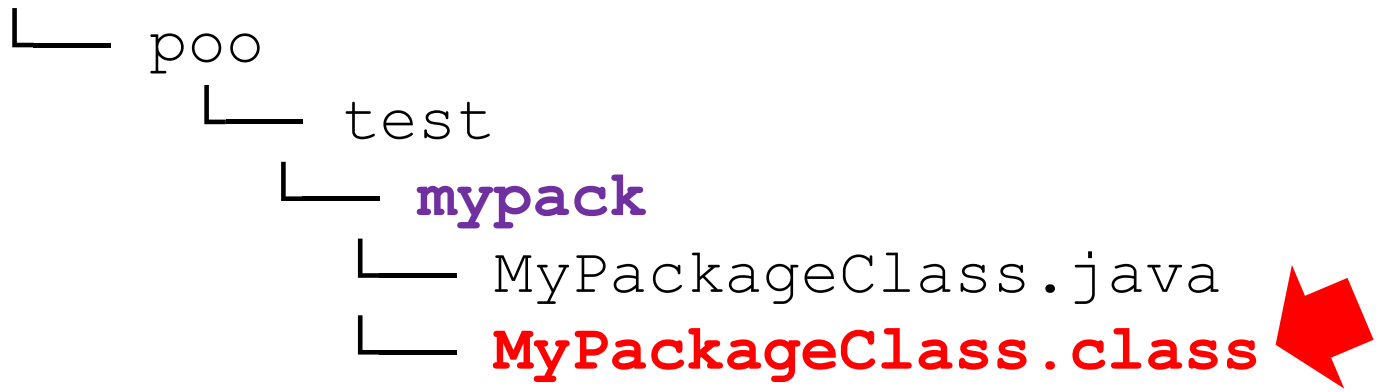
Compilar o código

```
└── poo
    └── test
        └── mypack
            └── MyPackageClass.java
```

```
package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

```
C:\poo\test>javac .\mypack\MyPackageClass.java
C:\poo\test>java mypack.MyPackageClass
```

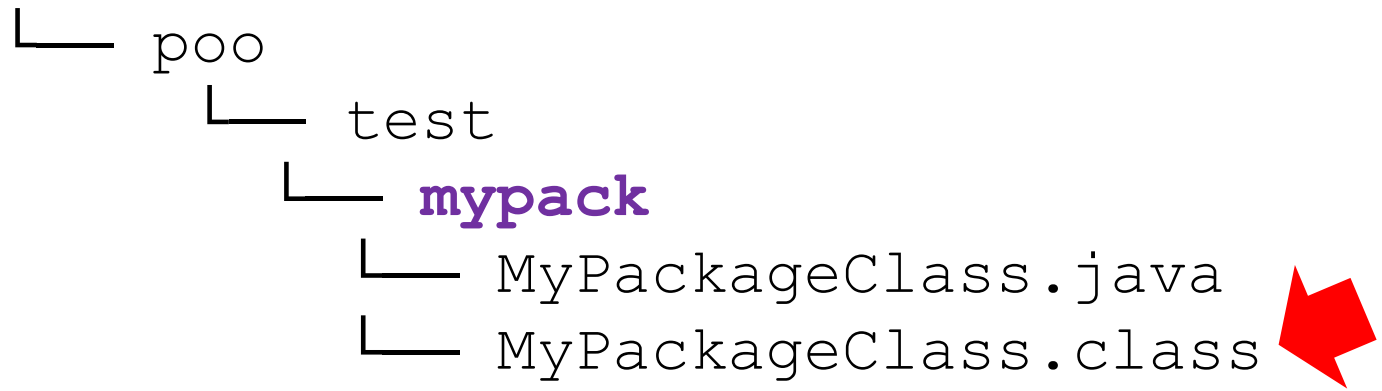
Compilar o código gera .class no pacote (diretorio)




```
package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

```
C:\poo\test>javac .\mypack\MyPackageClass.java
```

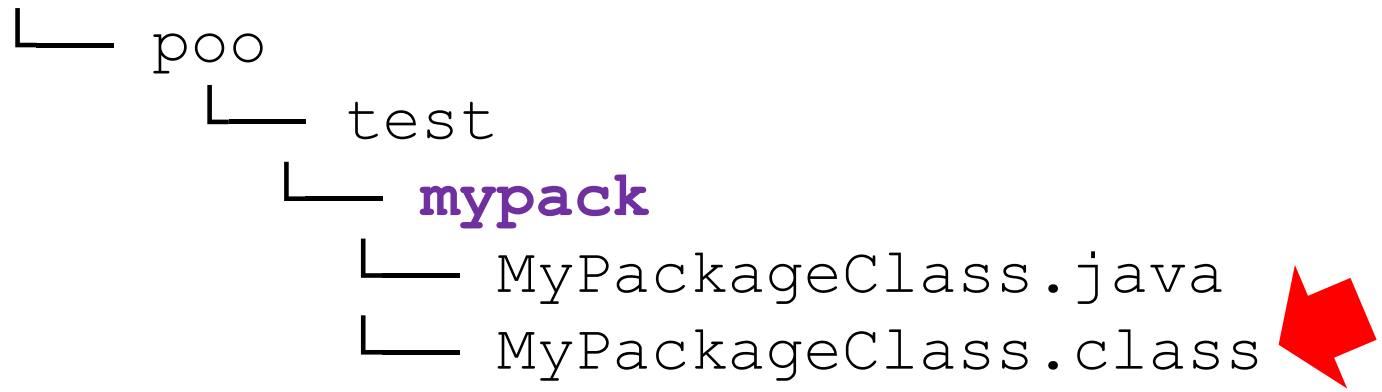
Executar o código implica colocar o package




```
package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

```
C:\poo\test>javac .\mypack\MyPackageClass.java
C:\poo\test>java mypack.MyPackageClass
```

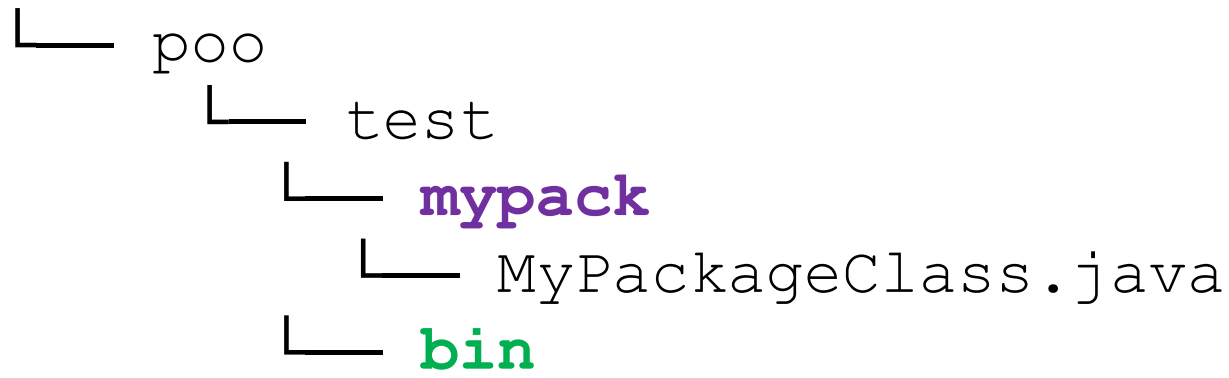
Executar o código implica colocar o package




```
package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

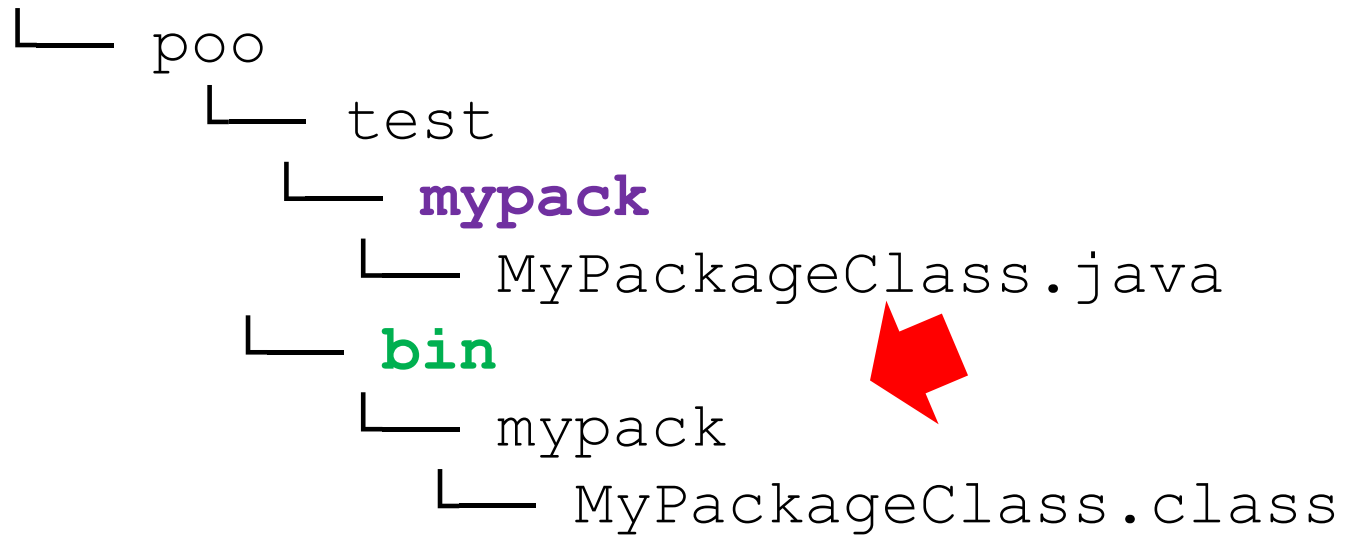
```
C:\poo\test>javac .\mypack\MyPackageClass.java
C:\poo\test>java mypack.MyPackageClass
This is my package!
```

Se quiser colocar os executáveis noutro sitio



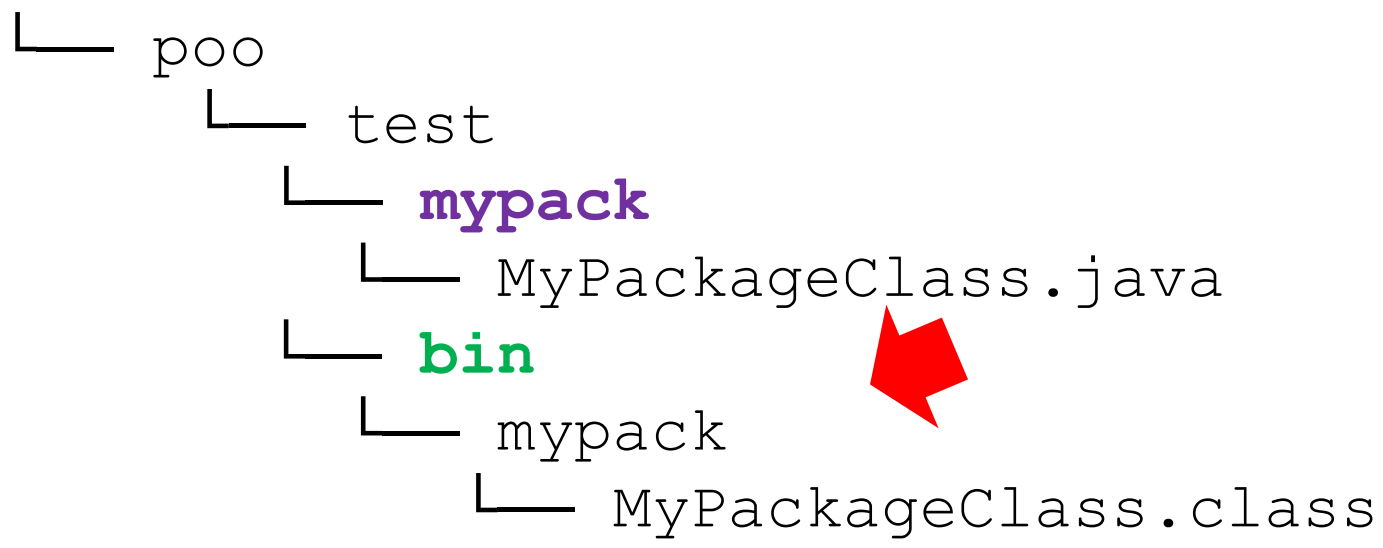

```
C:\poo\test>javac -d .\bin .\mypack\MyPackageClass.java
```

Se quiser colocar os executáveis noutro sitio




```
C:\poo\test>javac -d .\bin .\mypack\MyPackageClass.java
```

Se quiser colocar os executáveis noutro sitio




```
C:\poo\test>javac -d .\bin .\mypack\MyPackageClass.java
C:\poo\test>java -cp .\bin mypack.MyPackageClass
This is my package!
```

Integrated Development Environment (IDE)

- Automatizam este processo
- Identificam o código
- Guardam configurações
  - Onde está o código (*.java)
  - Onde colocam os executáveis (*.class)
  - Onde podem encontrar executáveis (*.jar, *.class)
- Compilam (javac)
  - Colocam no local configurado
- Executam (java)
  - Executam código a partir do local

https://code.visualstudio.com/docs/java/java-debugging

https://www.jrebel.com/blog/best-java-ide

BLOG January 2, 2026

Most Popular Java IDEs in 2026

JAVA TOOLS

Many team leaders struggle to get their Java developers to use the same Java tech stack — or they don't even try. That's because many developers are stalwart fans of what they know, despite what their coworkers use or company initiatives to standardize development toolchains.

Whether you're a novice Java developer learning an IDE for the first time or a seasoned programmer looking to see if the grass is greener on the other side of the fence, you've come to the right place. We've ranked the three most popular Java IDEs of 2026, and outlined the benefits, drawbacks and use cases for each. In addition, we talk about up-and-coming multi-language IDEs and how they may impact the Java development ecosystem.

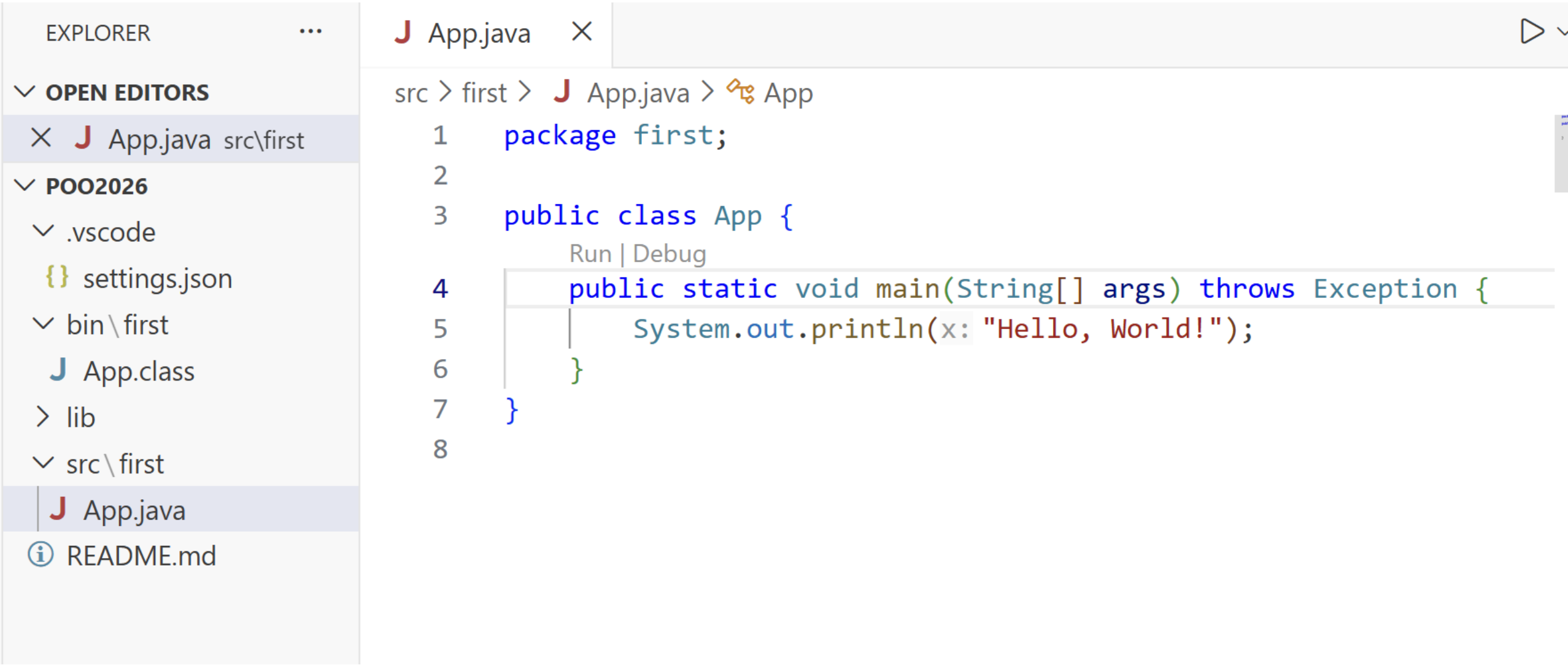
Table of Contents

https://www.jrebel.com/blog/best-java-ide


App.java - poo2026 - Visual Studio Code
EXPLORER
OPEN EDITORS
App.java src
POO2026
.vscode
settings.json
bin
lib
src
App.java
README.md
App.java
src > App.java
1 public class App {
Run | Debug
2 public static void main(String[] args) throws Exception {
3 System.out.println(x: "Hello, World!");
4 }
5 }
6
Run Java
Debug Java


App.java - poo2026 - Visual Studio Code
EXPLORER
OPEN EDITORS
App.java src
POO2...
.vscode
settings.json
bin
App.class
lib
src
App.java
README.md
App.java
src > App.java
1 public class App {
Run | Debug
2 public static void main(String[] args) throws Exception {
3 System.out.println(x: "Hello, World!");
4 }
5 }
6
Run Java
Debug Java


App.java - poo2026 - Visual Studio Code
EXPLORER
OPEN EDITORS
App.java src
POO2...
.vscode
settings.json
bin
App.class
lib
src
App.java
README.md
App.java
src > App.java
1 public class App {
Run | Debug
2 public static void main(String[] args) throws Exception {
3 System.out.println(x: "Hello, World!");
4 }
5 }
6
Run Java
Debug Java


EXPLORER
OPEN EDITORS
App.java src\first
POO2026
.vscode
settings.json
bin\first
App.class
lib
src\first
App.java
README.md
App.java
src > first > App.java > App
1 package first;
2
3 public class App {
Run | Debug
4 public static void main(String[] args) throws Exception {
5 System.out.println(x: "Hello, World!");
6 }
7 }
8

Sumário

- Modularidade
  - Classes
  - Métodos Estáticos
  - Pacotes