

# 1.1 Java 開発環境構築

- Java
- Java のインストール
- 初めての Java プログラム
- 統合開発環境



- 1 Java
- 2 Java のインストール
- 3 初めての Java プログラム
- 4 統合開発環境

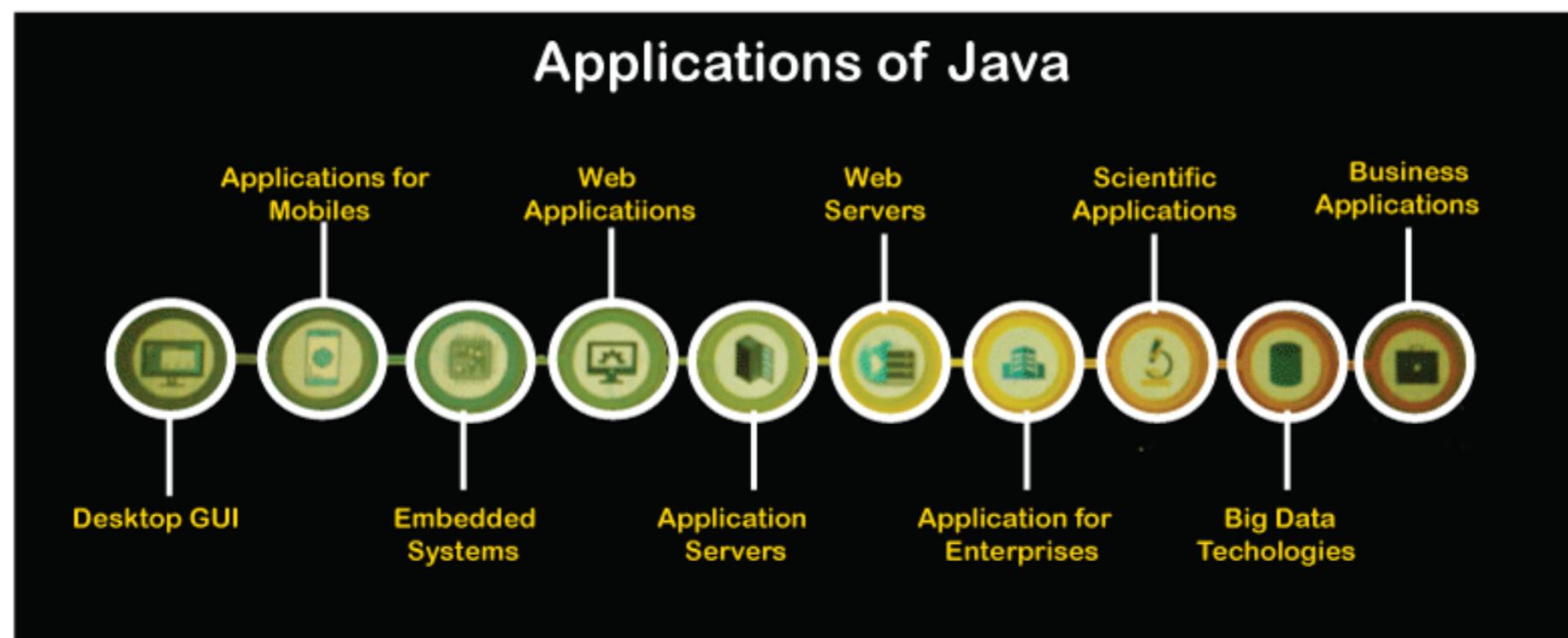
# Java とは

- Java とは、クロスプラットフォーム[Cross-platform]、オブジェクト指向[Object-oriented]、ジェネリックタイプなどの特徴を持ち、広く使われているプログラミング言語の一つ。
- C++ という言語が複雑なプログラムを書くと、色々な問題が生じます。Java は、このような問題を解決するために発明されました。そのため、Java は C や C++ と構文的に似ています。



# Java の応用

- Java は、サーバーやモバイルアプリケーション、ビジネスシステム、ウェブアプリ、デスクトップアプリ開発など、さまざまな用途に使用することができます。



# Java の主な特性

- オブジェクト指向プログラミング [Object-oriented Programming, OOP]。
- 様々な API (Application Programming Interface)：
  - グラフィカル・ユーザー・インターフェイス [GUI]。
  - マルチスレッド [Multithreading]。
  - ネットコミュニケーション。
- メモリ管理：ガベージ・コレクション [Garbage Collection, GC]。
- 静的型付け [Static Typing]：コンパイル時に変数の型はチェックされる。
- プラットフォームに依存しない、「Write Once Run Anywhere」、略して WORA という性質。



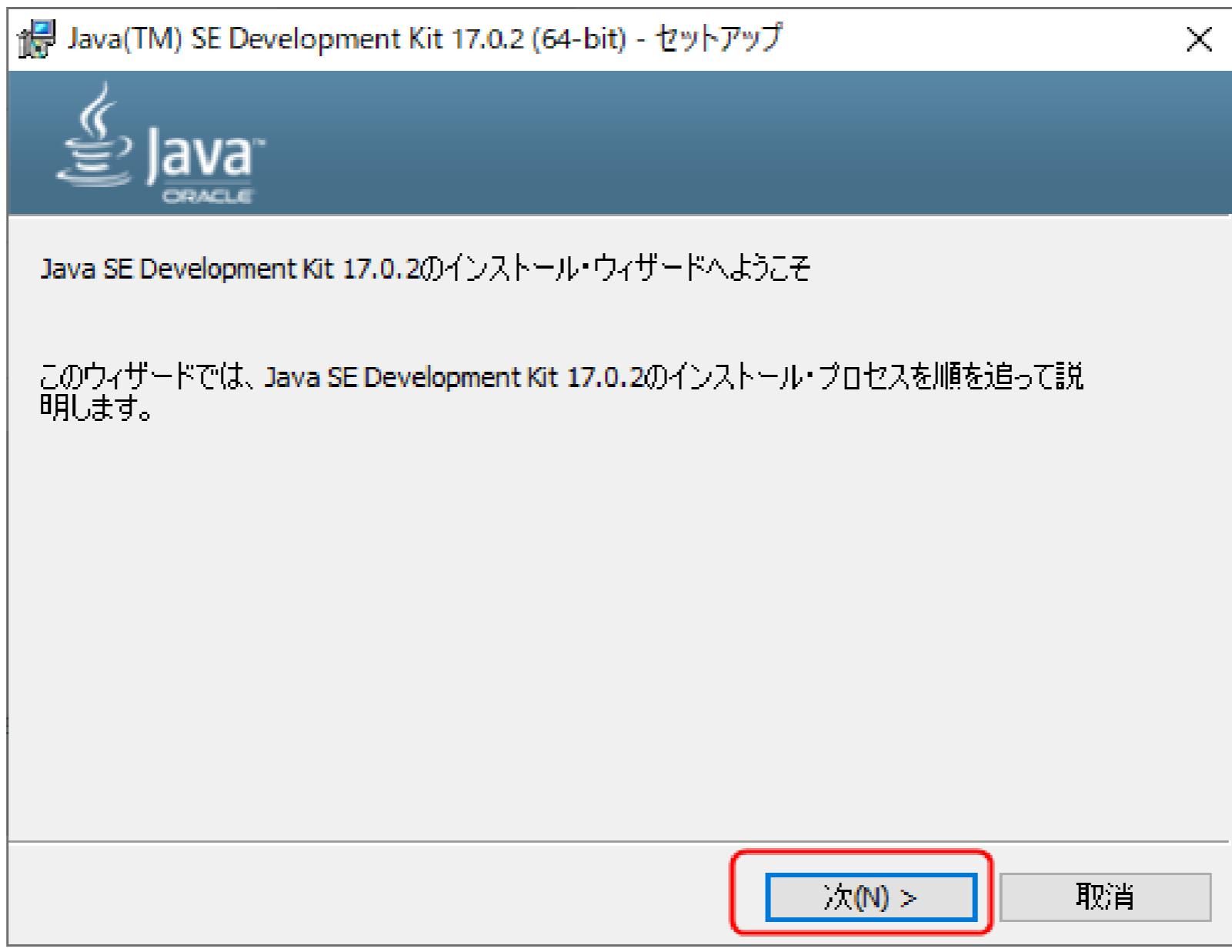
- 1 Java
- 2 Java のインストール
- 3 初めての Java プログラム
- 4 統合開発環境

# Step 1 開発ツールのダウンロード

- **JDK** (Java Development Kit) とは、Java の開発ツールです。
- オラクルの公式 JDK はこちちらでダウンロードできます：  
 <https://www.oracle.com/java/technologies/downloads/>
- 自分のコンピューターに合ったパッケージを探し、ダウンロードしてください。本講義では、JDK17 を使用します。
- これからステップは、Windows 用 (➡ p7) と Mac 用 (➡ p14) に分かれます。
- すでにインストールされている方は、スキップしても構いません。

## Step 2 JDK のインストール

- ダウンロードした .exe ファイルをダブルクリックして実行。



◀ 前へ

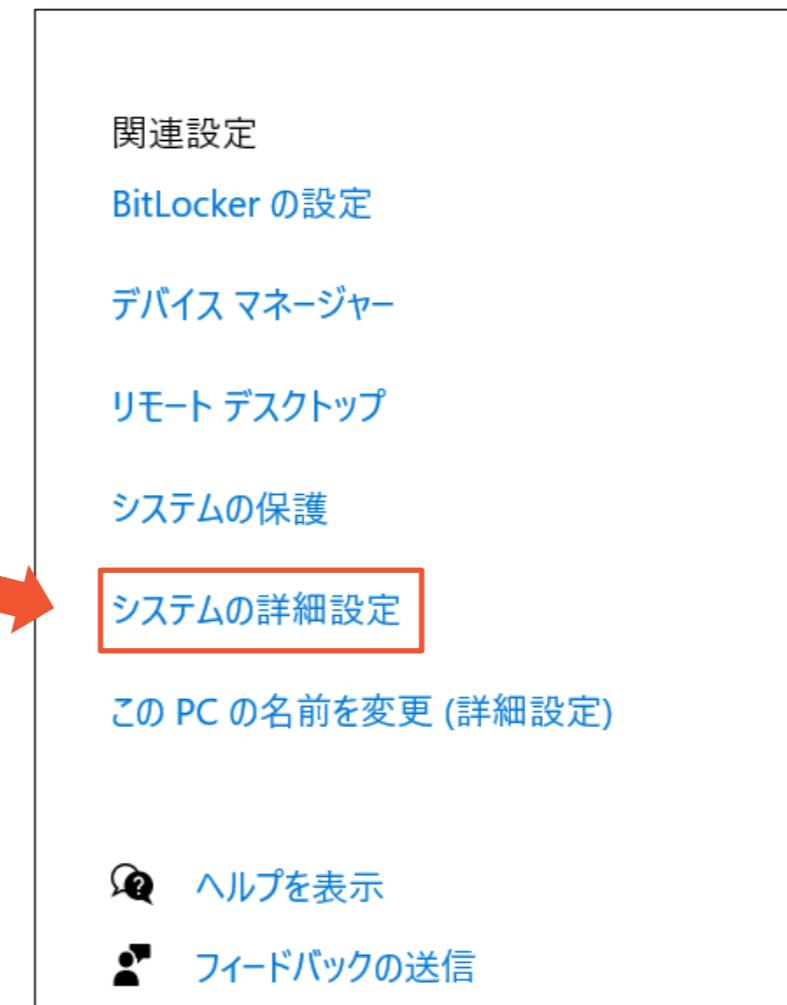
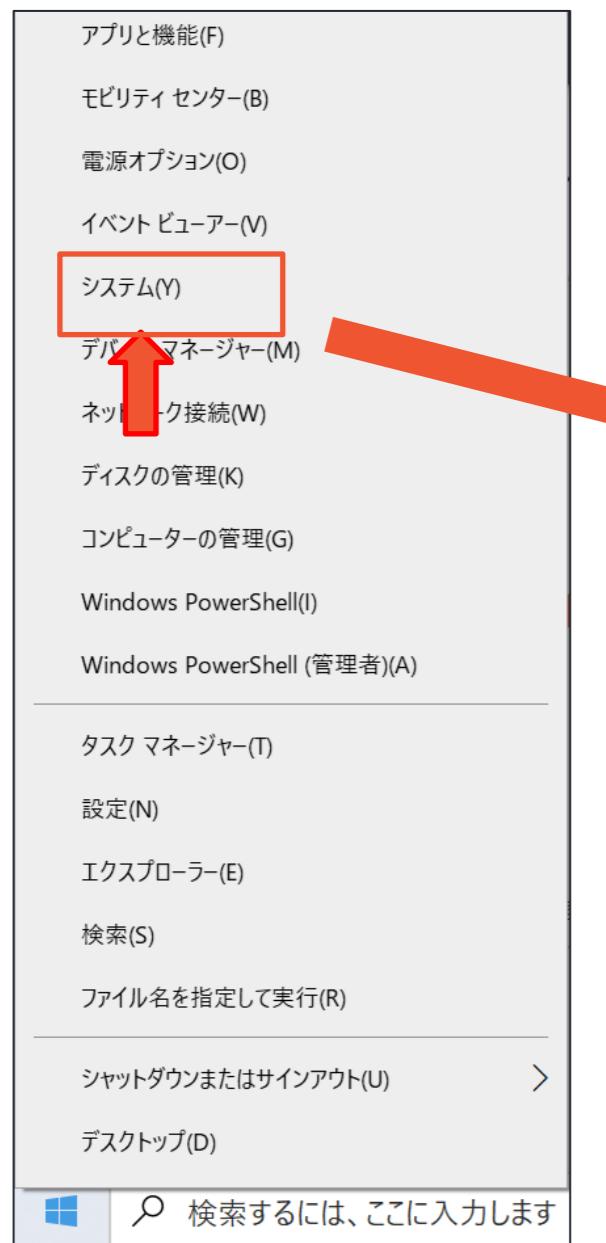


このパスを覚えてください



# Step 3 環境変数の設定

- システム → システムの詳細設定 → 環境変数。



次へ ➞

◀ 前へ



次へ ▶

 前へ

- 「Path」をダブルクリック:

システム環境変数(S)

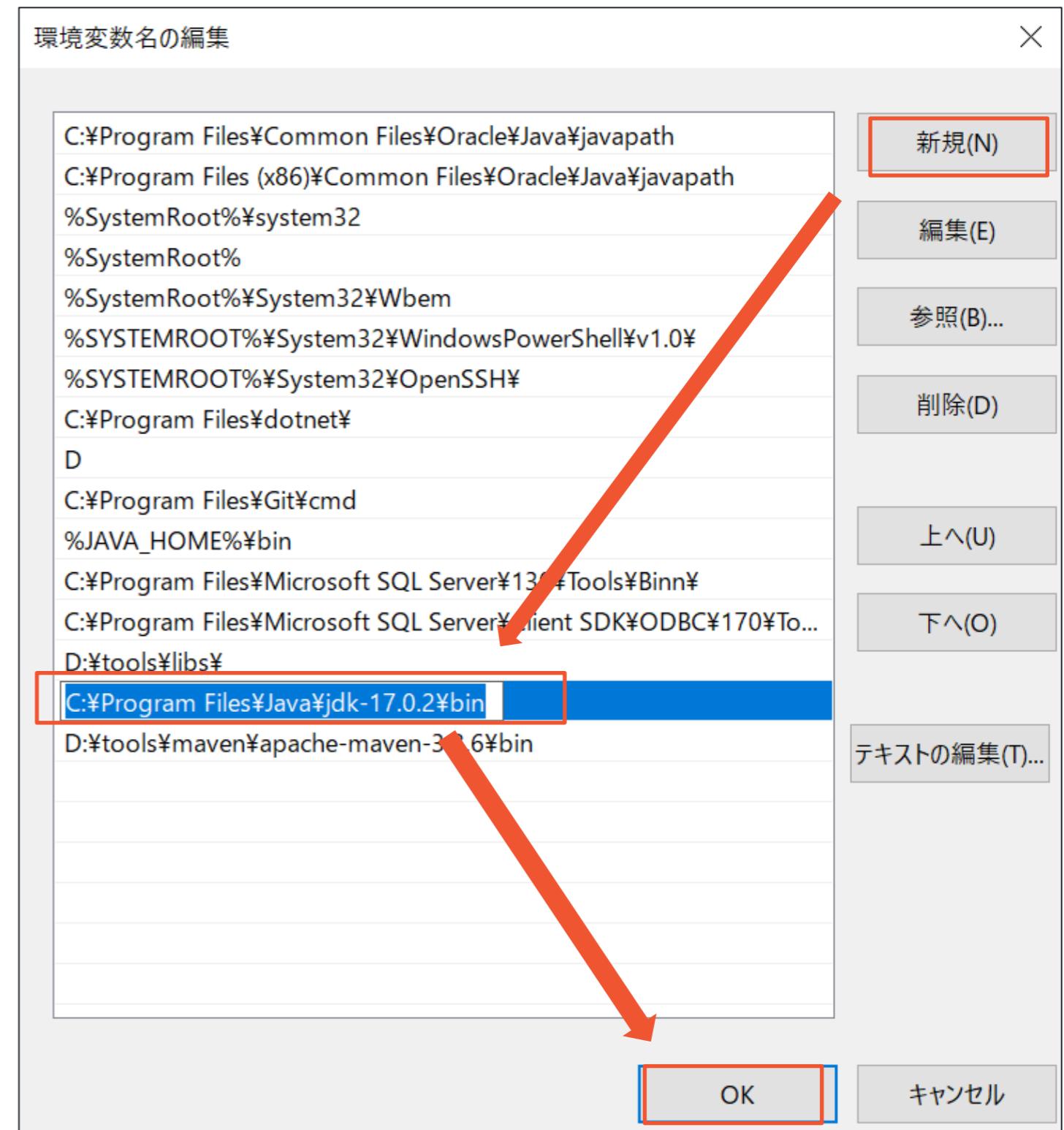
変数	値
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	C:\Users\45649\jdk\openjdk-16.0.1
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Progr...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

 新規(W)... 編集(I)... 削除(L)

次へ 

 前へ

- 新規:  
[JDK のパス]\bin\



## Step 4 インストールが成功したことを確認

- cmd を起動し、「`java -version`」（半角スペースに注意）と入力し、Enter キーを押します。

```
C:\Users\Your Name>java -version
```

- 環境の設定に成功すると、以下の結果が表示されます。

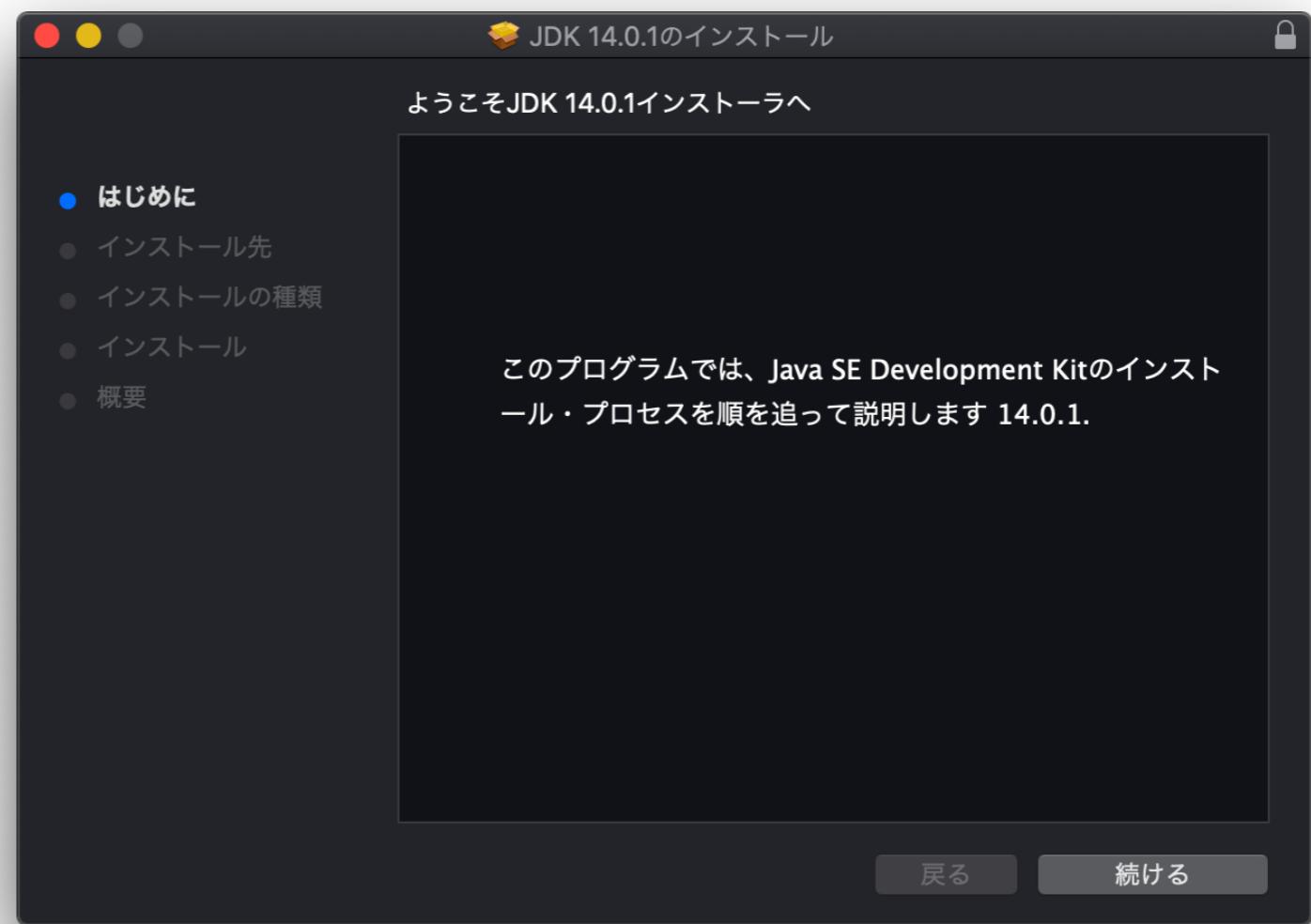
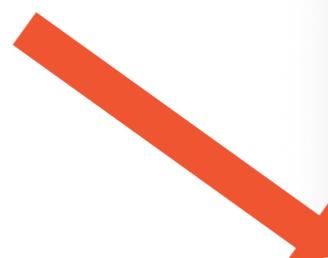
```
java version "11.0.1" 2018-10-16 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.1+13-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.1+13-LTS, mixed mode)
```

- こうして、無事に環境が整い、Java プログラムを書き始めることができます。



## Step 2 JDK のインストール

- ダウンロードした .dmg ファイルをダブルクリックし、その中の .pkg ファイルをダブルクリックしてオープン。



## Step 3 インストールが成功したことを確認

- Mac は自動的に環境変数を設定します。
- ターミナルを開き、「`java -version`」（半角スペースに注意）と入力し、Enter キーを押します。

```
(base) zmnnoMacBook-Pro:~ zmn$ java -version
```

- このような結果が表示されれば、設定は成功で、Java プログラムを書き始めることができます：

```
(base) zmnnoMacBook-Pro:~ zmn$ java -version
java version "14.0.1" 2020-04-14
Java(TM) SE Runtime Environment (build 14.0.1+7)
Java HotSpot(TM) 64-Bit Server VM (build 14.0.1+7, mixed mode, sharing)
```



# Q & A

*Question and answer*



- 1 Java
- 2 Java のインストール
- 3 初めての Java プログラム
- 4 統合開発環境

# 「Hello World」を書きましょう

- テキストエディタでファイルを作成し、次のコードを入力：

```
1 public class Hello {  
2     public static void main(String[] args) {  
3         System.out.println("Hello world!");  
4     }  
5 }
```

- インデント [Indent] (字下げ) に注意。
- 入力ソフトを英語・半角入力にしてください。
- 「Hello.java」という名前で保存します。（Mac のテキストエディットを使用する場合は、プレーンテキスト形式で保存する必要があります。ショートカットキー：Shift + Command + T。）

# Hello World の実行

- cmd (Windows) またはターミナル (macOS) を開いて、Hello.javaのあるディレクトリに cd します。
- 「javac Hello.java」と入力し、Enterキーを押します：

```
1.1 Java Environment Setup$src>javac Hello.java  
1.1 Java Environment Setup$src>java Hello
```

- 正しく書き込まれていれば、画面には「Hello world!」が表示されるはずです。
- おめでとう！Java プログラムを作成しました！

# Hello World の解説: クラス

- Java コードは、すべてなにかの「**クラス**[Class]」に属します。この例では、「Hello」というクラスが定義されています：

```
public class Hello {
```

- Java コードファイルの名前は、それが含むクラスの名前と**同じでなければなりません**。保存する際は、ファイル名としてクラス名を使用し、末尾に「.java」を付けてください。
- クラスは、オブジェクト指向プログラミングの基本要素であり、後(➡§2.1.1)で詳しく説明します。

# Hello World の解説: メインメソッド

- 「**メインメソッド**[Main Method]」とは、すべての Java コードの出発点です。メインメソッドの中に書かれるコードは、Java 起動時に順番に実行されます。
- メインメソッドの書き方はどのプログラムでも同じ:

```
public static void main(String[] args) {  
    実行させたいコード  
}
```

- 「**メソッド**[Method]」については、後 ([➡§1.3.3](#)) で詳しく説明します。今は、メインメソッドのコードをコピーして、各プログラムに貼り付けるだけで大丈夫です。

# Hello World 説明: System.out.println()

- **System.out.println()** メソッドは 1 行の文章を画面にプリント（出力）します:

```
System.out.println("出力させたいテキスト");
```

- 文章は必ず二重引用符「" "」で囲むようにしてください。
- このメソッドを使って、見たい結果を出力したり、デバッグしたりすることができます。

# Hello World 説明: Java の基本文法

- Java 文は、上から下へ順次実行されます。
- 各文は、セミコロン「;」で終了する。改行は文の区切りには使えません！
- 各単語は、スペース、タブ、ラインフィード（改行）などの空白記号で区切ることができます。
- 中括弧「{ }」で囲まれた文は、ブロック [block] といい、例えば、同じクラスや、同じメソッド内の文をまとめることができます。また、特別な場合を除いて、「}」の後にセミコロンで区切る必要はありません。

# コメント

- コメント [Comment] とは、コードとして実行されないテキストです。コメントでコードの説明を加えたり、コードを一時に消したり（コメントアウトという）することができます。
- 「//」で一行コメントが書けます：

```
System.out.println("A"); // This will print "A".
```

- 「/\* \*/」で複数行コメントが書けます：

```
1 /* The next code will print "A" onto the screen.  
2 You can change "A" to any sentence you want. */  
3  
4 System.out.println("A");
```

# 練習

- 以下のコードを完成せよ:

```
1 public class Hello {  
2     public static void main(String[] args) {  
3         // Write your code here.  
4     }  
5 }
```

- 以下の文字を出力できるようにしてください:

hello  
world

Tips 

System.out.print() メソッドと  
System.out.println() の違い  
を試してみてください。



*Question and answer*

## Coffee ☕ Break

### プログラミング作法と可読性

同じ機能を持つコードでも、さまざまな書き方があります。最近のプログラミング言語では、スペース、インデント、改行などの空白記号は、プログラマーが自由に配置できるようなものが多い。そのため、開発者ごとにコードのスタイルや習慣、つまり**プログラミング作法**[Programming Style]が異なります。

一方、一貫した書き方は、コードの**可読性**を高め、共同開発時のコミュニケーションの効率化を図るための鍵になります。ぜひ、主流で権威のある書き方から学んでください。

本講義で使用している Java コードは、Google の開発スタイルを参考にしています：

 <https://google.github.io/styleguide/javaguide.html>



- 1 Java
- 2 Java のインストール
- 3 初めての Java プログラム
- 4 統合開発環境

# 統合開発環境

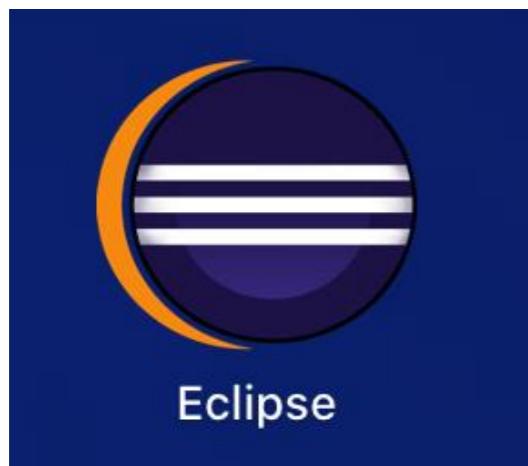
- **統合開発環境**[Integrated Development Environment, IDE]とは、プログラムの開発を支援するソフトウェアのことです。IDEは、テキストエディタで直接開発するよりも便利な機能をたくさん備えます：
  - ワンクリックでコードを実行、
  - 文法エラーの自動検出、
  - デバッグ機能、
  - コードの自動生成、
  - 多言語開発、
  - .....
- Java開発に使える代表的なIDEには、Eclipse、VS Code、IntelliJ IDEAなどが挙げられます。この講義ではEclipseを使って開発します。

# Eclipse のダウンロード

- こちらのリンクでダウンロード:  
 <https://www.eclipse.org/downloads/>
- お使いのシステムに適したものを選んでダウンロードしてください。
- 既にインストールした方は、再度ダウンロードする必要はありません。

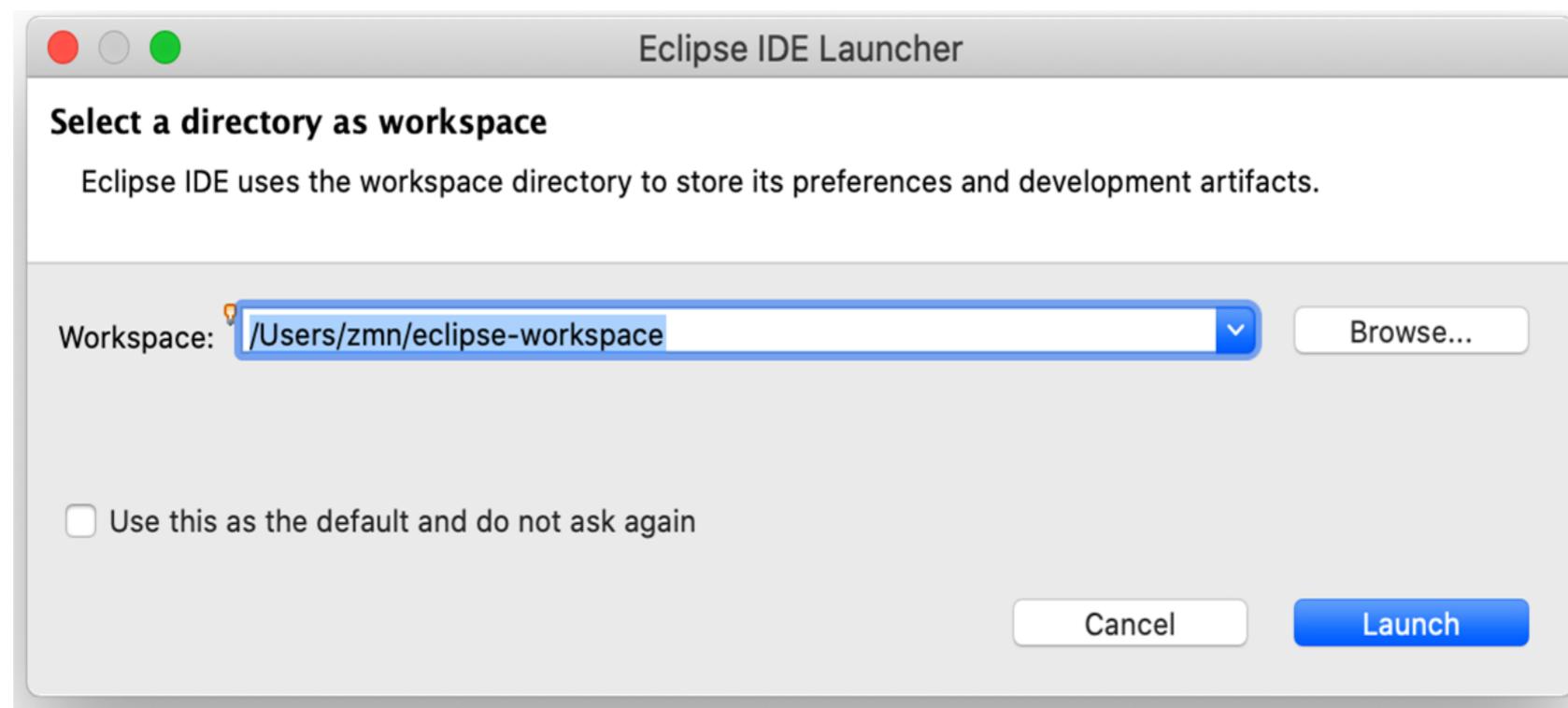
# Eclipse をインストール

- ダウンロードした .exe あるいは .dmg ファイルを開いてインストールしてください。
- インストールが終わったら、以下のアイコンが見れます：

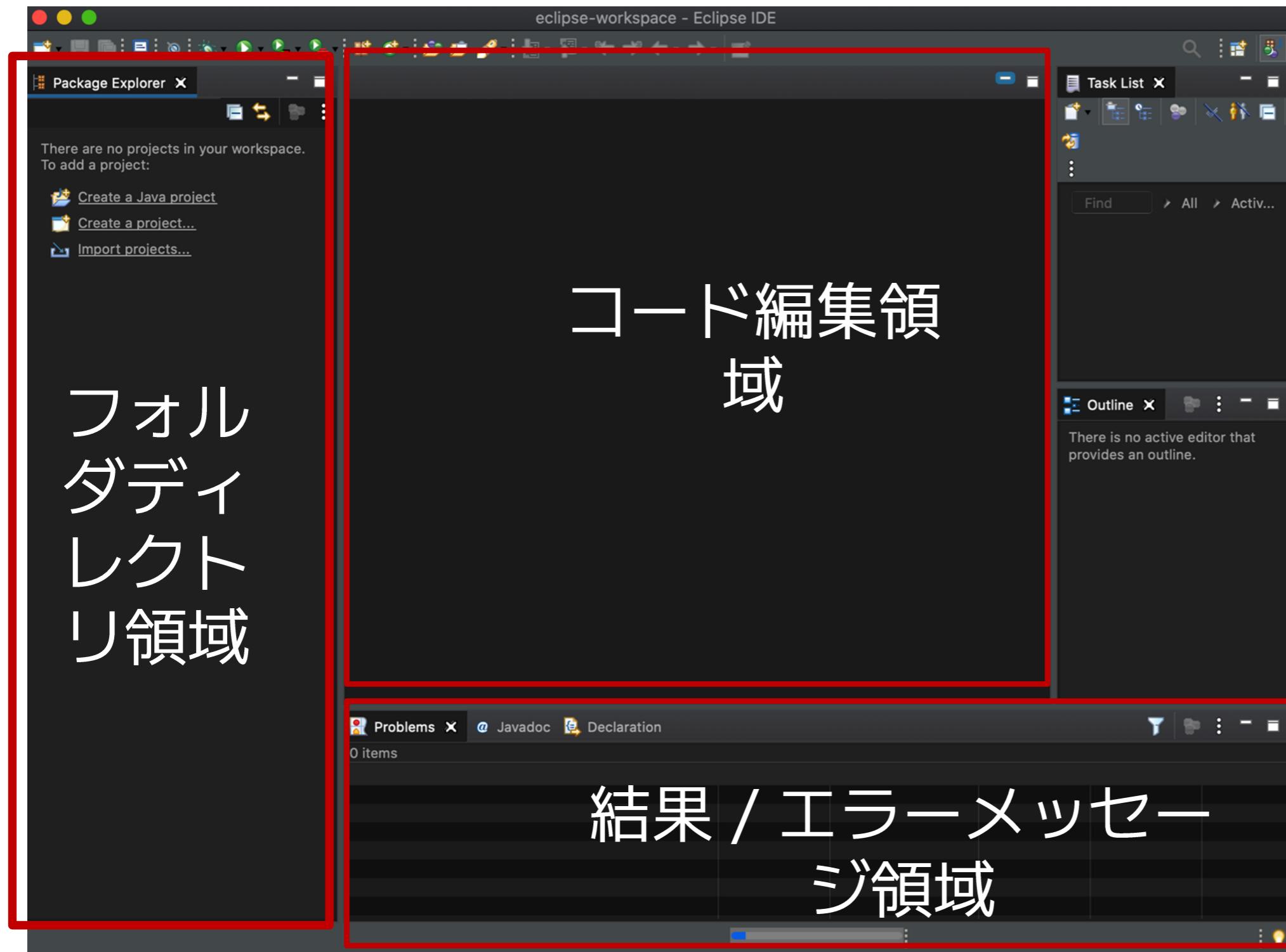


# Eclipse を起動

- ダブルクリックで Eclipse を起動します。開くたびにワークスペースの選択ダイアログが表示されますが、デフォルトのフォルダを使用しても構いません。Launchをクリックします。

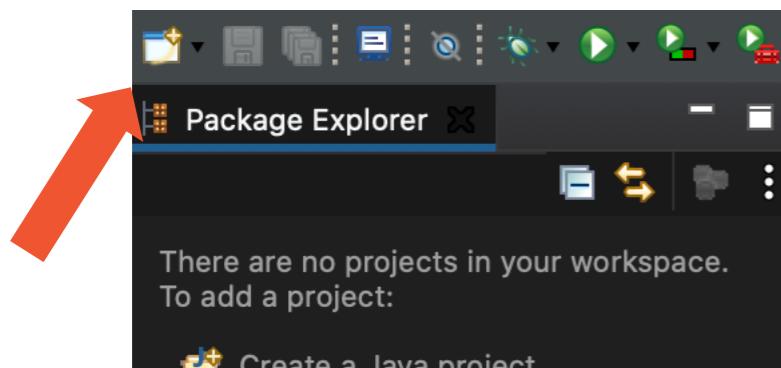


# Eclipse 画面

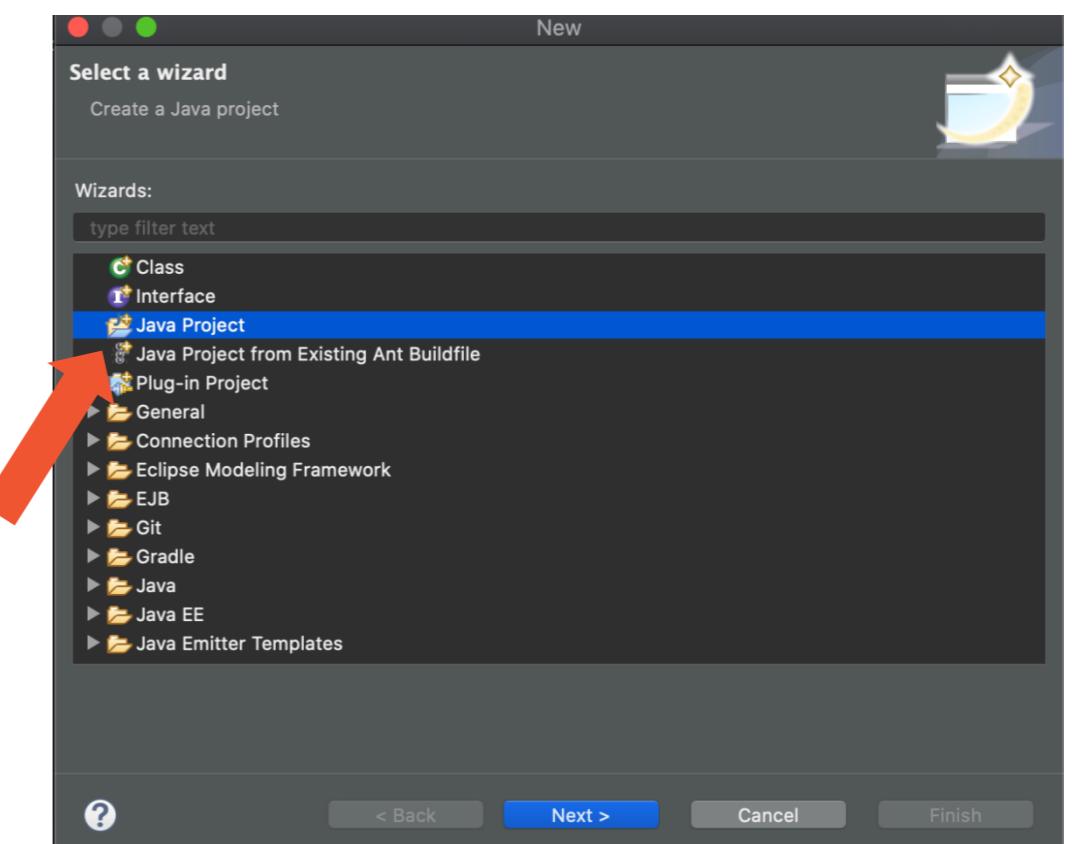


# Java プロジェクトの新規作成（1 / 2）

- 左上の新規作成アイコンをクリック：

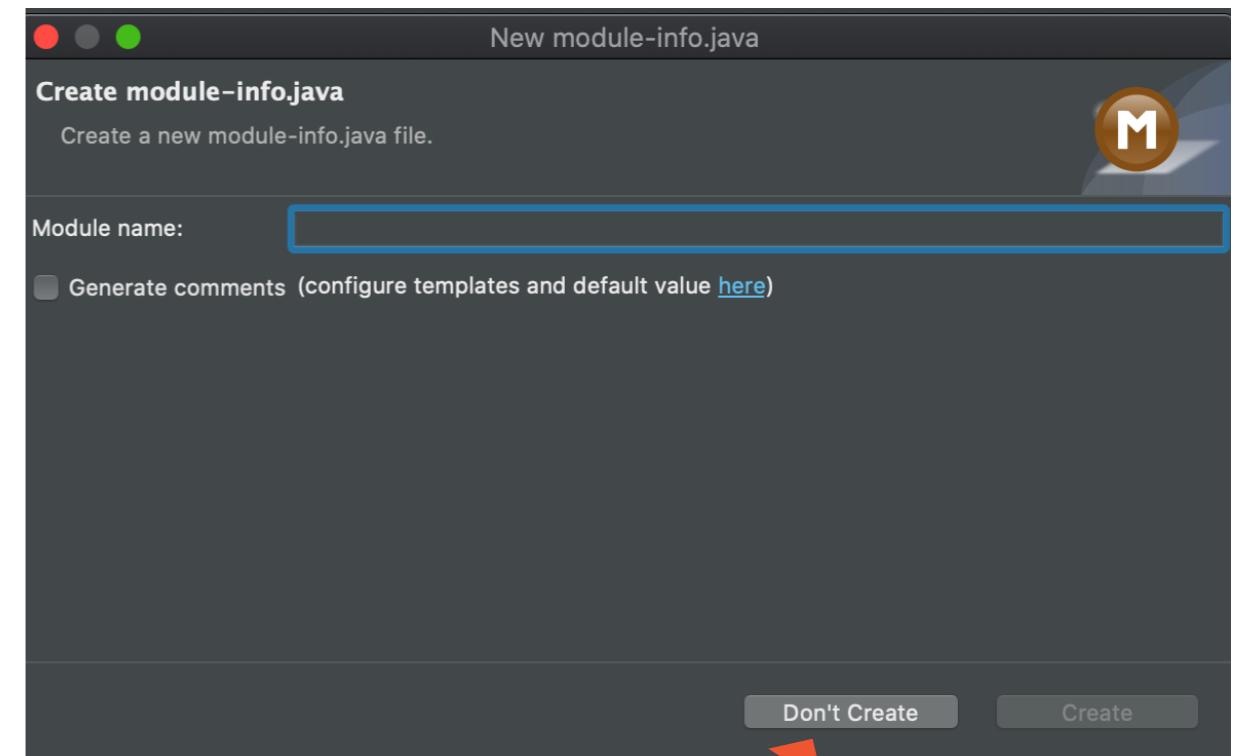
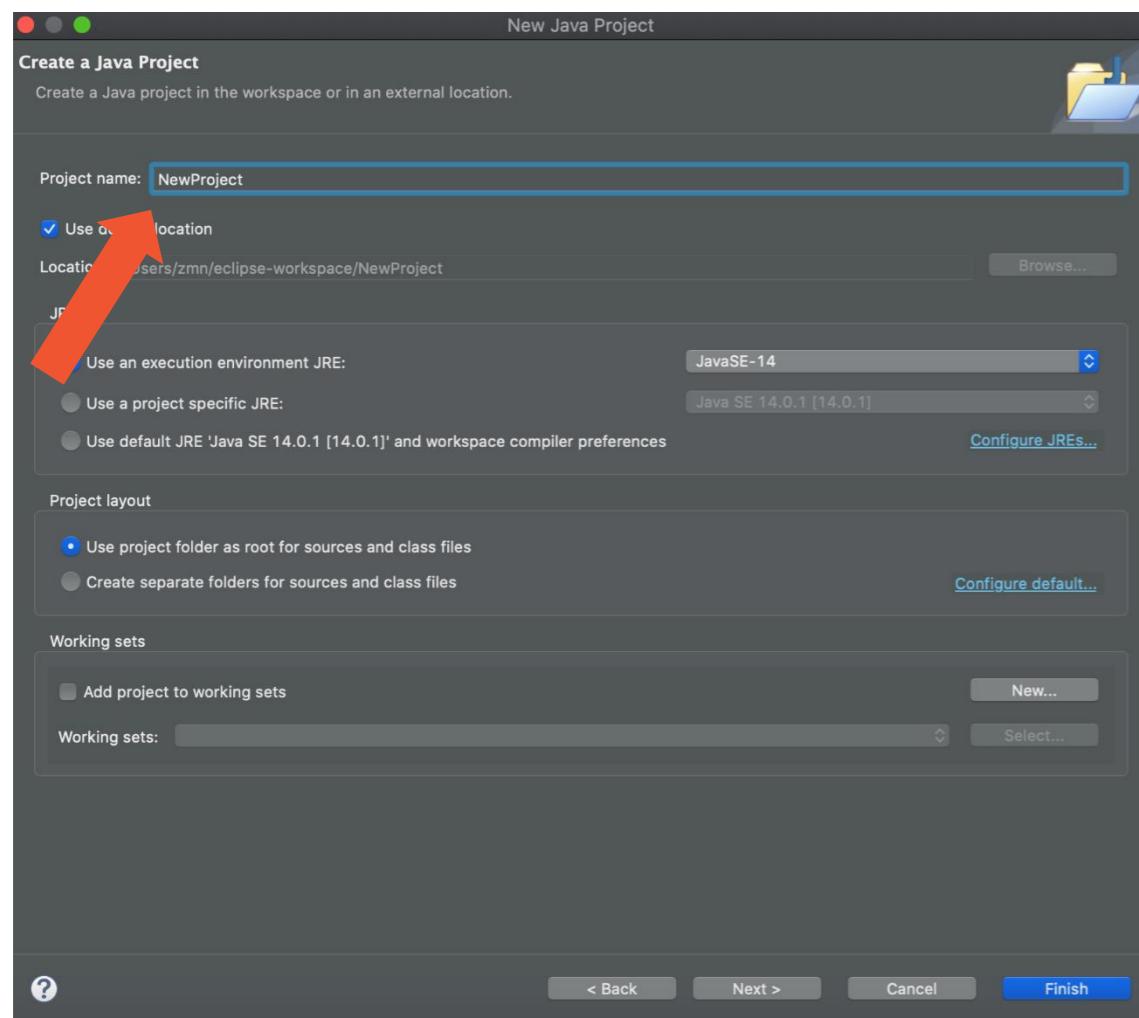


- Java Project → Next を選択：



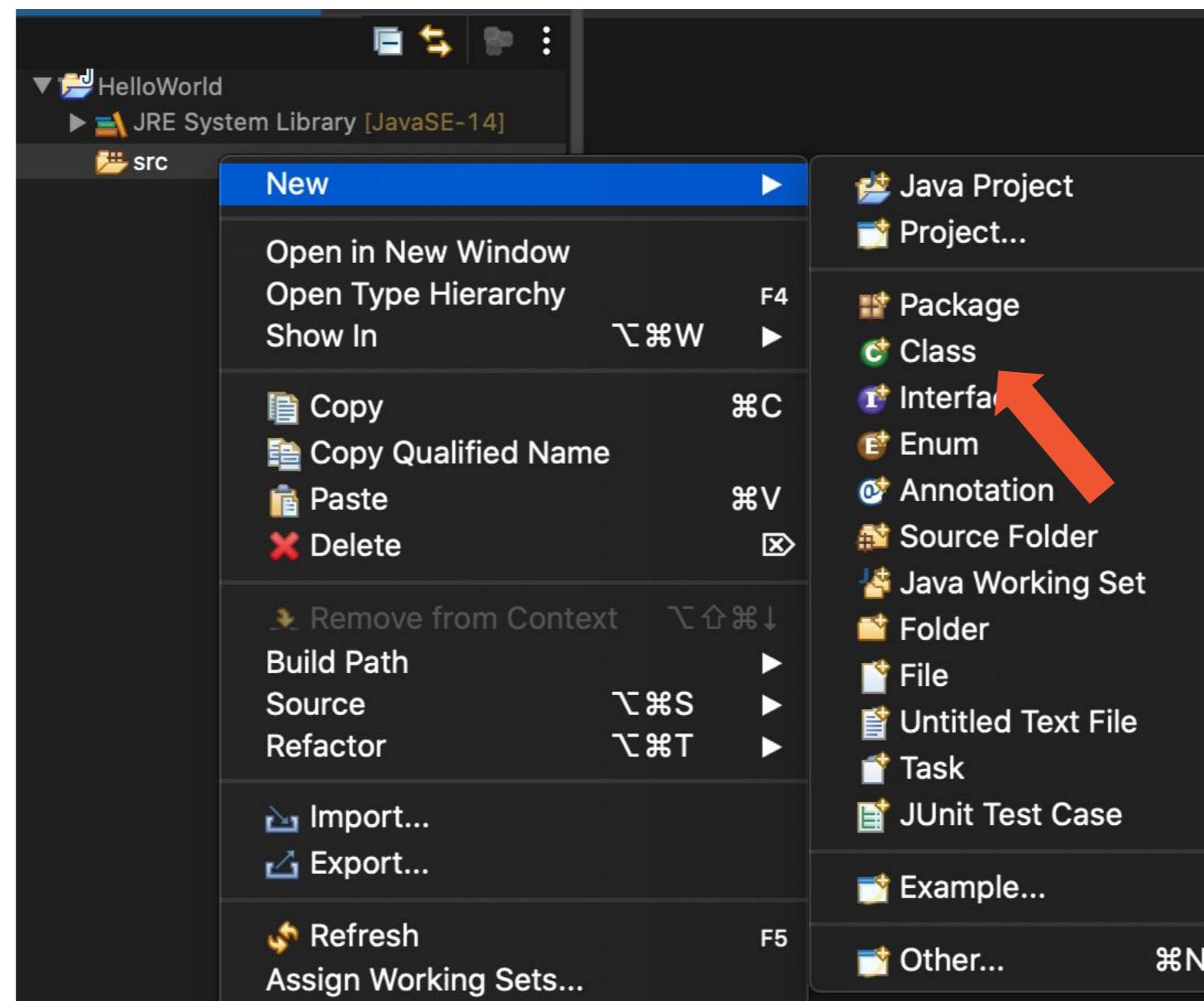
# Java プロジェクトの新規作成（2 / 2）

- Project name に、プロジェクトの名前を書く。
- あとはデフォルトで問題ないので、Finish をクリック。
- モジュールを作成するかどうか尋ねるウィンドウが表示されますが、Don't Create を選択してください。



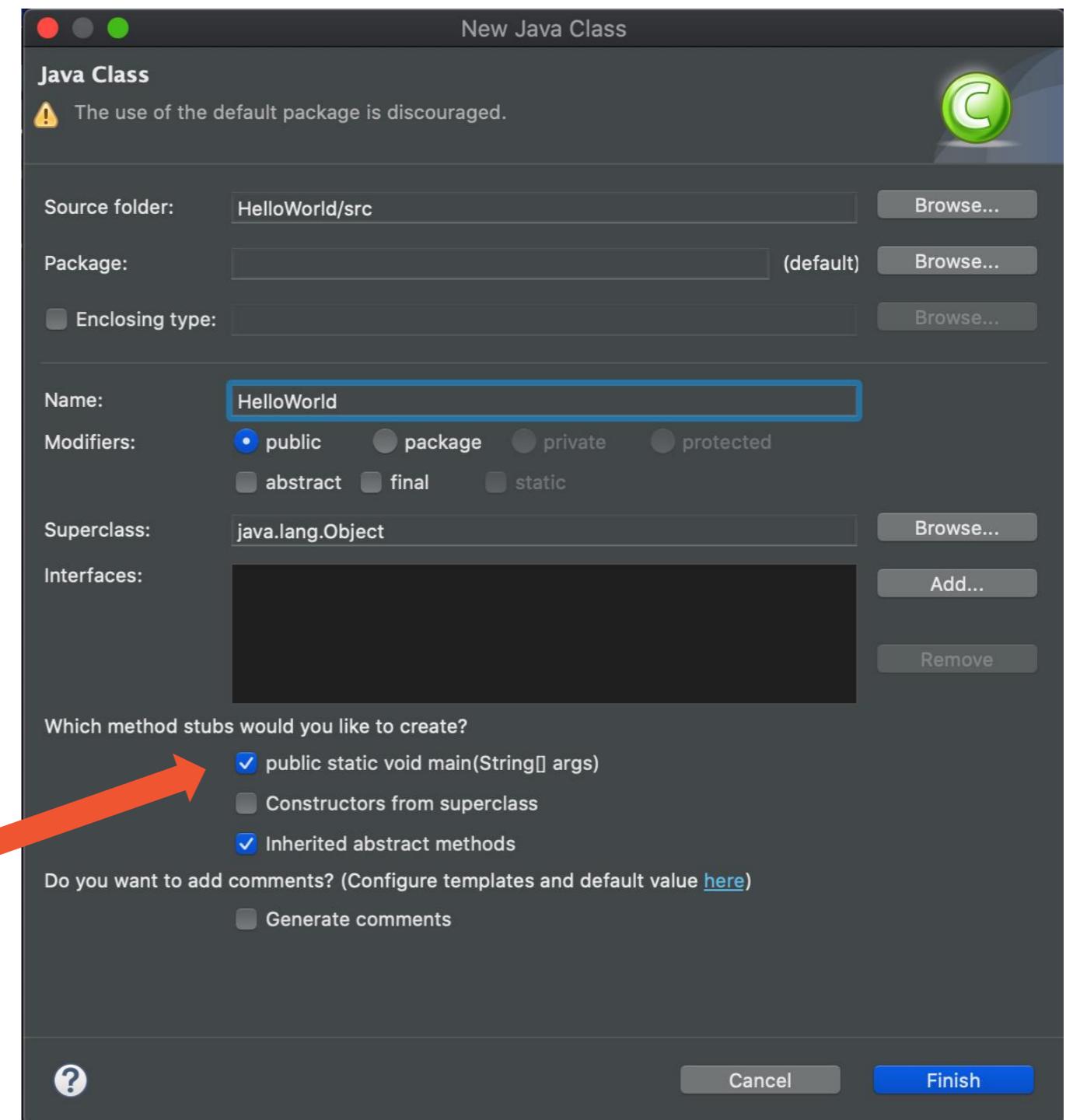
# Java クラスの新規作成（1 / 2）

- プロジェクトディレクトリにある `src` フォルダを右クリックし、`New → Class`。



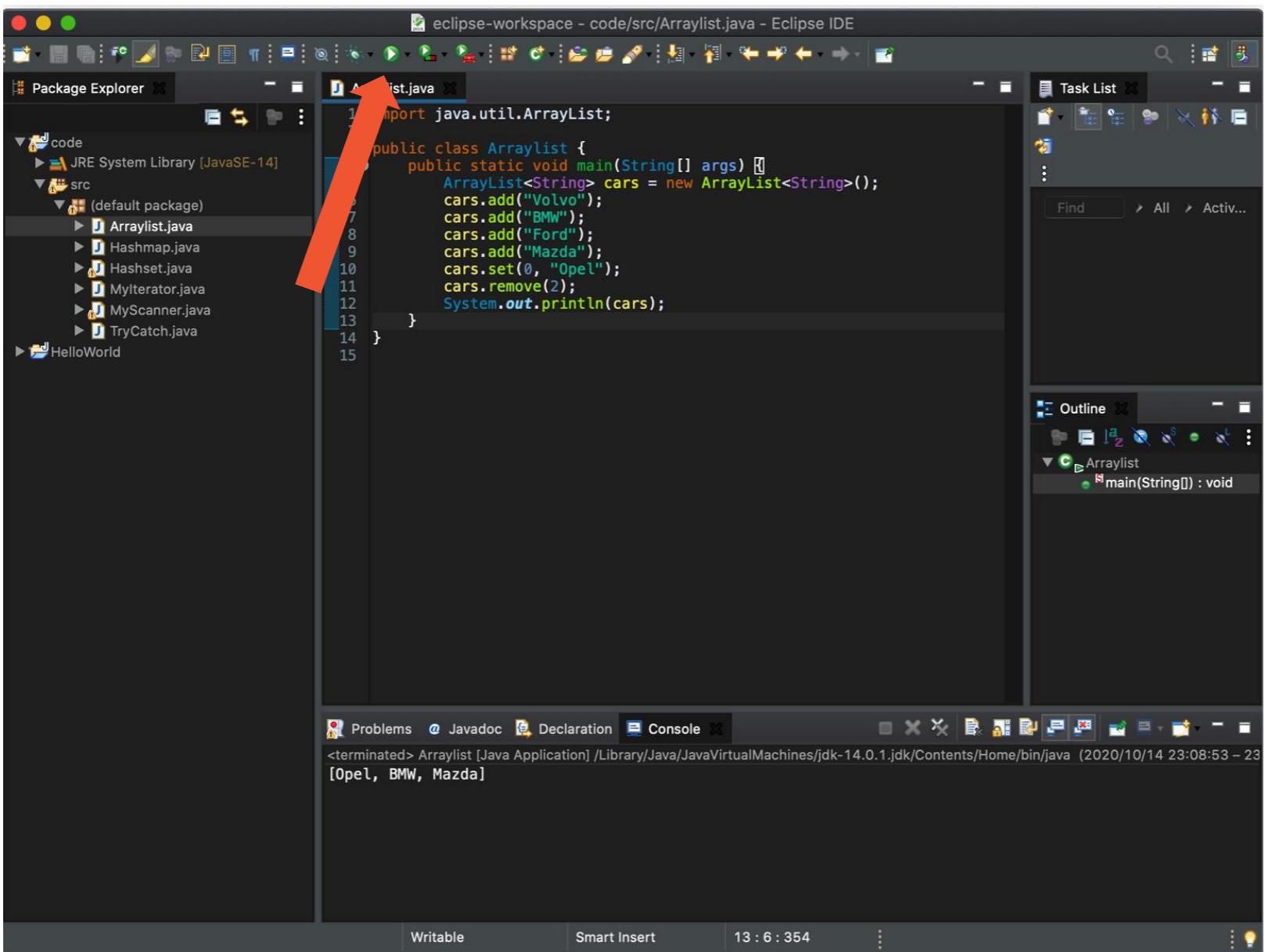
# Java クラスの新規作成（2 / 2）

**Tips**   
このボックスを  
チェックすると、メ  
インメソッドが自動  
的に生成されます。



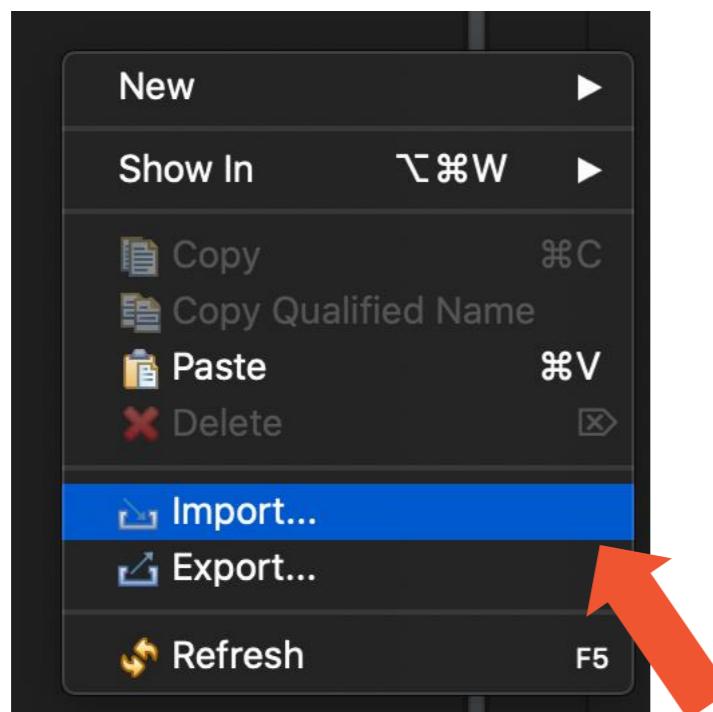
# コードの実行

- 緑色の口ゴ  をクリックして、今編集中のコードを実行することができます:



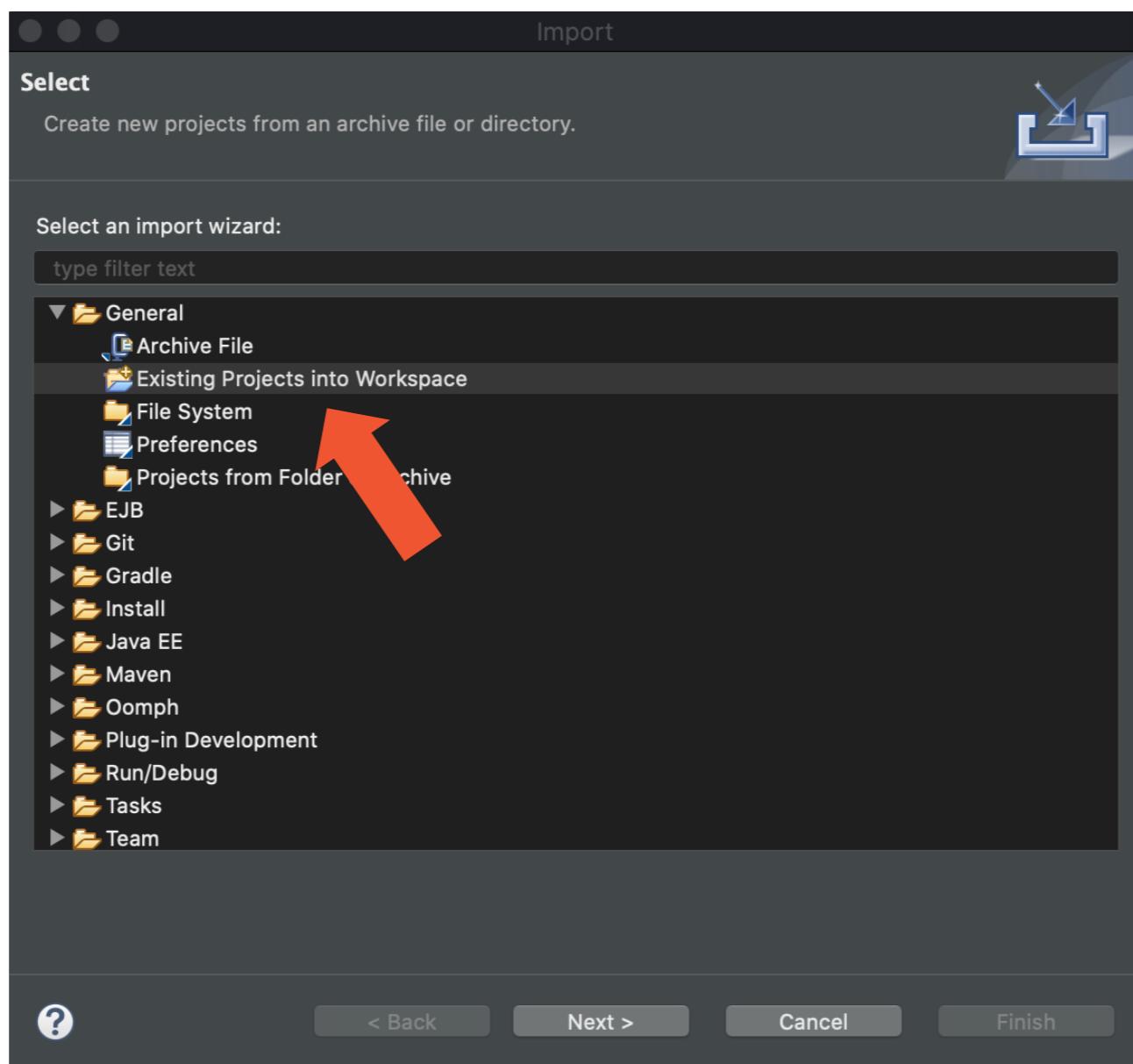
# プロジェクトをインポート（1 / 3）

- 今後コードの一部は、Eclipse プロジェクトとして送られますので、次の方法でインポートする必要があります。
- フォルダ領域で右クリックし、「Import」を選択します。



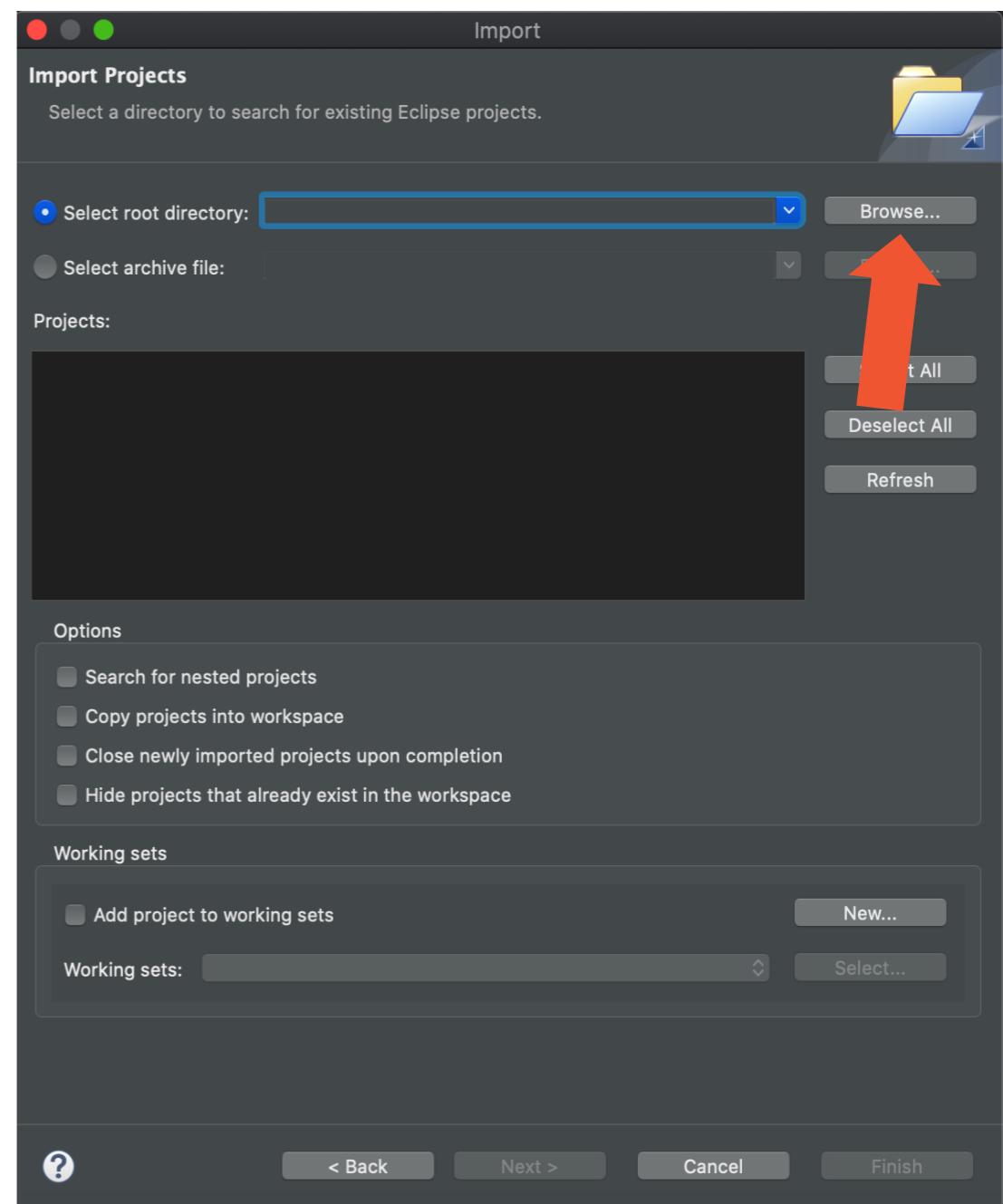
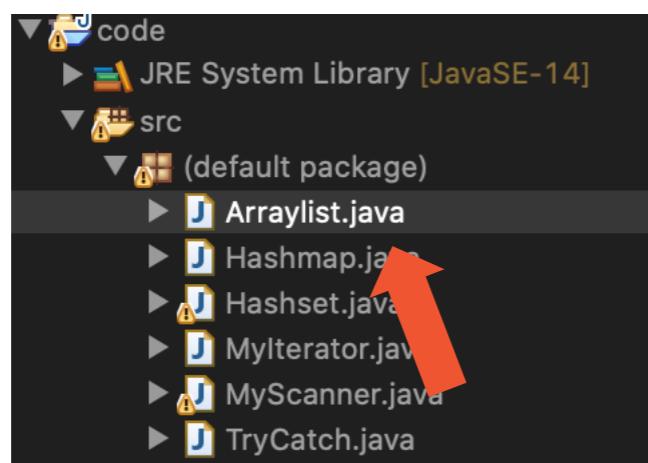
## プロジェクトをインポート（2 / 3）

- General を選択 → Existing Projects into Workspace → Next。



# プロジェクトをインポート（3 / 3）

- Browse をクリックし、プロジェクトを探して選びます。  
「Open」「Finish」を順次クリック。
- インポート後、Java ファイルをダブルクリックしてコードを表示します。





*Question and answer*

**THANK YOU!**