

# オブジェクト指向言語

## 第1回

講義概要・開発環境の導入・変数と型

# 目標

- ・ シラバスの的には…
  - ・ データ構造の設計とライブラリ化 云々
  - ・ 既存ライブラリの活用
  - ・ チーム開発
- ・ 要するに「Javaでまともなプログラムが書けるようになること」

# これまでの内容

- ・ プログラミング演習1
  - ・ PythonとCを見よう見まねで学ぶ
- ・ プログラミング演習2
  - ・ Cをガチで学ぶ
- ・ プログラミング演習3
  - ・ Pythonでオブジェクト指向の入門を学ぶ

# なぜJava？

- ・ 大規模開発で利用されるオブジェクト指向言語
- ・ 「静的に強く型付けされた言語」をしっかりと学ぶべき
- ・ Cのように、変数宣言と型の指定が必要
- ・ プログラムの実行前に、できるだけ誤りを発見するための仕組み
- ・ プログラムの実行時にエラーが起きると、販売したプログラムがお客様の前でエラーを起こす ← これを避けたい

# 講義内容（予定）

- ・ 第1回 (2023/09/15) 講義概要・開発環境の導入・変数と型
- ・ 第2回 (2023/09/22) Java: 制御構造、配列、メソッド
- ・ 第3回 (2023/09/29) 文字列、ファイル入出力と例外
- ・ 第4回 (2023/10/06) 代表的なデータ構造(1)
- ・ 第5回 (2023/10/13) 機能のライブラリ化
- ・ 第6回 (2023/10/20) 代表的なデータ構造(2)
- ・ 第7回 (2023/10/27) GUIプログラミング・GUI作成実習1
- ・ 第8回 (2023/11/10) GUI作成実習2
- ・ 第9回 (2023/11/17) 画面描画、インタフェース、ポリモルフィズム
- ・ 第10回 (2023/11/24) ネットワーク、スレッド
- ・ 第11回 (2023/12/01) プロジェクト(1)
- ・ 第12回 (2023/12/08) プロジェクト(2)
- ・ 第13回 (2023/12/15) プロジェクト(3) 中間報告
- ・ 第14回 (2024/01/05) プロジェクト(4) デモ
- ・ 第15回 (2024/01/19) プロジェクト(5) 成果発表

下線の回は出張かも知ですが、教室に集合 + TA/SAサポートで演習実施

# 受講の前提

- ・ 自分で調べ、試行錯誤し、なぜそうなるかを考え、必要に応じて質問できる
- ・ GUIプログラミングはおまじないが多い
  - ・ 多くの調査、経験が必要 → 自習が大事
- ・ 正直、レポート課題はちょっと重いです
  - ・ 予習より、復習 + レポートが大事

# プログラミングの学習

- ・ Yahoo知恵袋に珍しくいいことが書いてあった
  - ・ [http://detail.chiebukuro.yahoo.co.jp/qa/question\\_detail/q1313696133](http://detail.chiebukuro.yahoo.co.jp/qa/question_detail/q1313696133)
- ・ 自分で書いて、いじってなんぼ
  - ・ サンプルコードも自分の手で打ち込む
  - ・ ところどころを書き換えて試してみる
  - ・ 変更とその結果を考察する

# この授業の受け方

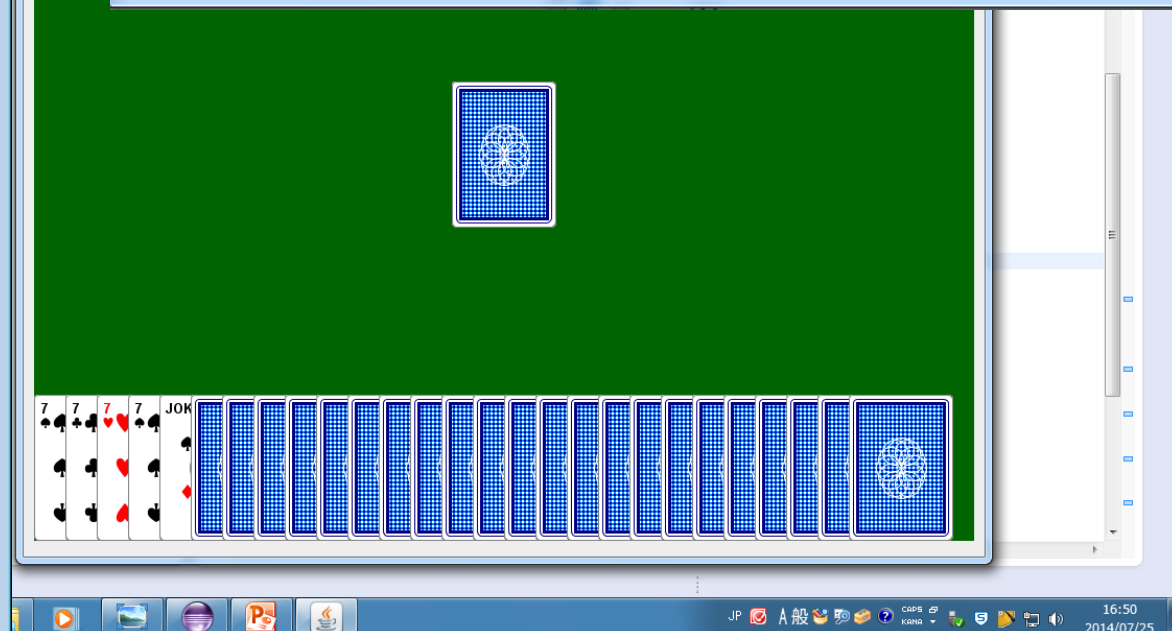
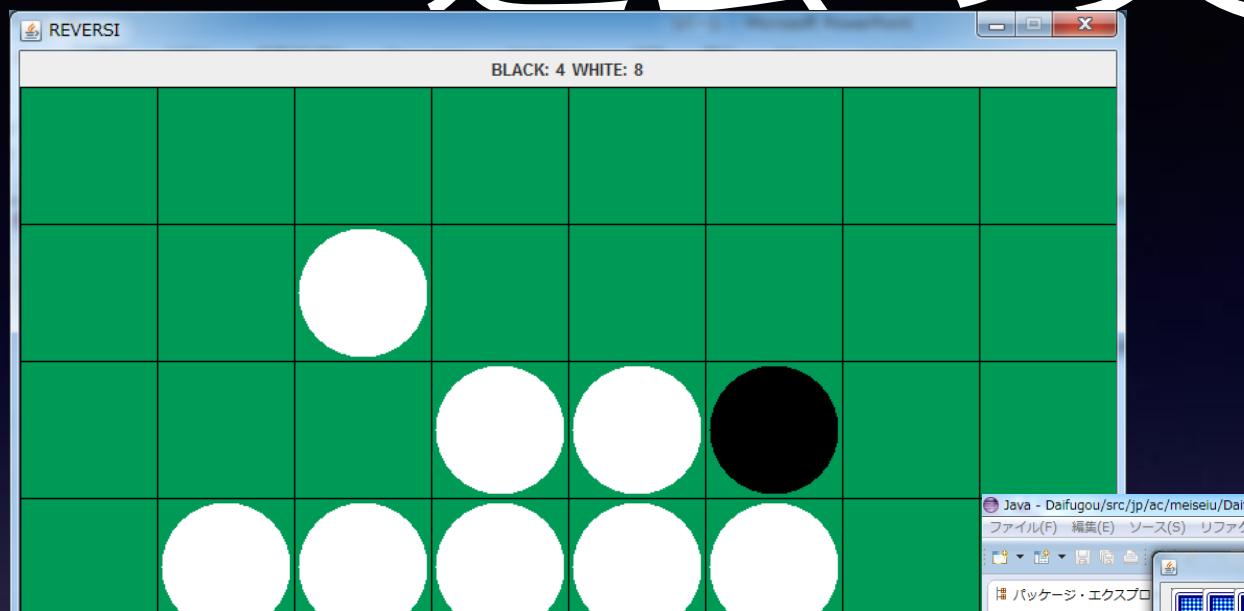
- ・ 資料を読む、説明を聞く
- ・ サンプルコードを開発環境に手入力して試す
  - ・ わざと間違えたり、書き換えて結果がどう変わるかを（エラーも含めて）確認
- ・ 演習に取り組む
  - ・ 解答例を授業中に公開するので、答え合わせ + 書き方が違うところ等を吟味 + 質問
- ・ その週のレポートに取り組む



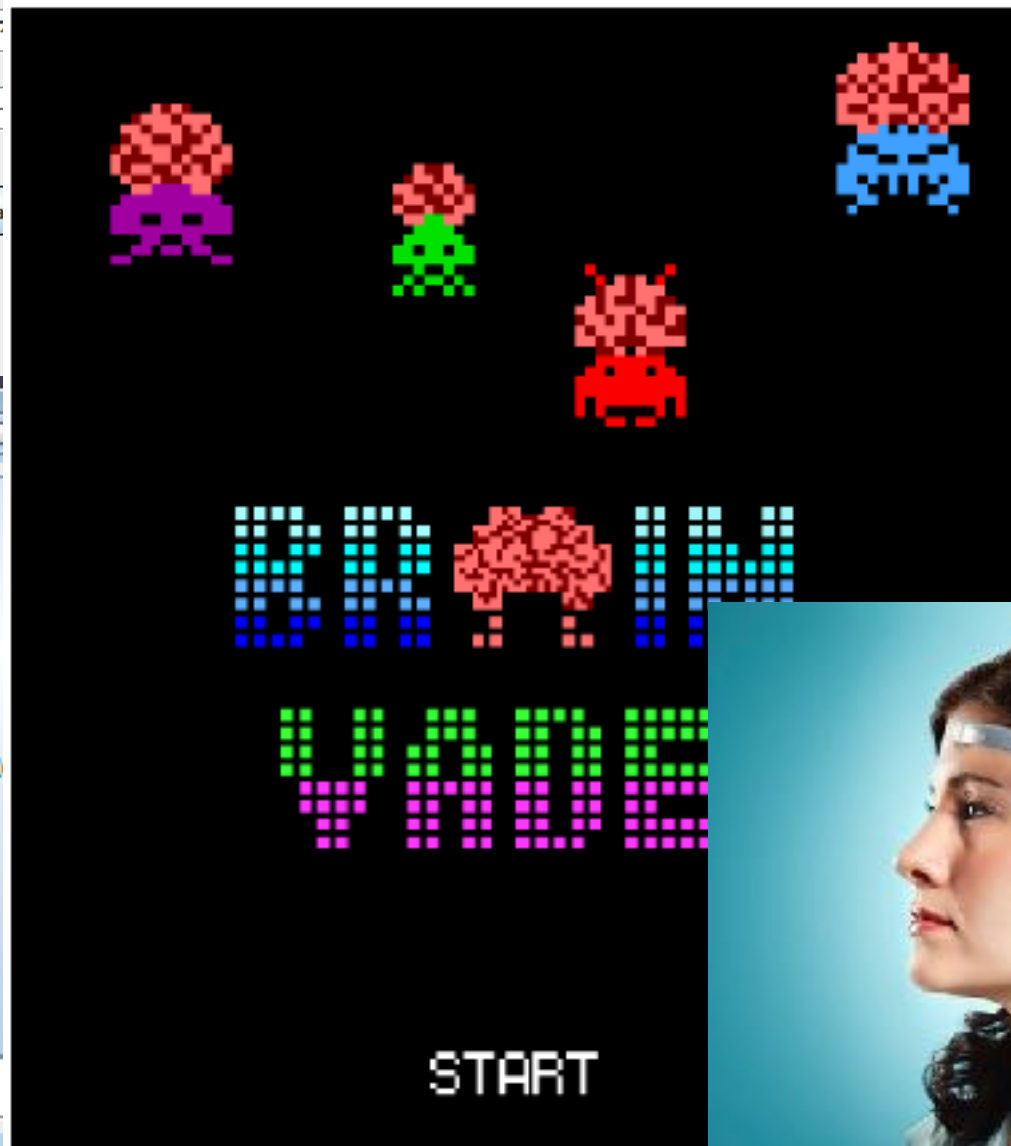
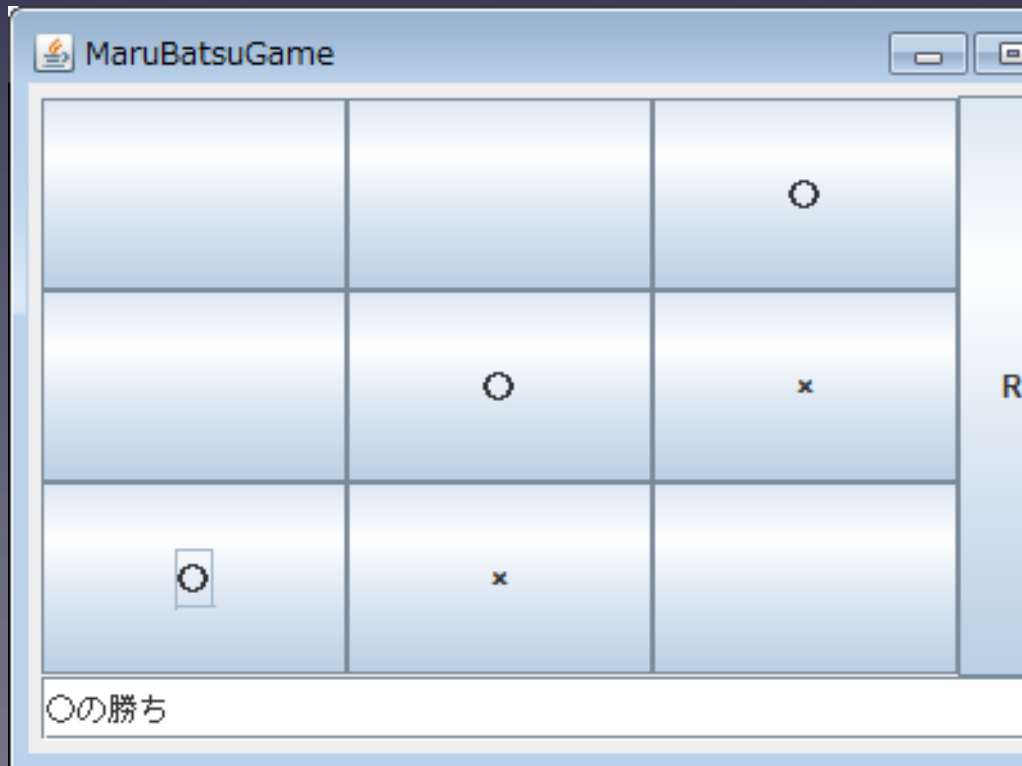
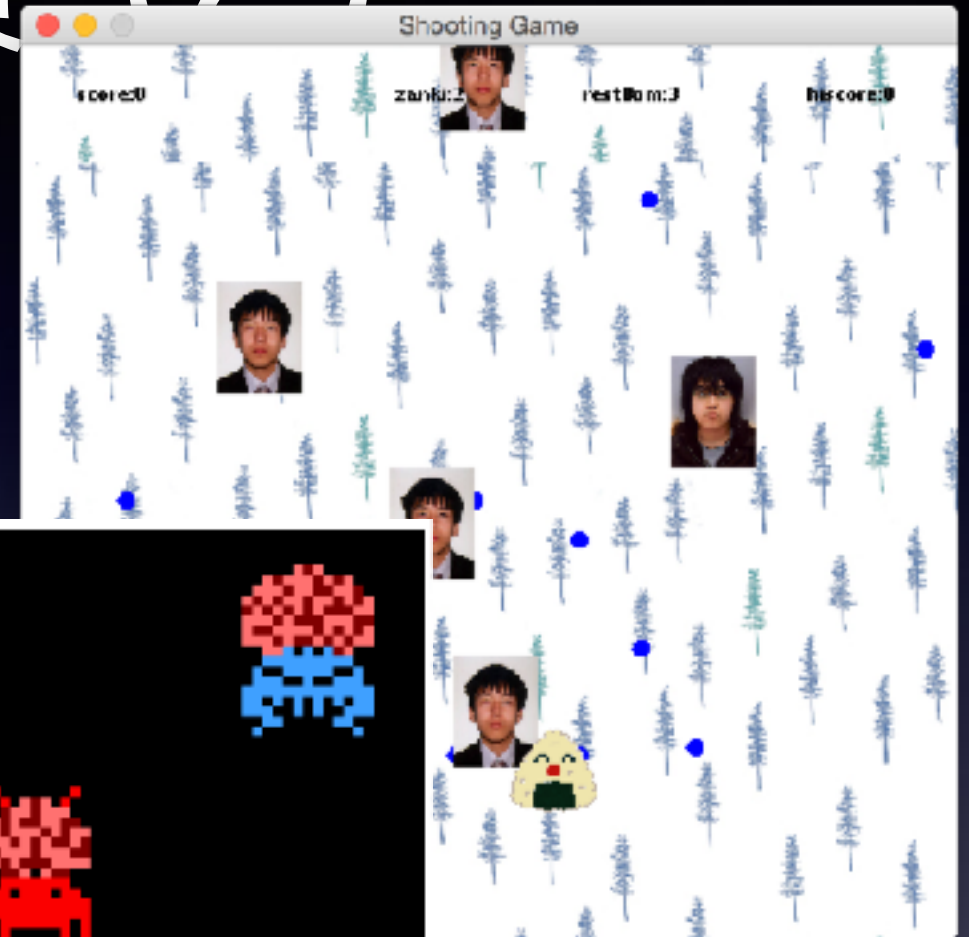
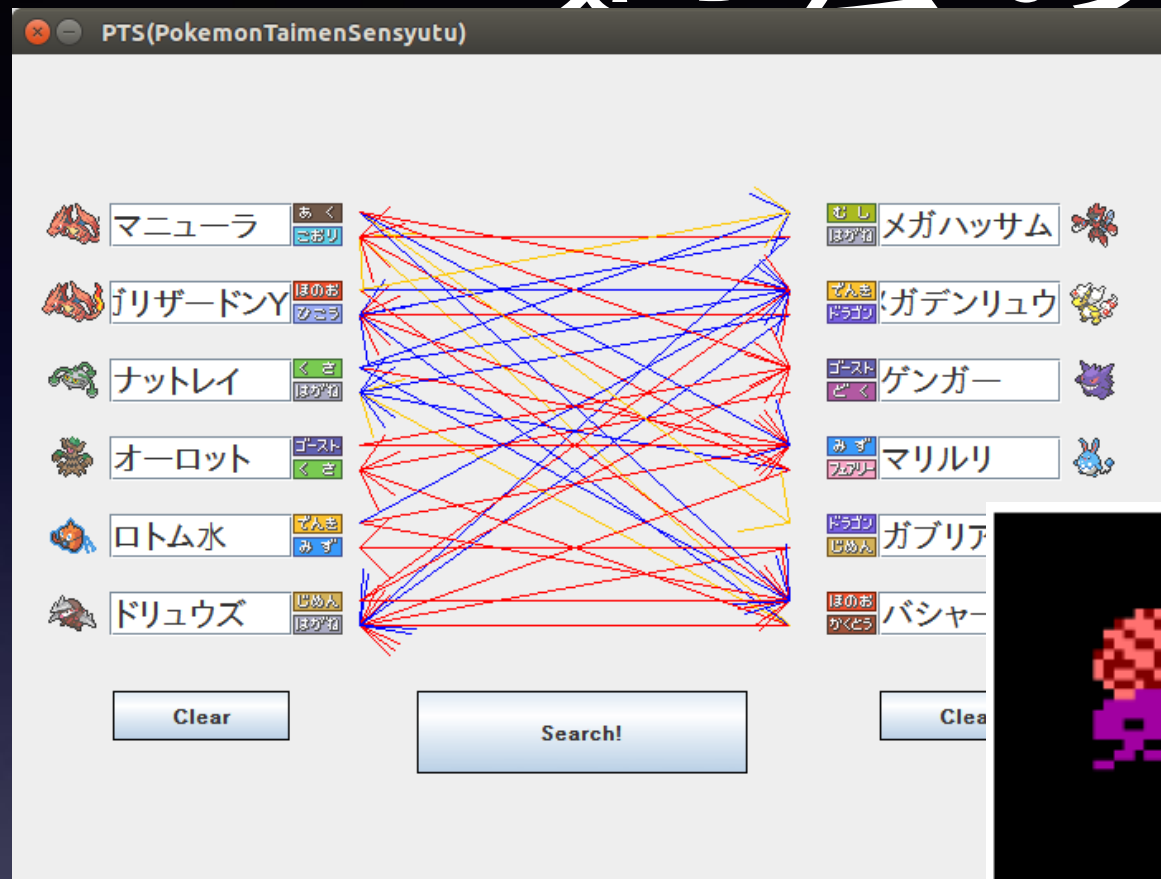
# 成績評価

- ・ 出席
  - ・ 11回以上の出席が必須、学生証のタッチで
  - ・ 理由がある場合は証明を
    - ・ 交通機関 → 遅延証明書
    - ・ 病欠 → 診断書
    - ・ 不祝儀 → 会葬礼状等
    - ・ ワクチン接種の副反応 → 接種済証
- ・ レポート(10回程度、最終は期末試験相当)
- ・ プロジェクト課題の成果発表

# 過去の実績 (1)




# 過去の実績 (2)





# 過去の実績 (3)

362  
time : 6



score :

ワンペア


♠ 5 ♣ 1 ♦ 13 ♠ 5 ♣ 4


♠ 7 J 0 ♠ 6 ♠ 9 ♠ 7

のこす のこす のこす

リセット

1 2 3 4 5





0 1 2 3 4

# Javaの特長

- ・ C/C++に似た構文
- ・ オブジェクト指向
- ・ バイトコードと仮想マシンによる実行
- ・ Portability（可搬性）
  - ・ Write once, run anywhere.

# Javaの動作の仕組み(1)

Cの場合

ソースコード

test.c

コンパイラ

gcc

実行コード

a.out

実行コード

OS

コンピュータ・ハードウェア

OS/HW依存

Javaの場合

ソースコード

Test.java

コンパイラ

javac

バイトコード

Test.class

Javaバイトコード

Java仮想マシン

OS

コンピュータ・ハードウェア

OS/HW非依存

OS/HW依存

# Javaの動作の仕組み(2)

- ・ 仮想マシン
  - ・ バイトコードを実行コードに変換しながらプログラムを実行する（インタプリタ）
  - ・ 参考：JIT（Just-In-Time）コンパイラ
    - ・ 複数回実行されるコードをその場でコンパイル
    - ・ コンパイル済みのコードを再利用

# Javaプログラミングの環境

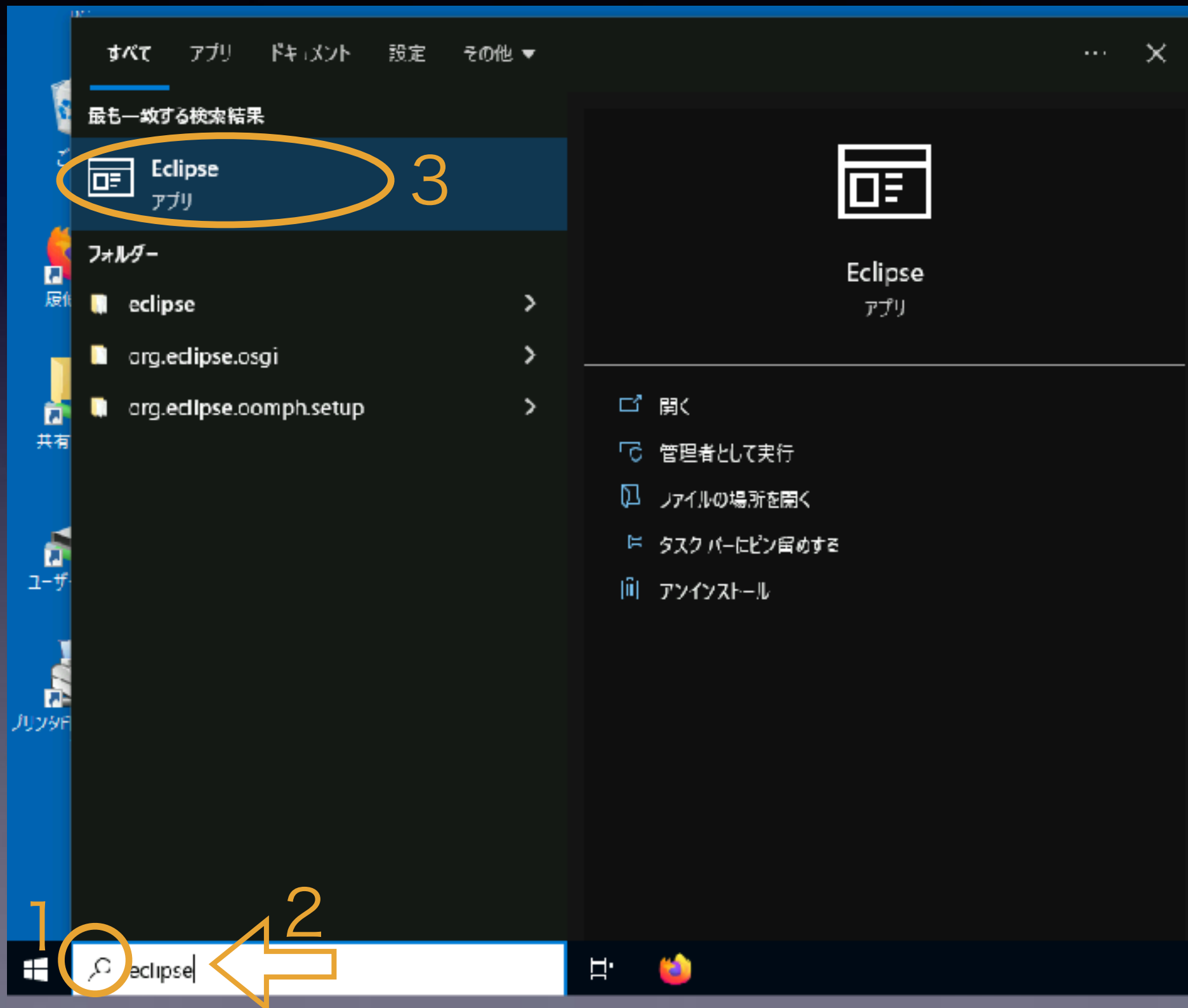
- ・ 一般的にはIDE（統合開発環境）を利用
  - ・ JavaではEclipse、NetBeans、IntelliJなど
- ・ 本講義ではEclipseを利用
  - ・ 仮想Windows環境に導入済み



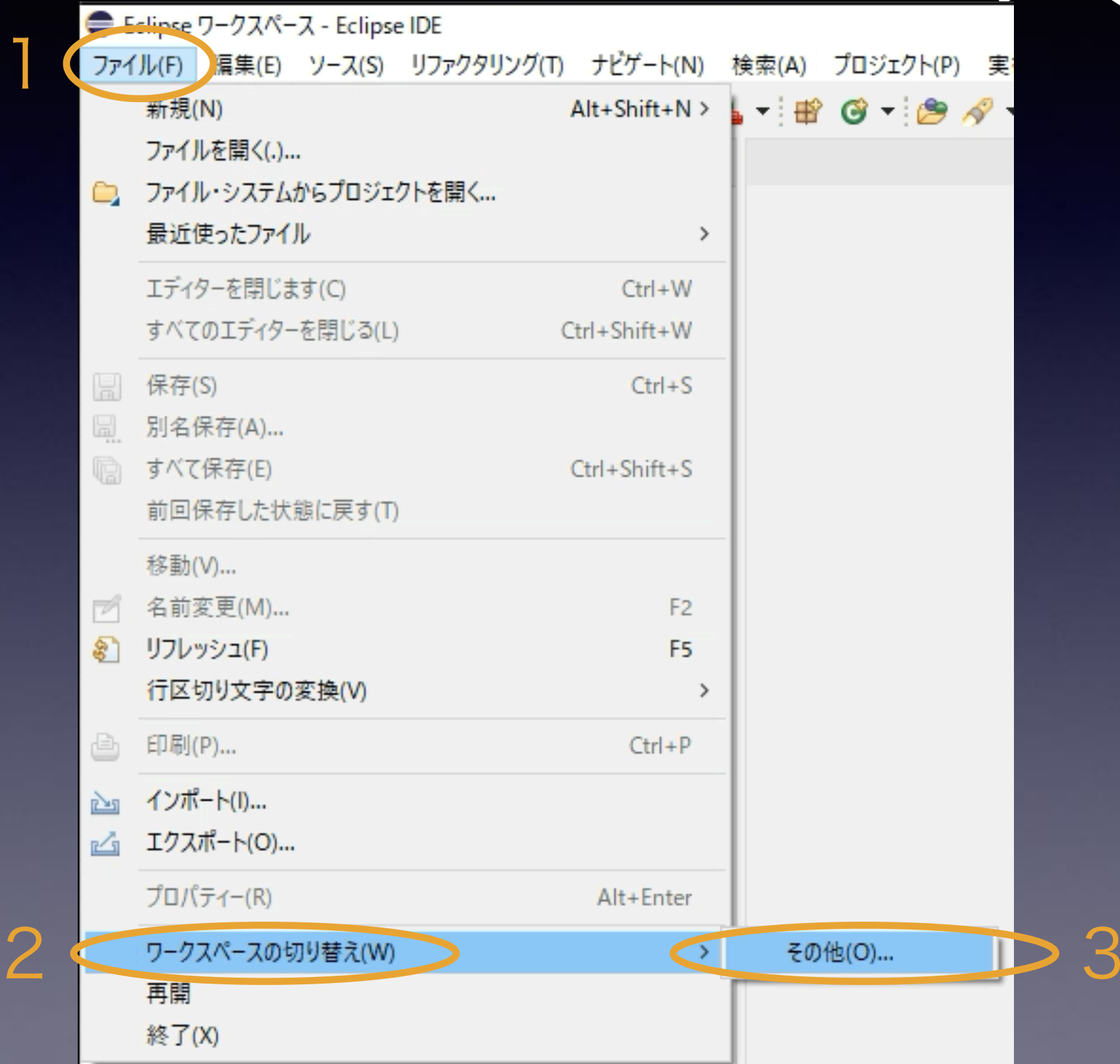
# 演習環境の利用 (1)

- ・ ITサポートサイトのマニュアル M172
  - ・ <https://its.hino.meisei-u.ac.jp/manual/m172/>
- ・ [プライベートモードのブラウザで https://vc.stu.meisei-u.ac.jp/](https://vc.stu.meisei-u.ac.jp/) に接続してサインイン
- ・ 科目名の記載がある仮想マシンの「接続」をクリック
  - ・ US配列キーボードの場合は、画面上のメニューで「English (en-US)」を選ぶこと

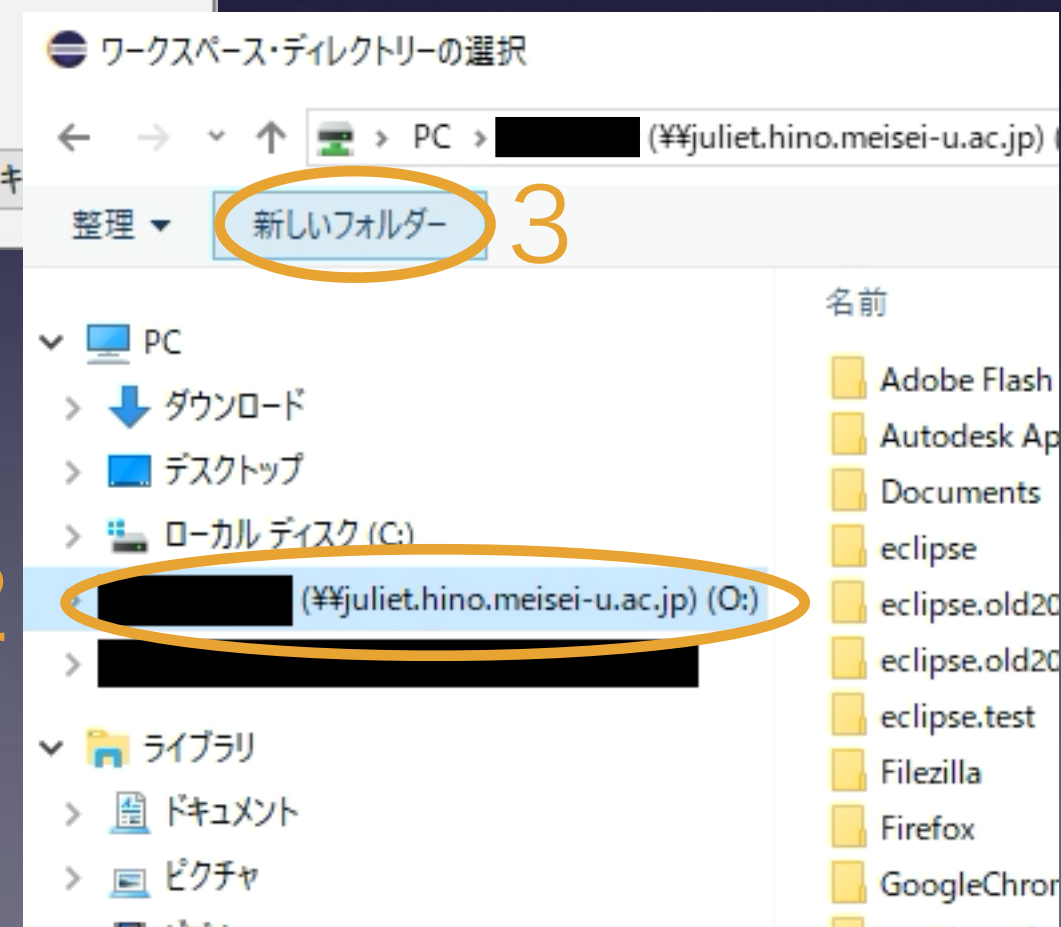
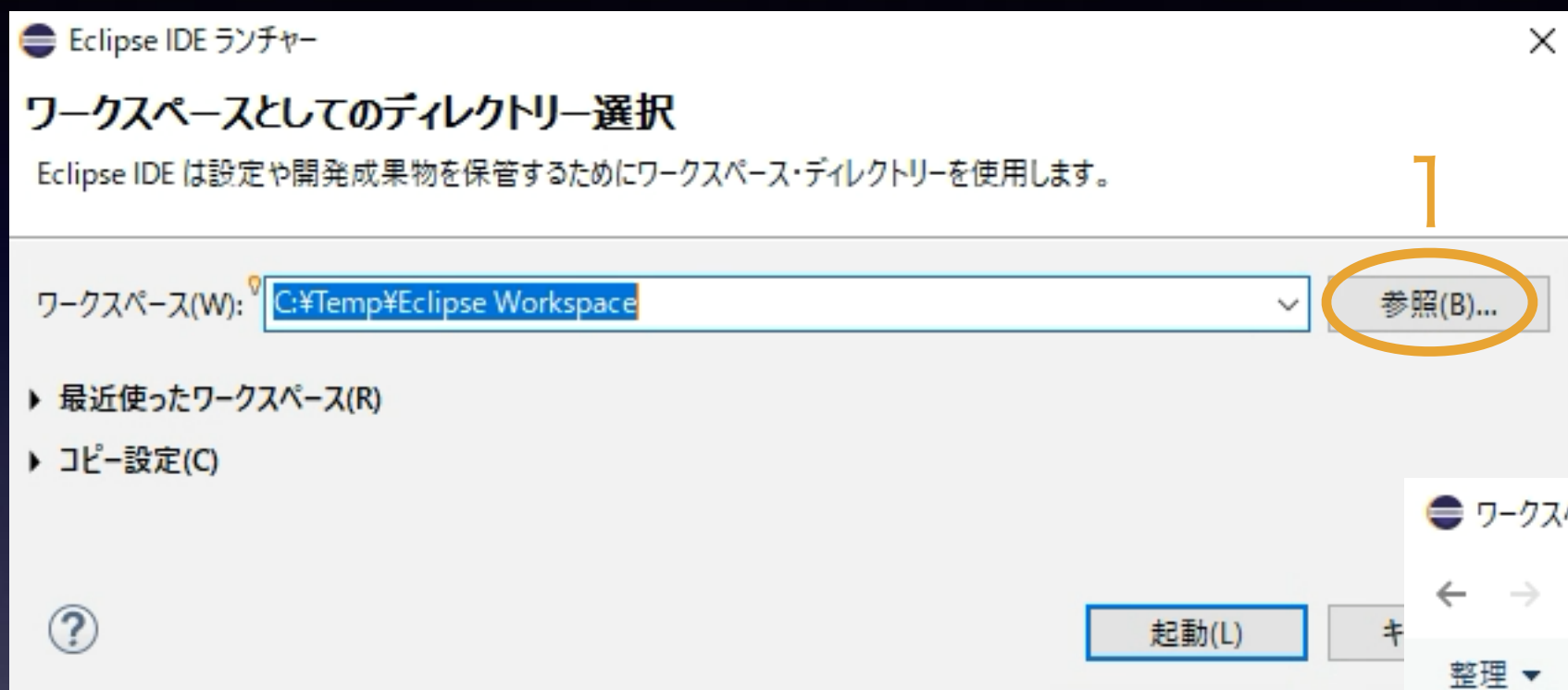
# 演習環境の利用 (2)



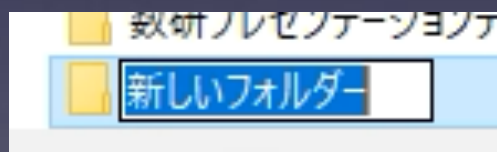
# 演習環境の利用 (3)



# 演習環境の利用 (4)

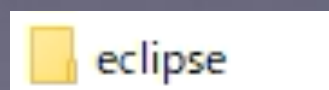


4

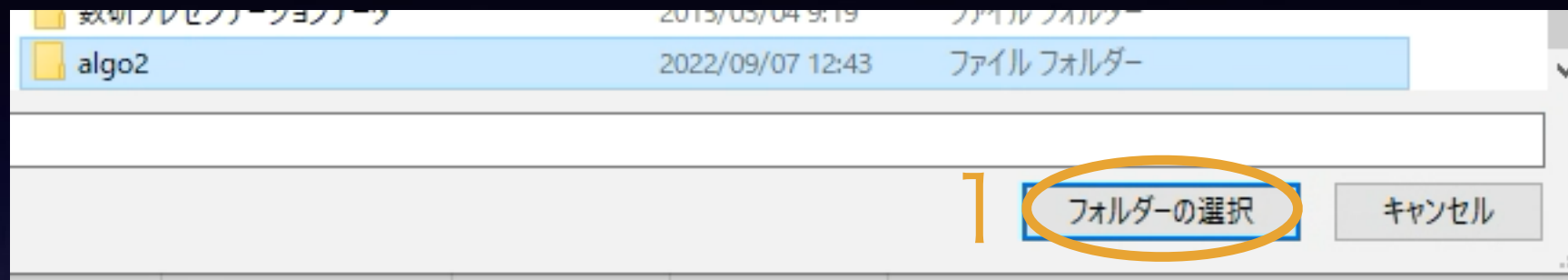


↓ eclipse 【Enter】

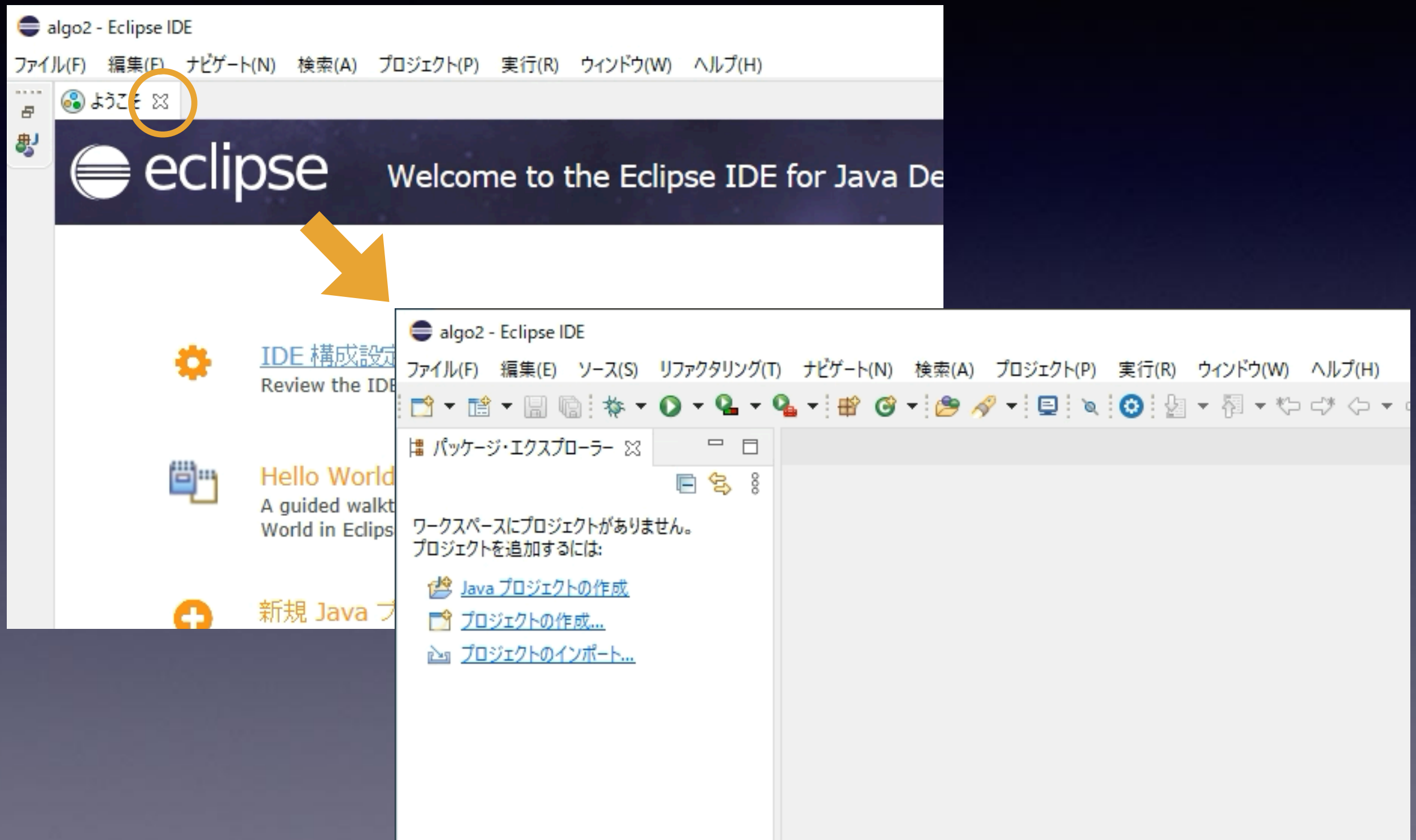
5



# 演習環境の利用 (5)

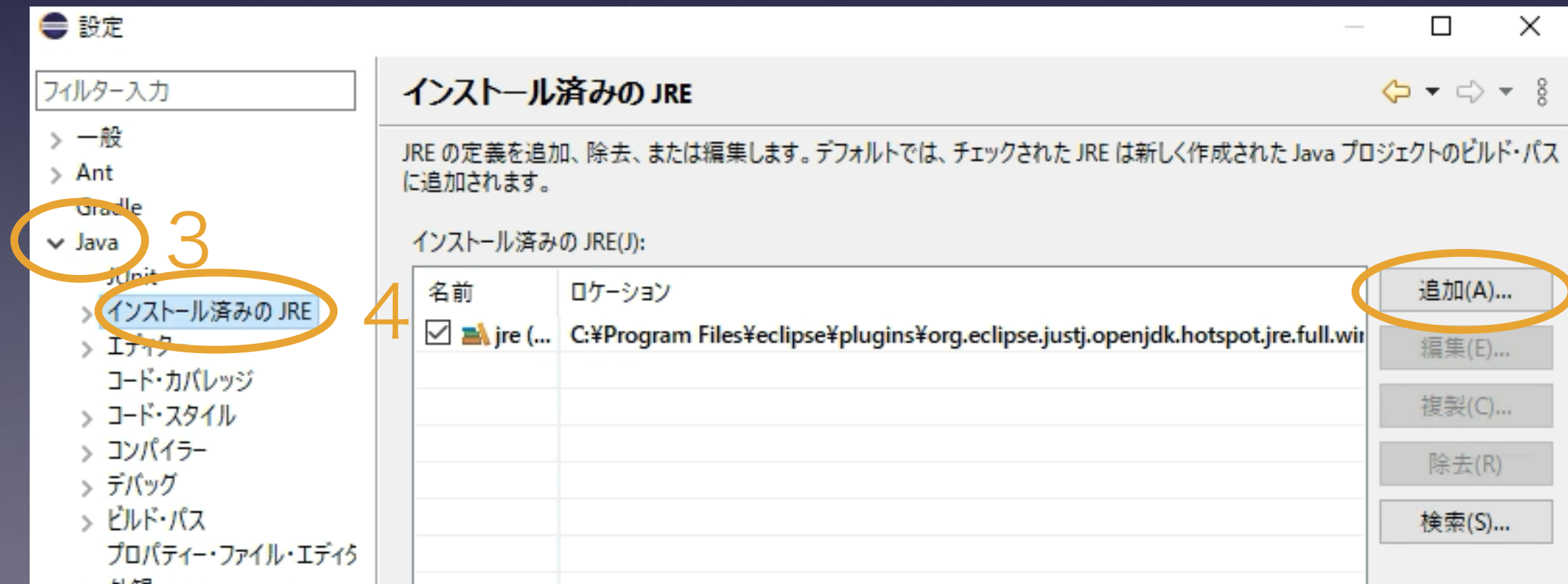
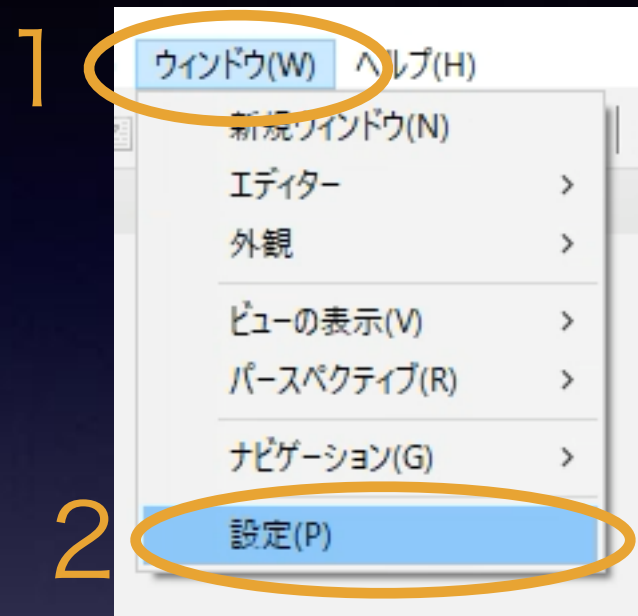


# 演習環境の利用 (6)



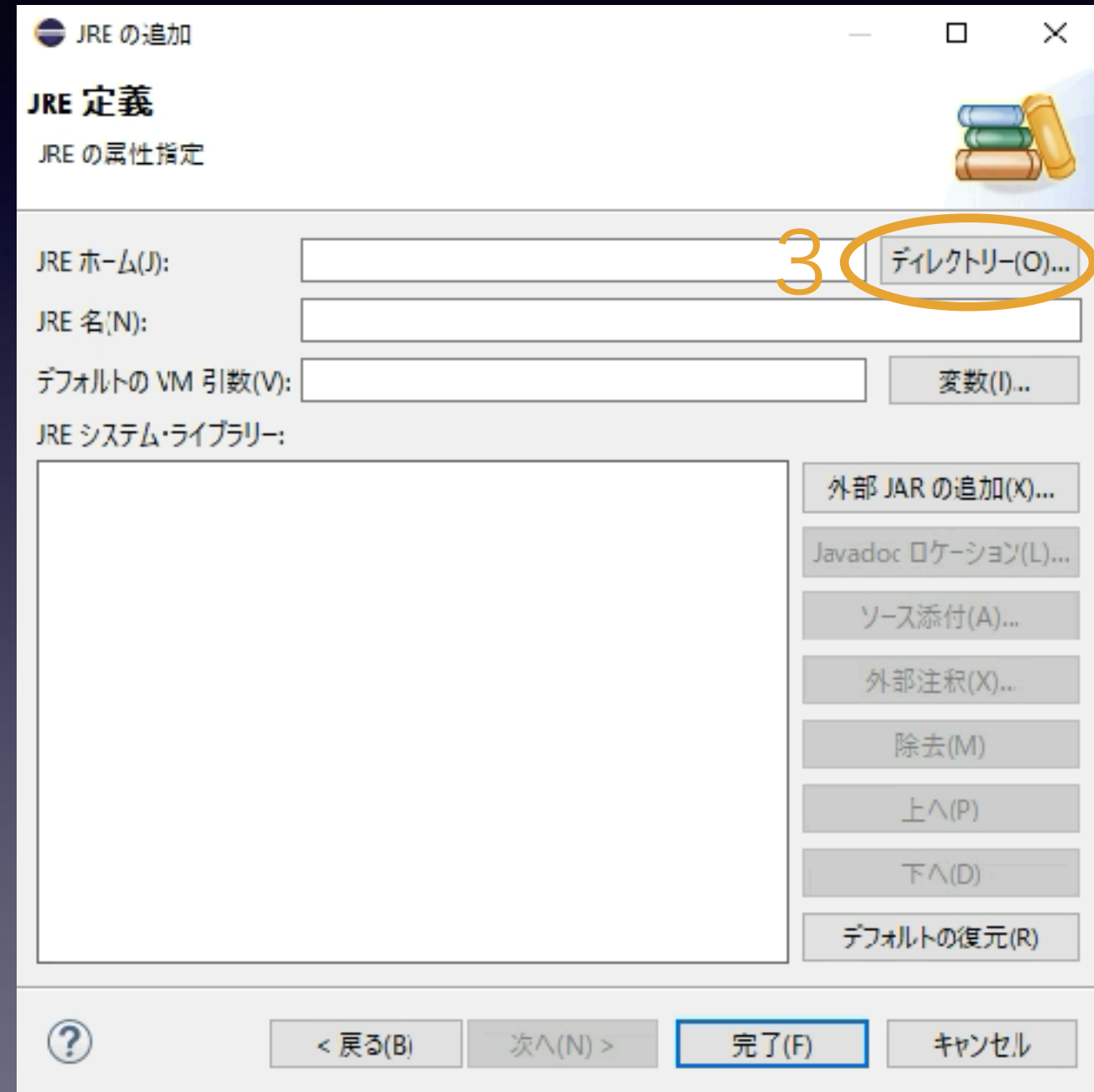
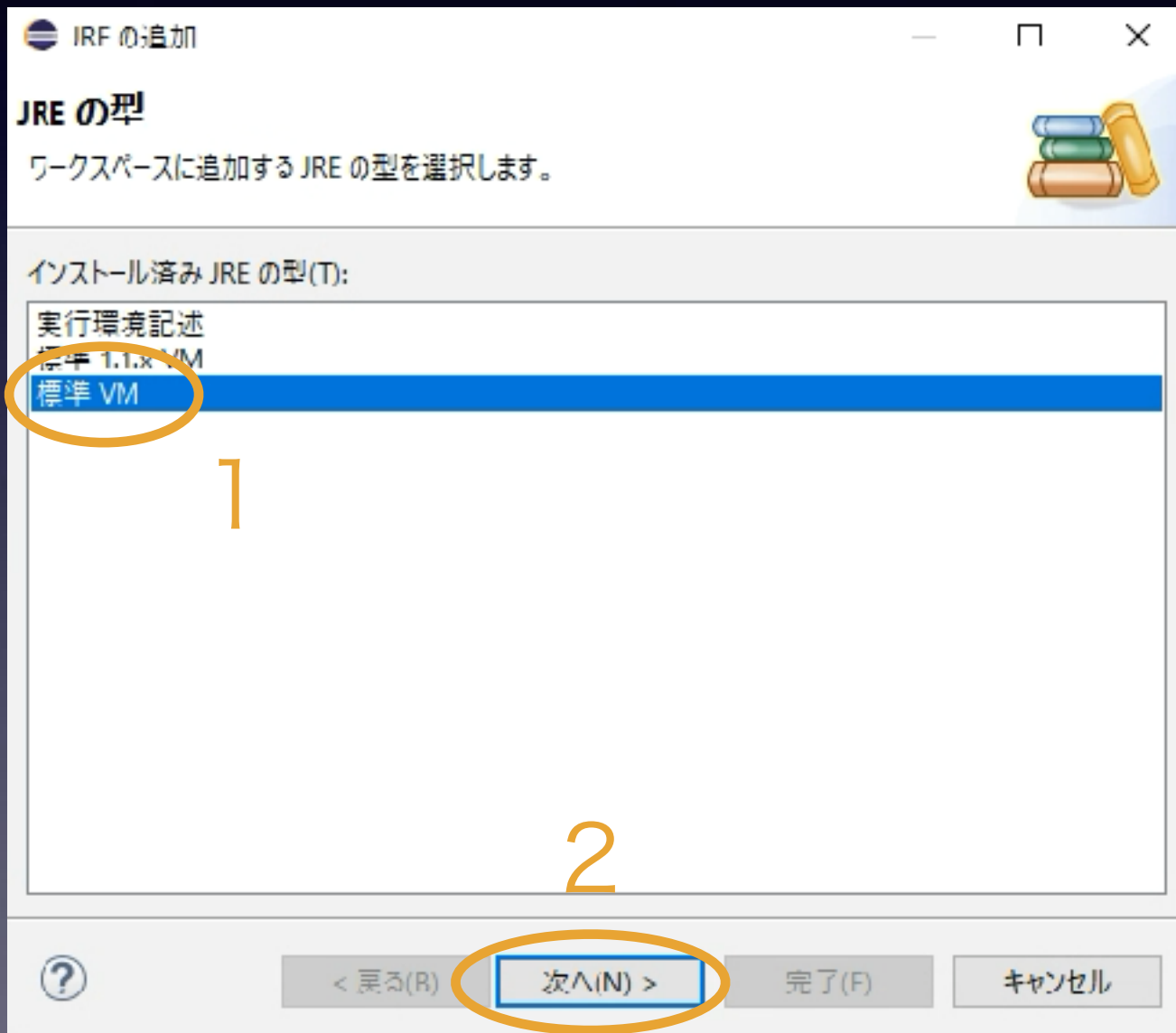


# 演習環境の利用 (7)



5

# 演習環境の利用 (8)





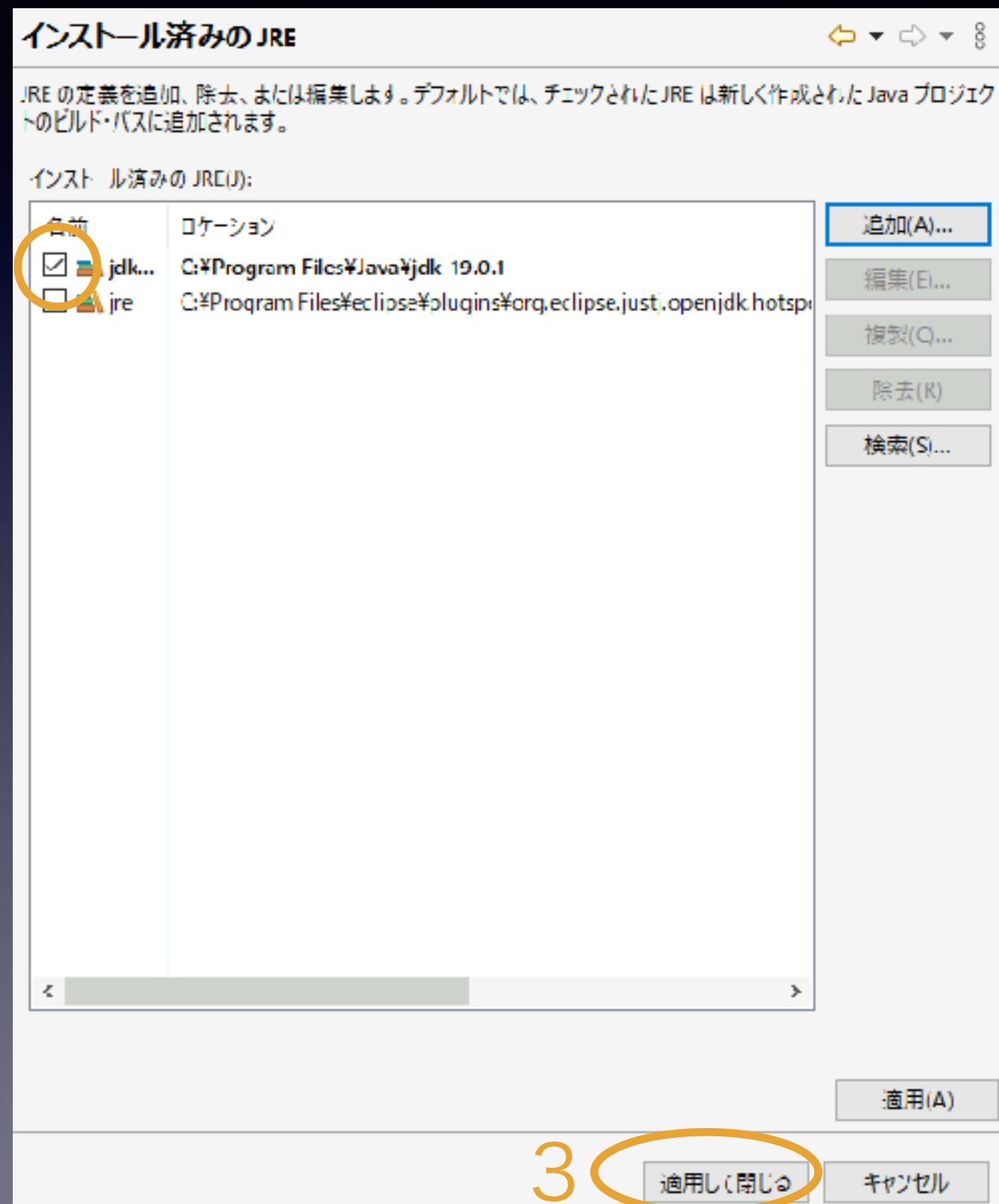
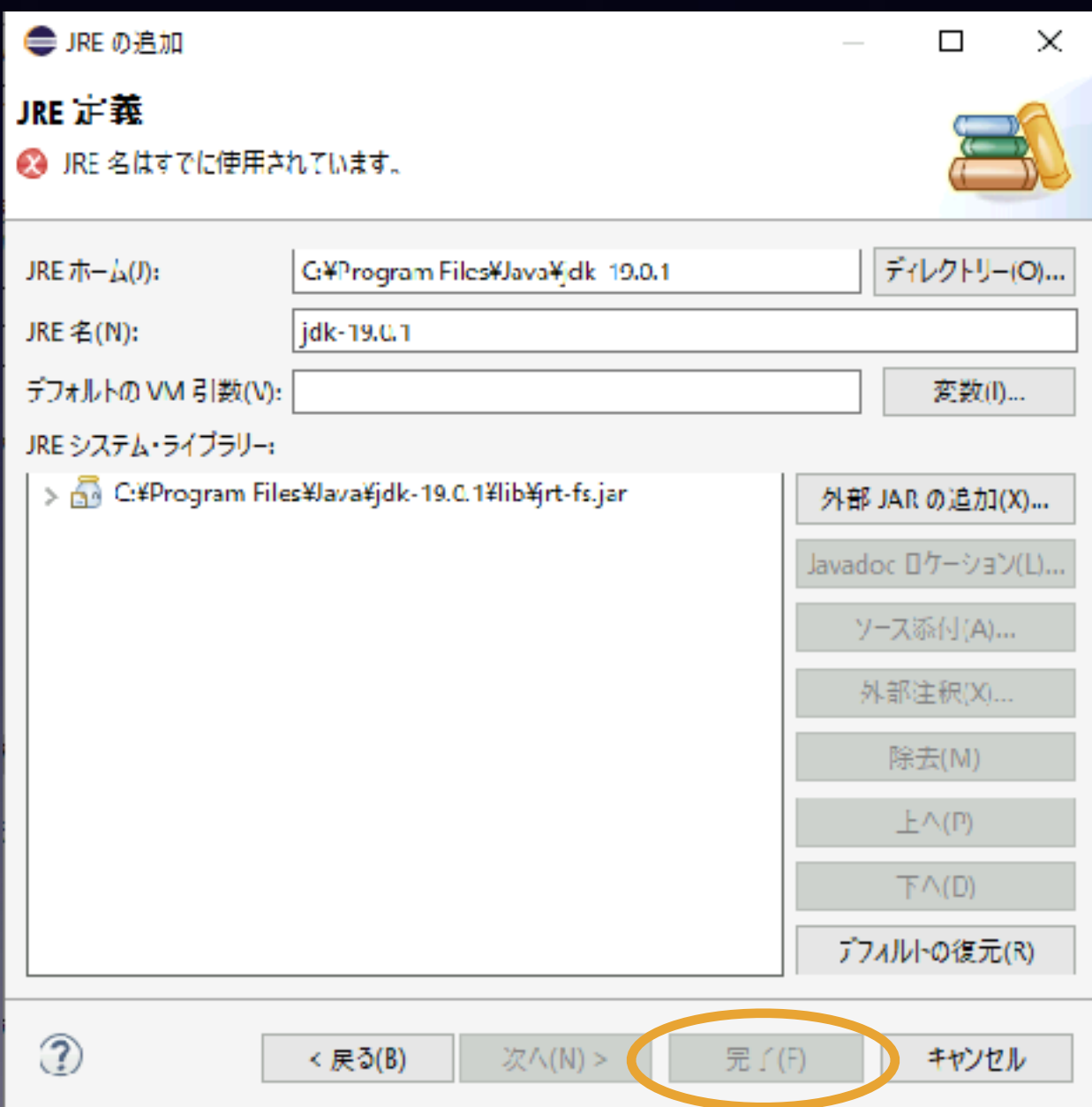
# 演習環境の利用 (9)



# 演習環境の利用 (10)

2

3

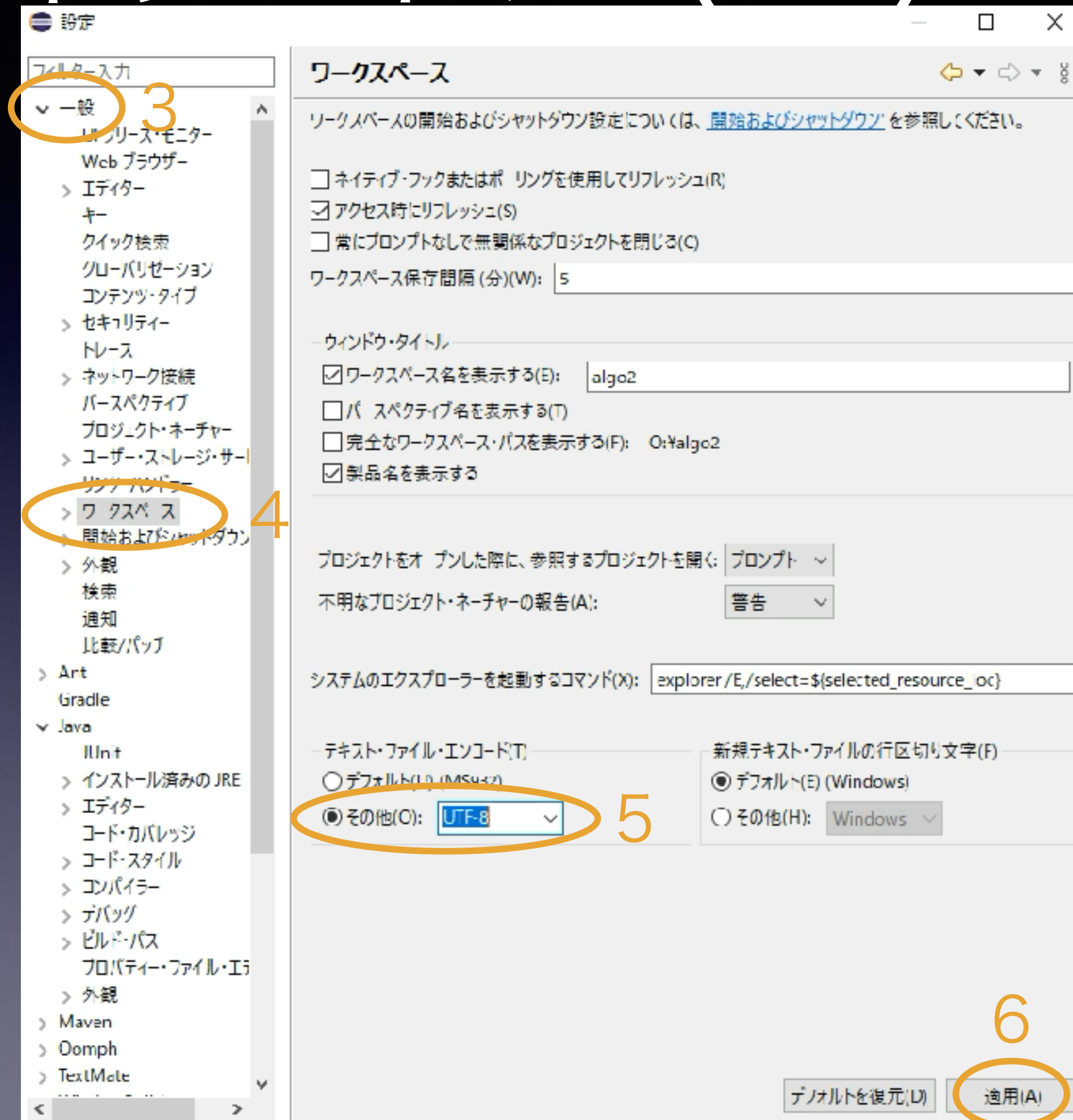


# 演習環境の利用 (11)

1



2

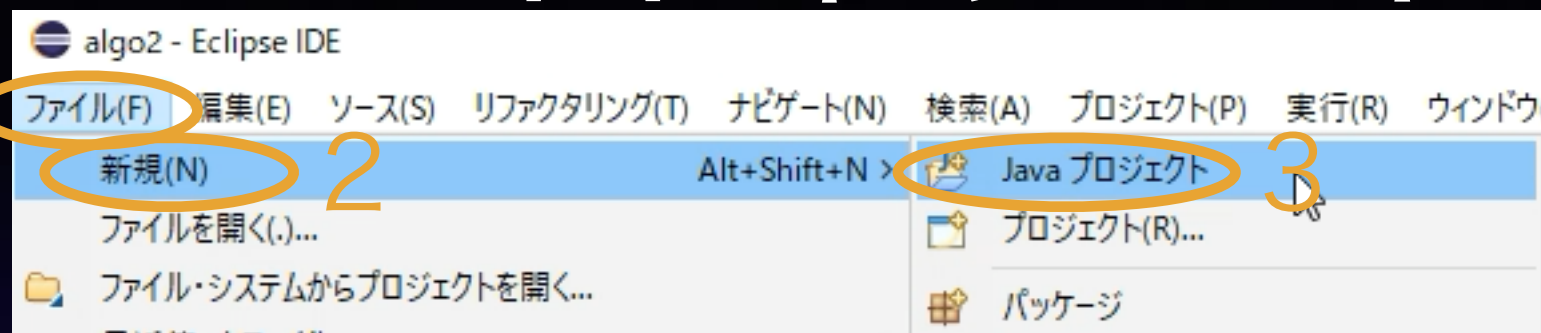


# 演習環境の利用 (12)

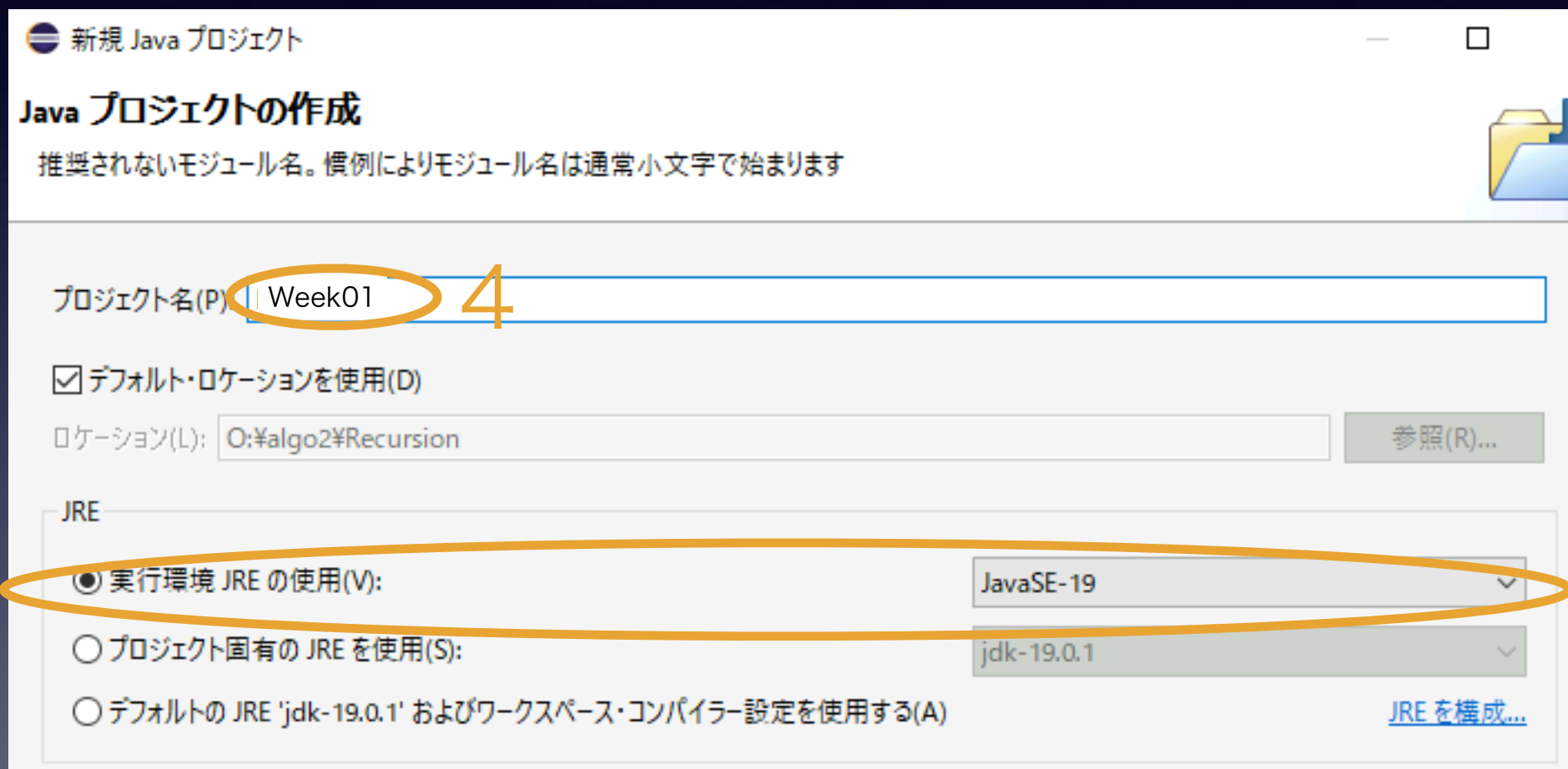
- ・ ここまでが初期設定 = 1回だけやればいい作業
- ・ 次回以降は「Eclipseの起動」と「ワークスペースの選択」だけやればok
- ・ ワークスペースの選択は、ログオンごとに必要
  - ・ やらないと、作成したデータが保存されない（ログオフすると消える）ので注意
- ・ フォルダ作成はもちろん今回だけの作業

# 演習環境の利用 (13)

1



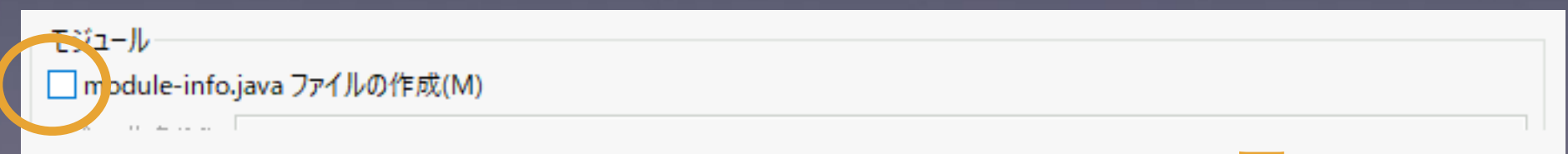
2



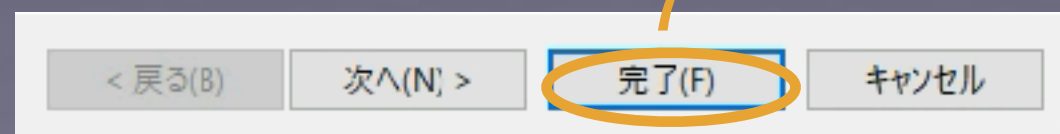
5

下にスクロール

6



7



# 演習環境の利用 (14)

- ・ srcを右クリック > New > Class
- ・ ここでは、クラス名は“HelloWorld” とする
- ・ main() にチェックを入れて、雛形を自動生成させる

Name: HelloWorld

Modifiers: ☒ public ☐ default ☐ private ☐ protected  
☐ abstract ☐ final ☒ static

Superclass: java.lang.Object Browse...


Interfaces: Add... Remove

Which method stubs would you like to create?

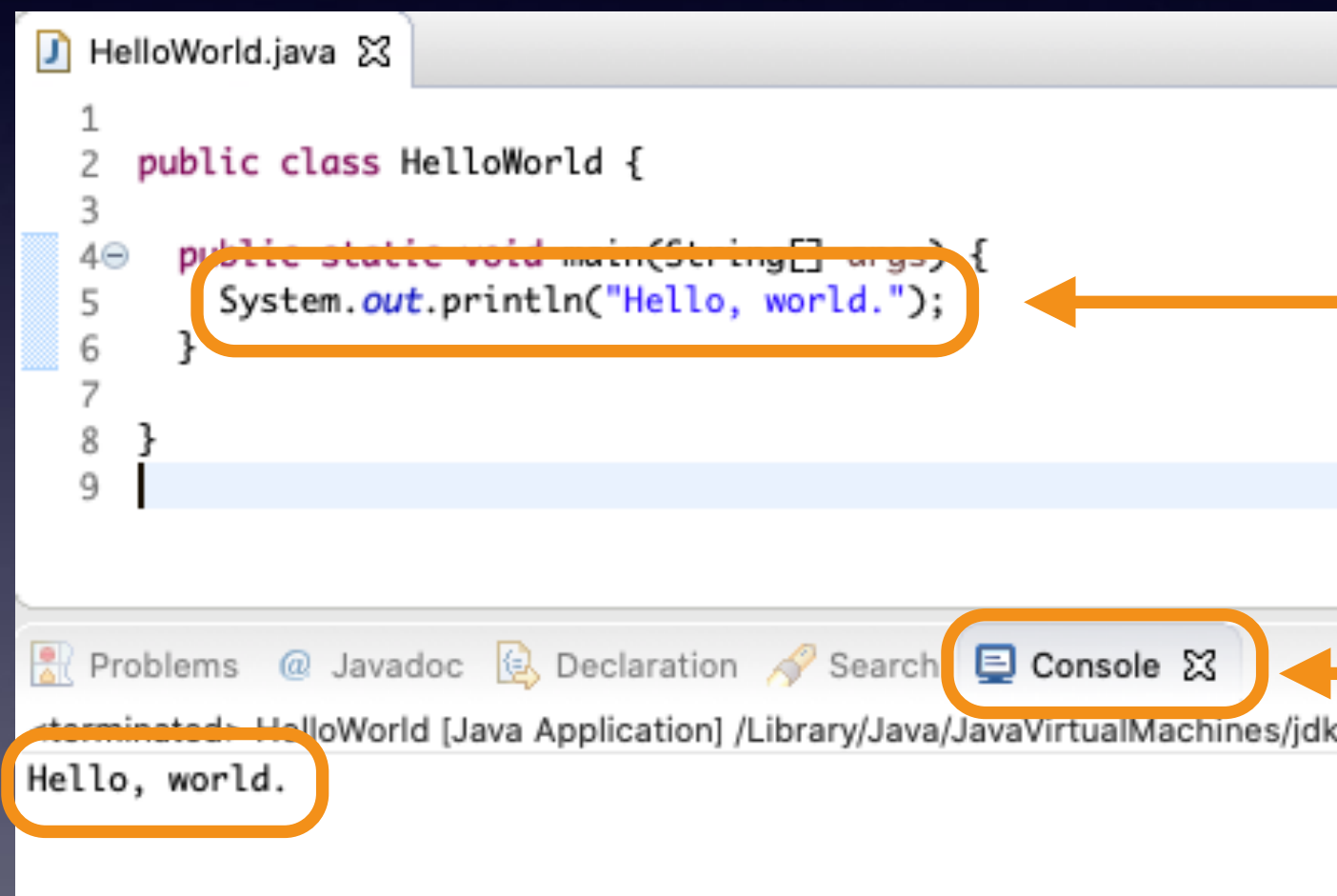
☒ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods



# 演習環境の利用 (15)

- ・ 雛形が自動生成され、ソースが表示される
- ・ `main()` の中身を作成する
  - ・ `System.out.println("Hello, world.");`
- ・ 実行ボタン  をクリック
  - ・ ファイルの保存を確認されたら OK を選ぶ
  - ・ “Always save resources before launching”  
をチェックしておくと、実行時に自動保存される
- ・ 実行結果は Console タブに表示される (次ページ)

# 演習環境の利用 (16)



The screenshot shows an IDE window with a tab for 'HelloWorld.java'. The code is as follows:

```
1  
2 public class HelloWorld {  
3  
4 public static void main(String[] args) {  
5     System.out.println("Hello, world.");  
6 }  
7  
8 }  
9 |
```

Below the code editor, there is a toolbar with buttons for 'Problems', 'Javadoc', 'Declaration', 'Search', and 'Console'. The 'Console' button is highlighted with an orange circle. Below the toolbar, the console output is displayed: 'Hello, world.', which is also highlighted with an orange circle.

この1行を入力  
// TODO: の行は削除

実行結果は  
Consoleタブに出る



# 演習環境の利用 (17)

- ・ Java Project は1週ごとに作成し、その週のサンプルや課題のクラスはその中に作成する
- ・ 本来、Java Project は開発するプログラムごとに作るのが普通

# Eclipseのワークスペース

- ・ ここまでやったら、Eclipseのワークスペースをエクスプローラで確認
  - ・ O:ドライブの eclipse フォルダを開く
- ・ 1つの Java Project が1つのフォルダに対応
- ・ src と bin という2つのフォルダがある
  - ・ ソースコード HelloWorld.java はsrcの中
    - ・ レポート課題等で提出するファイルはこちら
  - ・ クラスファイル HelloWorld.class はbinの中

# 変数

- ・ プログラムにおける記憶領域
- ・ 実体はハードウェア上（メモリ上）にある
- ・ プログラムで扱えるようにしたものが「変数」
- ・ 名前を付けて、データをしまっておける
- ・ 例：変数 age に 19 をしまう

# 基本データ型 (1)

- ・ 変数には型がある = 対応するデータだけしまえる
- ・ Javaには8つの基本データ型
- ・ 基本じゃないデータ型 = 参照型 (オブジェクト)
- ・ ここでは「授業で使う予定の型」だけを紹介する

# 基本データ型 (2)

- ・ int : 整数 (integer) を1つ記憶
  - ・ 32ビット符号付き整数
  - ・ -2,147,483,648~2,147,483,647
  - ・ Cとは異なり「32ビット」と決まっている

# 基本データ型 (3)

- ・ double : 有効数字約15桁の実数を1つ記憶
  - ・ 64ビット符号付き浮動小数点数
- ・ boolean : 真理値を1つ記憶
  - ・ 取り得る値は **true** か **false** のいずれか
  - ・ Cとは異なり、真理値を表す型がある
    - ・ `if(a = 0) ...` というバグを防ぐため

# 変数名の規則

- ・ アルファベットとアンダースコア "\_" が使える
- ・ 2文字目以降は数字も使える (a1とか)
- ・ 予約語は使用不可 (次のスライド参照)
- ・ 大文字・小文字は区別される (abcとAbc)
- ・ 名前付けのルール
  - ・ 用途が分かりやすい名前をつけること
  - ・ 業界のお約束がある (code conventions)
  - ・ 先頭は小文字
  - ・ 英単語を続けて書き、区切りは大文字にする

# Javaの予約語一覧

- Java Language Specificationより
  - <https://docs.oracle.com/javase/specs/jls/se19/html/index.html>

Keyword:

[ReservedKeyword](#)

[ContextualKeyword](#)

ReservedKeyword:

(one of)

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while
_ (underscore)				

ContextualKeyword:

(one of)

exports	opens	requires	uses
module	permits	sealed	var
non-sealed	provides	to	with
open	record	transitive	yield



# int型変数の使用例

- ・ Age.javaを以下の内容で Week01 内に作成せよ
- ・ 19の部分を実際の年齢に置き換えて実行せよ

```
public class Age {  
    public static void main(String[] args){  
        int age;  
        age = 19; // replace 19 with your age.  
        System.out.println("I'm " + age + " years old.");  
    }  
}
```

# Age.javaの補足 (1)

- ・ クラス名の命名規則
  - ・ 変数名と同じ、ただし大文字で始める
- ・ 変数は宣言してから使う： 型名 + 変数名
- ・ 文はセミコロンの ";" で終わる

```
public class Age {  
    public static void main(String[] args){  
        int age;  
        age = 19; // replace 19 to your age.  
        System.out.println("I'm " + age + " years old.");  
    }  
}
```

# Age.javaの補足 (2)

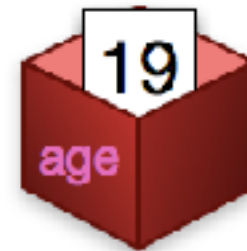
- ・ // から行末まではコメント（無視される）
- ・ Cと同様 /\* ... \*/ も使える
- ・ プログラムを読む人（自分も含む）のための説明を書く

```
public class Age {  
    public static void main(String[] args){  
        int age;  
        age = 19; // replace 19 with your age.  
        System.out.println("I'm " + age + " years old.");  
    }  
}
```

# Age.javaの補足 (3)

- ・ 等号 "=" は等式ではなく代入

```
int age;  
age = 19;
```



整数専用の箱

- ・ 右辺の値を左辺の中身にしまう (左右非対称)
- ・ 整数をしまうための箱 age の中に19という整数値が保存される

```
public class Age {  
    public static void main(String[] args){  
        int age;  
        age = 19; // replace 19 with your age.  
        System.out.println("I'm " + age + " years old.");  
    }  
}
```

# Age.javaの補足 (4)

- ・ 変数の値の表示も System.out.println() を使う
  - ・ 他のメッセージと続けて出すときは + でつなぐ
  - ・ 個別のメッセージはダブルクオート " で囲む
  - ・ Cと同じprintf()も使えるが、今は省略

```
public class Age {  
    public static void main(String[] args){  
        int age;  
        age = 19; // replace 19 with your age.  
        System.out.println("I'm " + age + " years old.");  
    }  
}
```

# double型変数の使用例

- ・ AgeDouble.javaを以下の内容で作成し、実行結果を確かめよ

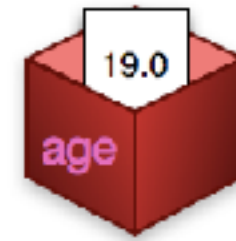
```
public class AgeDouble {  
    public static void main(String[] args){  
        double age;  
        age = 19; // replace 19 with your age.  
        System.out.println("I'm " + age + " years old.");  
    }  
}
```



# AgeDouble.javaの補足

- ・ ageの型をdoubleに変更した

```
double age;  
age = 19;
```



double専用の箱

- ・ 代入元は整数値だが、箱が浮動小数点数なので自動的に変換される
- ・ 逆もまた真（精度が落ちることに注意）

```
public class AgeDouble {  
    public static void main(String[] args){  
        double age;  
        age = 19; // replace 19 with your age.  
        System.out.println("I'm " + age + " years old.");  
    }  
}
```

# intとdoubleについて

- ・ intは32bit、 doubleは64bit
- ・ doubleの有効数字は15桁
  - ・ 不足の場合は近似値となる
- ・ intはdoubleで正確に表せる、逆は必ずしもできない

# 四則演算の使用例

- ・ AgeCalc.javaを以下のように作成せよ
- ・ yearOfBirthの値を各自で書き換えること

```
public class AgeCalc {  
    public static void main(String[] args){  
        int age, yearOfBirth, thisYear;  
        thisYear = 2023;  
        yearOfBirth = 2003; // replace this with yours.  
        age = thisYear - yearOfBirth;  
        System.out.println("I'm " + age + " years old.");  
    }  
}
```

# AgeCalc.javaの補足

- ・ 算術演算子
  - ・ 四則演算： $+$ ,  $-$ ,  $*$ ,  $/$
  - ・ 剰余： $\%$
  - ・ 整数演算では  $/$  は切り捨てになる
  - ・ 優先順位は数学と同じ

# 小レポート #1

- ・ 次頁に示す課題を行い、提出物 Arith.java を明星LMSに提出すること
- ・ 締切：9/22(金) 12:55
- ・ 提出の練習を兼ねているので、できる限りこの時間内に提出を

# 課題 (1)

- ・ 今日の資料で例示したプログラムを参考にして、以下のようなJavaプログラム Arith.java を作成せよ（クラス名は Arith とすること）
  - ・ 2つのint型変数 a, b を宣言
  - ・ 自分の学籍番号で - より前の数字3桁の各桁を足したものをaに代入、後半3桁の各桁を足したものをbに代入
    - ・ 例：17J5-789 => aに13を代入、bに24を代入
  - ・  $a+b$ ,  $a-b$ ,  $a*b$ ,  $a/b$  の結果を1行に1つずつ、合計4行で表示

# 課題 (2)

$$a + b = 24$$

$$a - b = -6$$

$$a * b = 135$$

$$a / b = 0$$

- ・ Consoleには上記のように出力せよ
- ・ a, b 以外の変数を使わずに実現した場合は加点として評価する



# 課題提出時の注意

- ・ 提出物は Arith.java ファイル（ソースファイル）
- ・ Arith.class はクラスファイル（バイトコード）
- ・ バイトコードだと採点できません → 0点