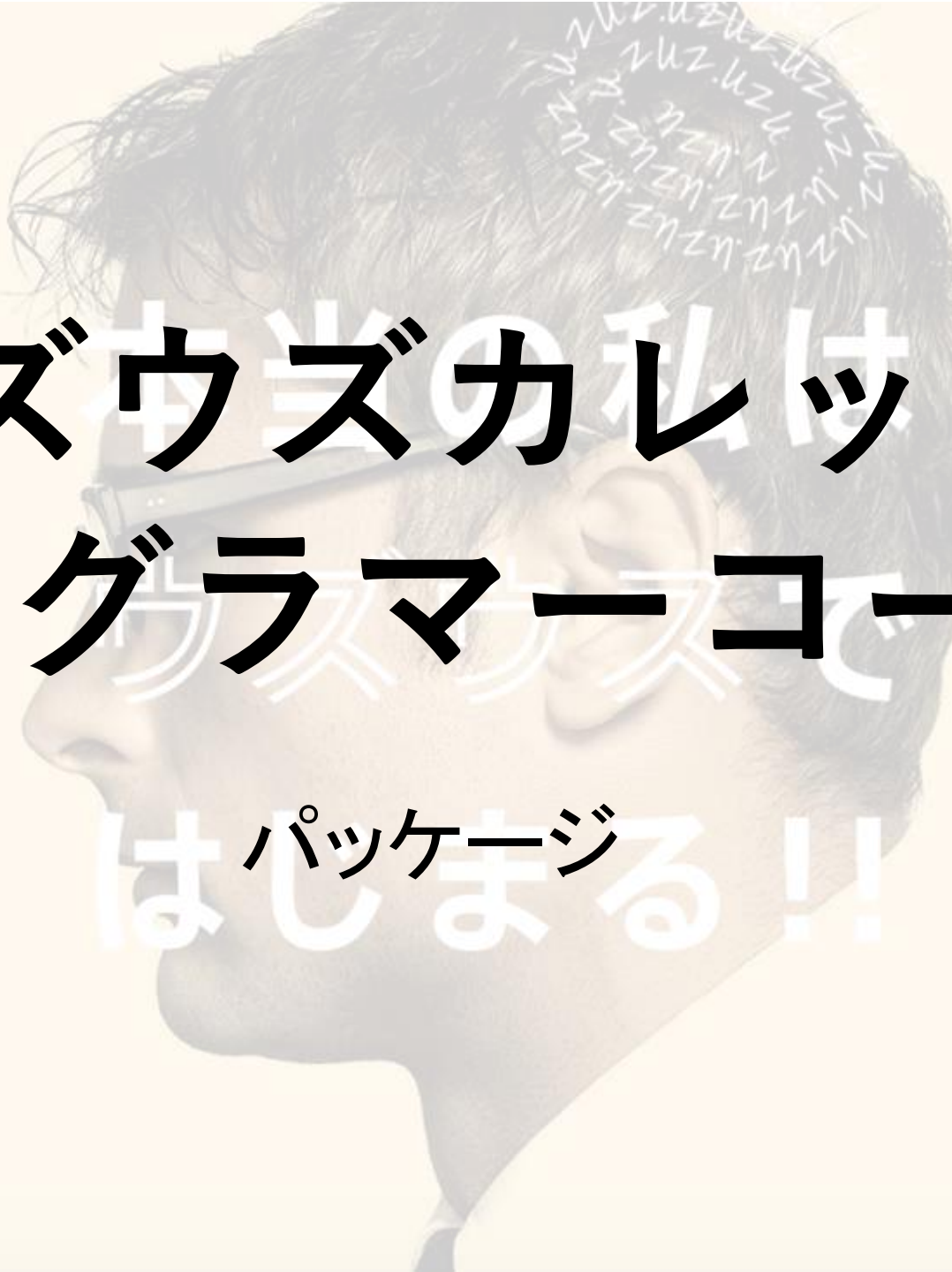
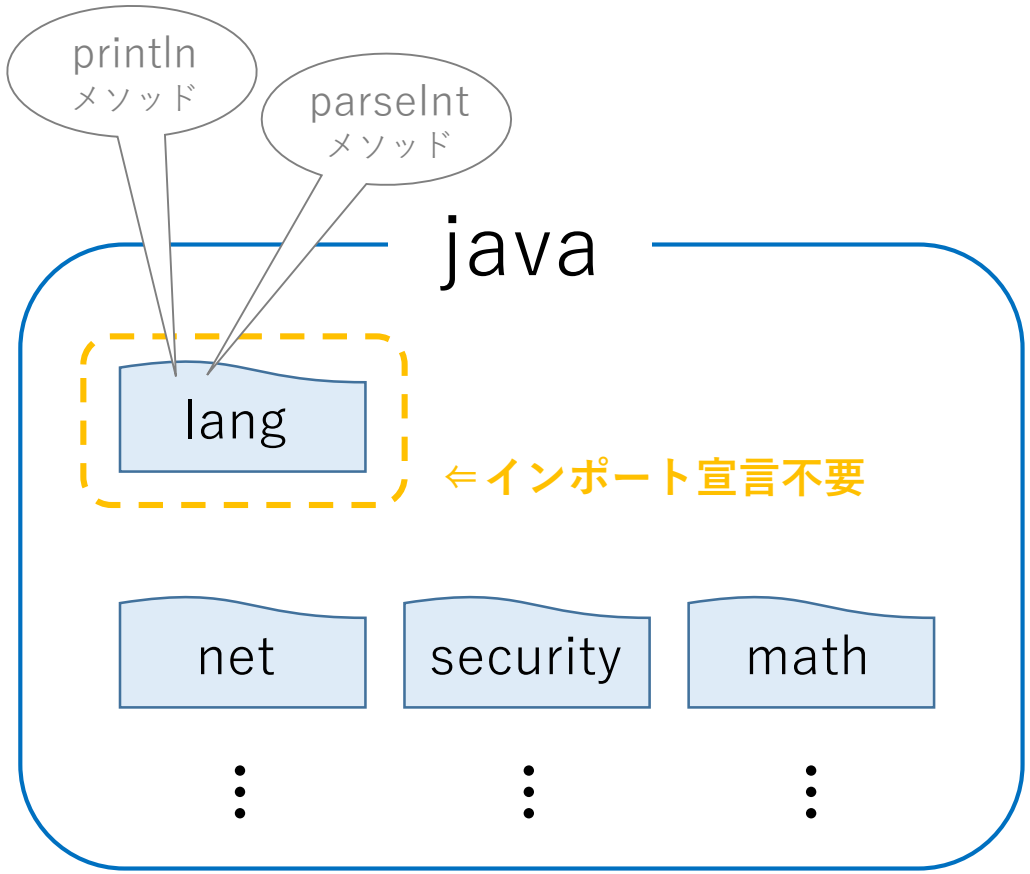


ウズウズカレッジ
プログラマーコース

パッケージ

復習



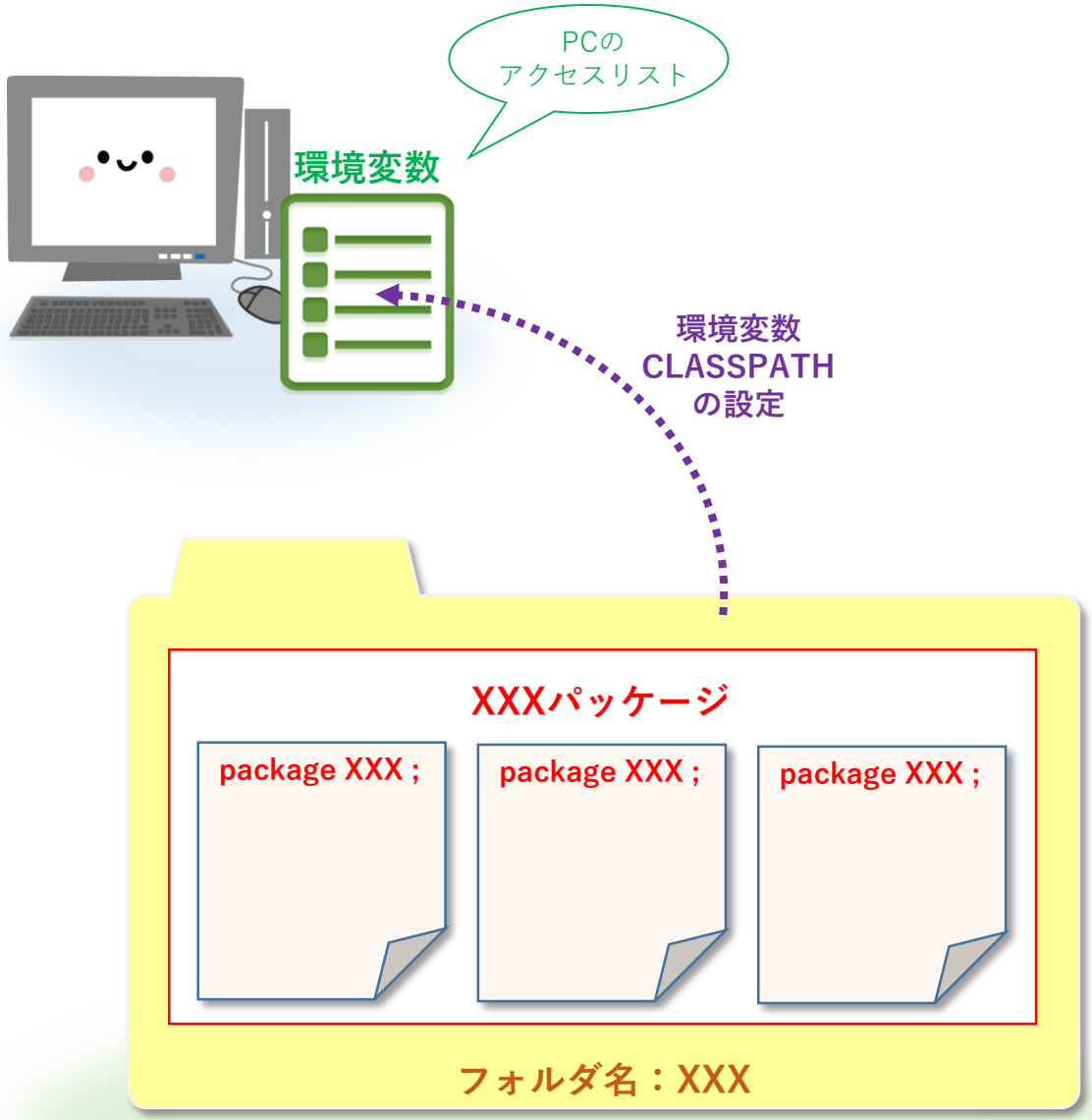
≪インポート≫

□APIは**パッケージ**と呼ばれるフォルダのようなもので管理されています。
まずすべてのAPIはjavaという名前のパッケージの中に入っており、その下で用途別のパッケージに分けられて管理されています。
例えばjava.securityパッケージにはセキュリティに関する機能を持ったクラス群が、java.netパッケージにはネットワークアプリケーションに関する機能を持ったクラス群が格納されています。
（「.」には「～の」という意味があります。）

□APIは基本的に使用したいクラスを**インポート**することで使用可能になります。インポートの方法は以下の通りです。

（クラスブロックの外側で）import パッケージ名.クラス名

□APIの中でも**java.langパッケージ**で管理されているクラス群は、その利用頻度があまりに高いため**インポート不要で使用可能**となっております。これまで使用してきたprintlnメソッド、parseIntメソッド、Stringメソッドはすべてjava.langパッケージに存在していたため当たり前のように使用することが可能となっておりました。



≪パッケージとは≫

□**パッケージ**とは**複数のクラスをグループ化**するためのものです。
実は**すべてのJavaプログラムは必ず何かしらのパッケージに属しています。**逆に**パッケージで管理されていないファイルはJavaのソースコードとして認められません。**

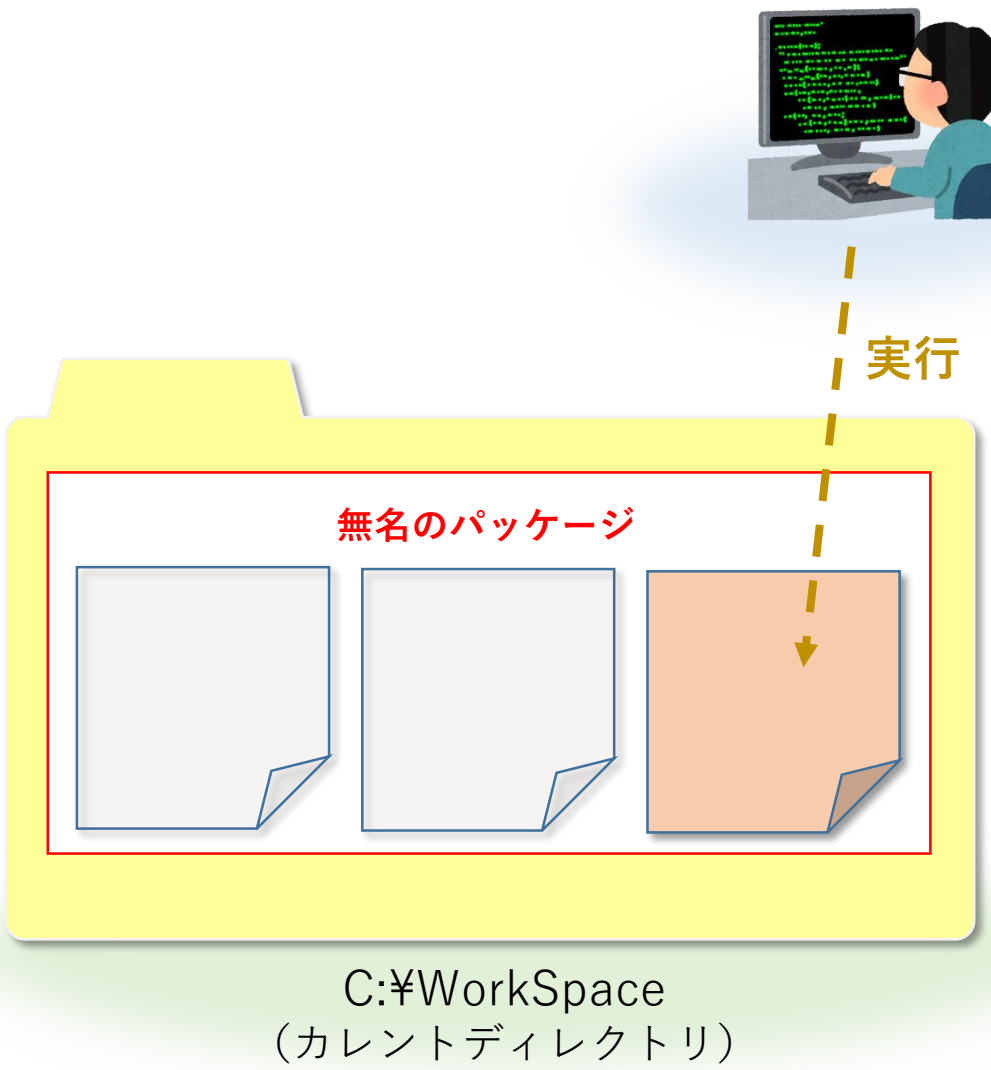
□ソースコードの先頭で**パッケージ宣言**することでそのクラスが所属するパッケージを定義することができます。なお、1つのクラスが所属できるパッケージは1つのみです。
パッケージ宣言は以下のように書きます。

package パッケージ名 ;

□パッケージは**必ずパッケージ名と同じ名前のフォルダで管理**されなければなりません。この仕様のため、パッケージ＝フォルダと捉えていただいても特に支障はありません。

□パッケージ宣言で指定するパッケージ名はCLASSPATHからの相対パスとなります。

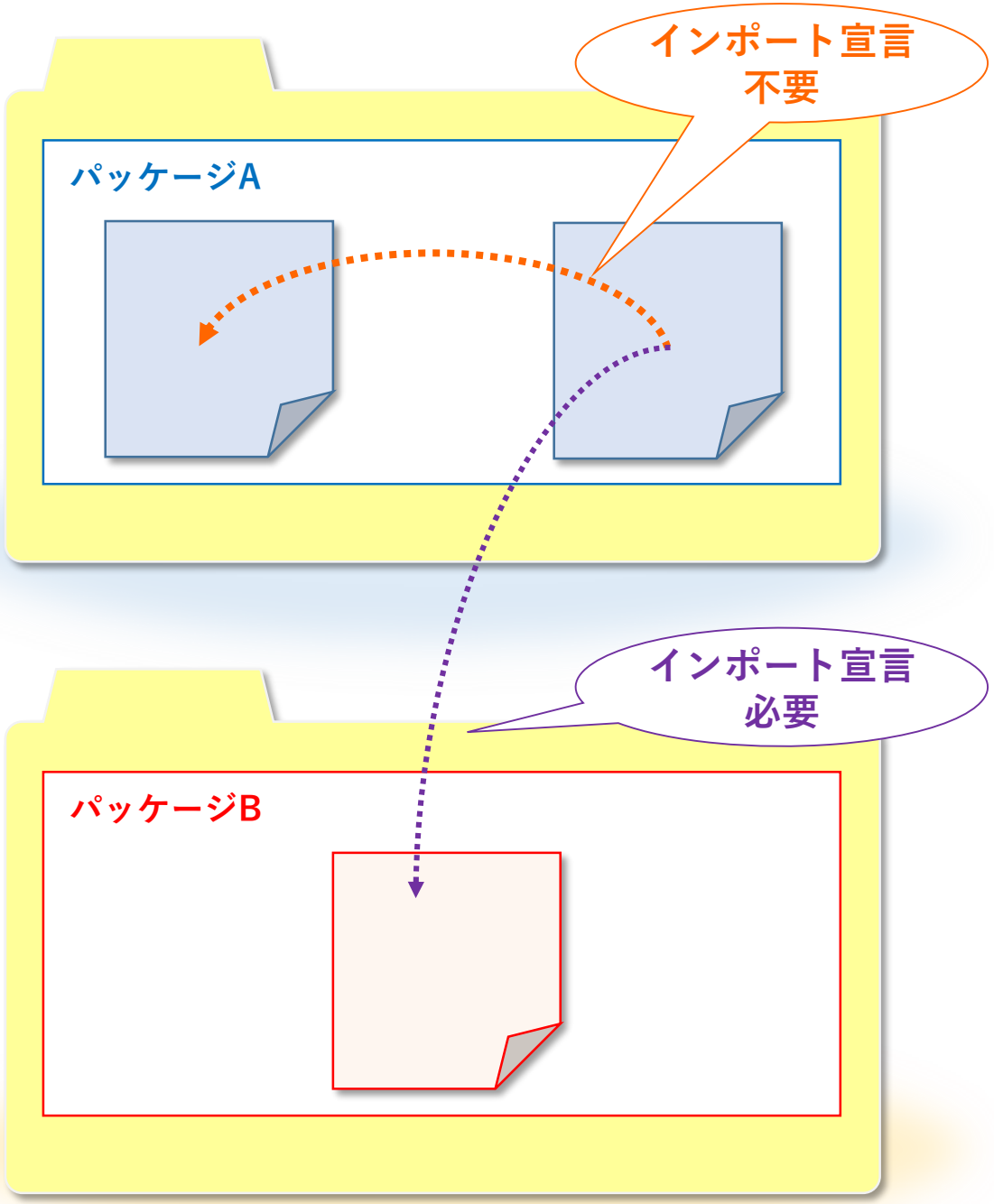
CLASSPATHは**環境変数**の1つで、 **Java実行時のアクセス先としてPC上におけるJavaのクラスやパッケージの格納場所を管理します。**
CLASSPATHの設定をしないと基本的にはJVMがアクセスできずにエラーとなります。



≪宣言不要のパッケージ≫

□**実行ファイルが存在するフォルダ（カレントディレクトリ）を自動的にパッケージとして扱われます。**これを**無名のパッケージ**といいます。

□無名のパッケージに格納されているファイルはすべて**package宣言が不要**です。

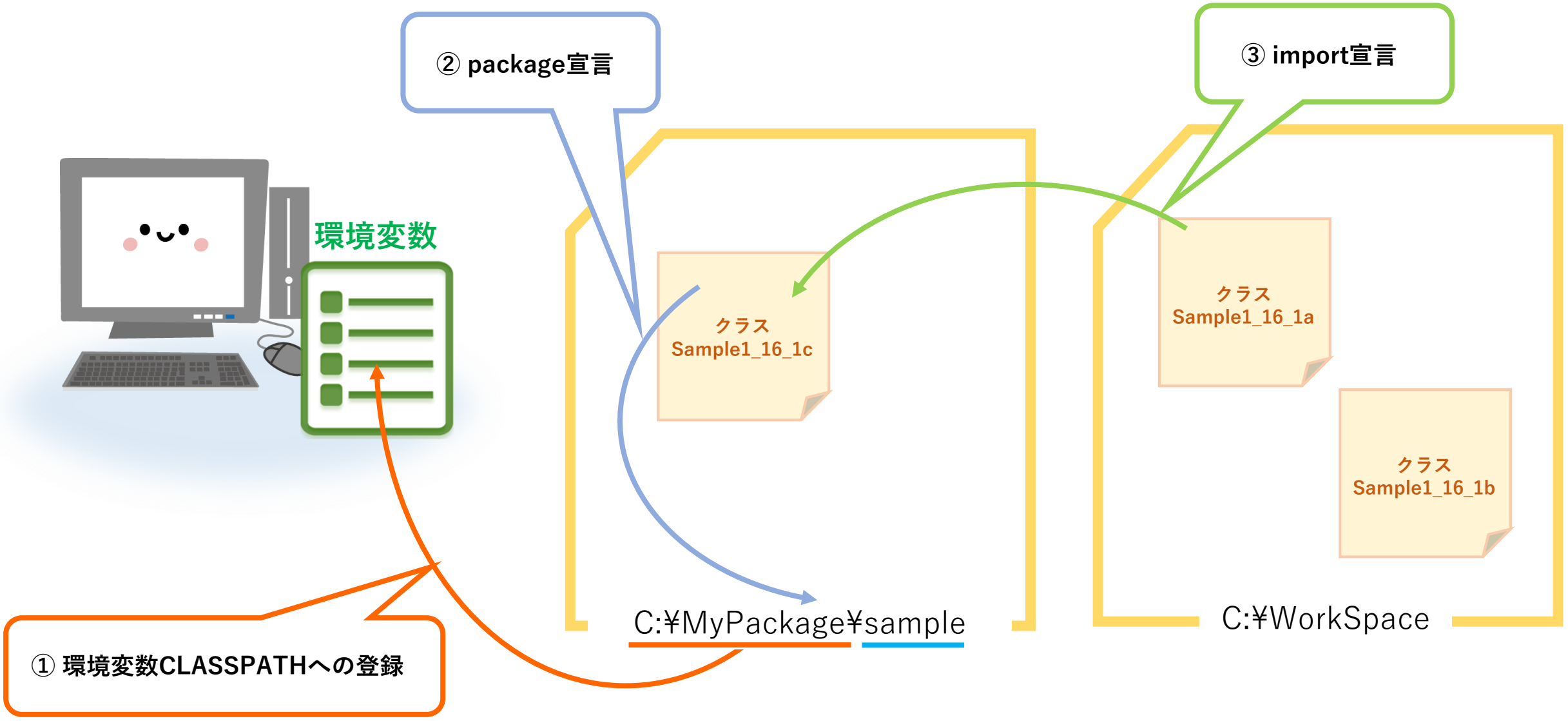


≪パッケージとインポートの要否≫

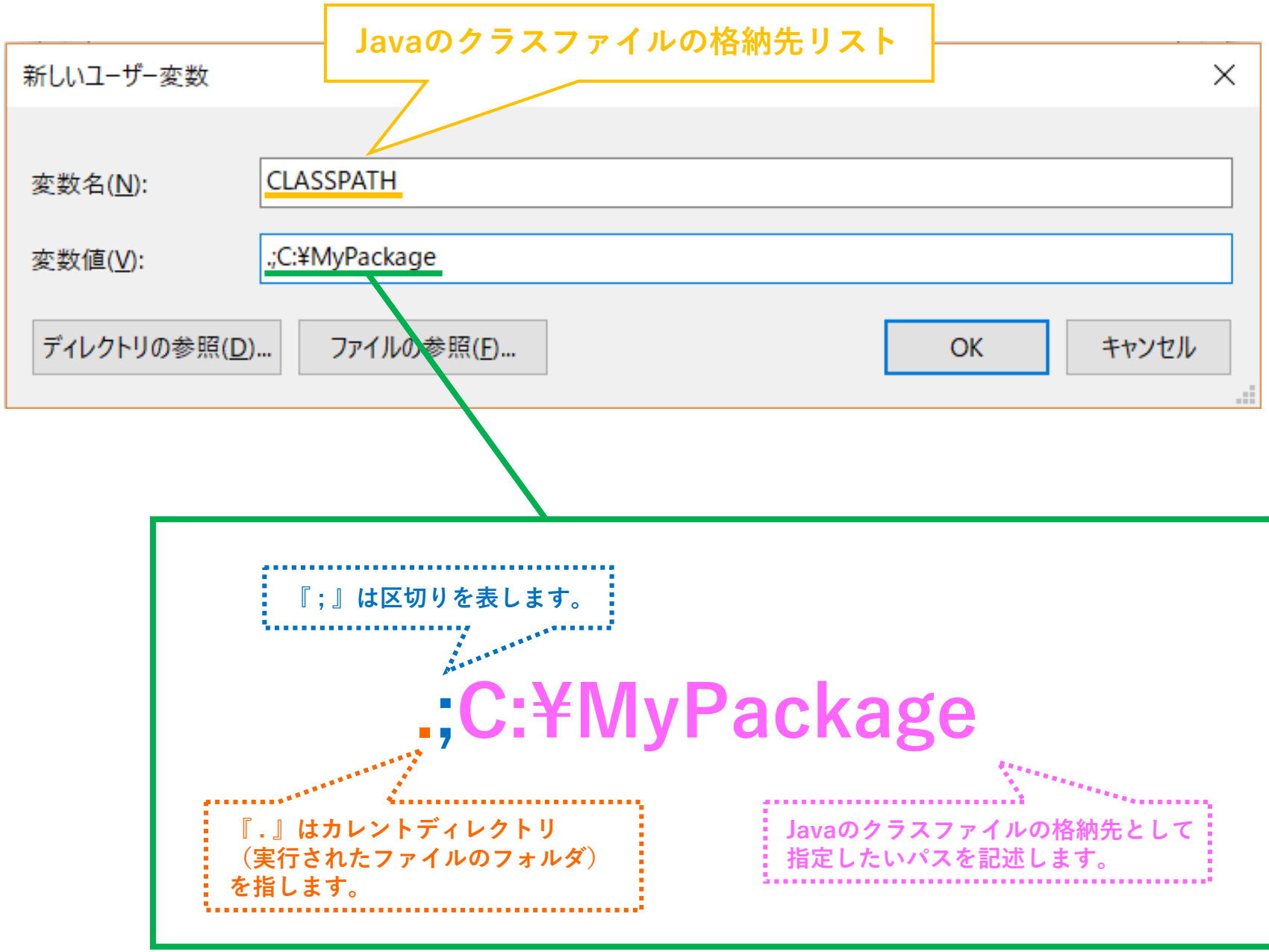
□同じパッケージ内であればインポート宣言なしで他のクラスを利用することができます。

□パッケージ外のクラスを利用する場合はインポート宣言が必要になります。（APIのjava.langパッケージを除く）

～Sample1_16_1aを動かそう！～



～Sample1_16_1aを動かそう！～



～Sample1_16_1aを動かそう！～

```java
import java.util.Calendar;     //APIからCalendarクラスをインポート
import sample.Sample1_16_1c;   //外部のパッケージsampleからinstanceSampleCクラスをインポート（CLASSPATHの設定が必要）

public class Sample1_16_1a {
    public static void main (String[] args) {

        //Calendarクラス（API）をインスタンス化（インスタンス名：instanceCal）
        Calendar instanceCal      = Calendar.getInstance();

        //Sample1_16_1bクラス（同じパッケージ）をインスタンス化（インスタンス名：instanceSampleB）
        Sample1_16_1b instanceSampleB = new Sample1_16_1b();

        //Sample1_16_1cクラス（外部のパッケージ）をインスタンス化（インスタンス名：instanceSampleC）
        Sample1_16_1c instanceSampleC = new Sample1_16_1c();

        /*
        //★インスタンスinstanceCalから現在の月日に関する情報を取得して画面表示する
        //  - instanceCal.get( Calendar.MONTH ) … インスタンスinstanceCalより現在の月に関する値※を取得します
        //      ※月の値は1月=0,2月=1,…,12月=11（0からカウント）で値を返します
        //  - instanceCal.get( Calendar.DATE )  … インスタンスinstanceCalより現在の日に関する値※を取得します
        //      ※日の値は1日=1,2日=2,…（1からカウント）で値を返します
        //  - instanceSampleB.realMonth( ～ )   … 引数に＋１した値を返します
        //  - instanceSampleC.makeMessage(〇,△)    … 文字列「今日は〇月△日です」を返します
        */
        int month = instanceSampleB.realMonth( instanceCal.get( Calendar.MONTH ) );     //現在の月の値を取得

        int date   = instanceCal.get( Calendar.DATE );                                   //現在の日の値を取得

        System.out.println( instanceSampleC.makeMessage( month , date ) );               //画面に「今日は〇月△日です」と表示
    }
}
```

C:¥WorkSpace

```java
//mainメソッドを持たないため、別のクラスからインスタンス化されて使用される

class Sample1_16_1b {

    int realMonth( int m ){
        return m + 1;                  //引数の値に+1した値を返す
    }

}
```

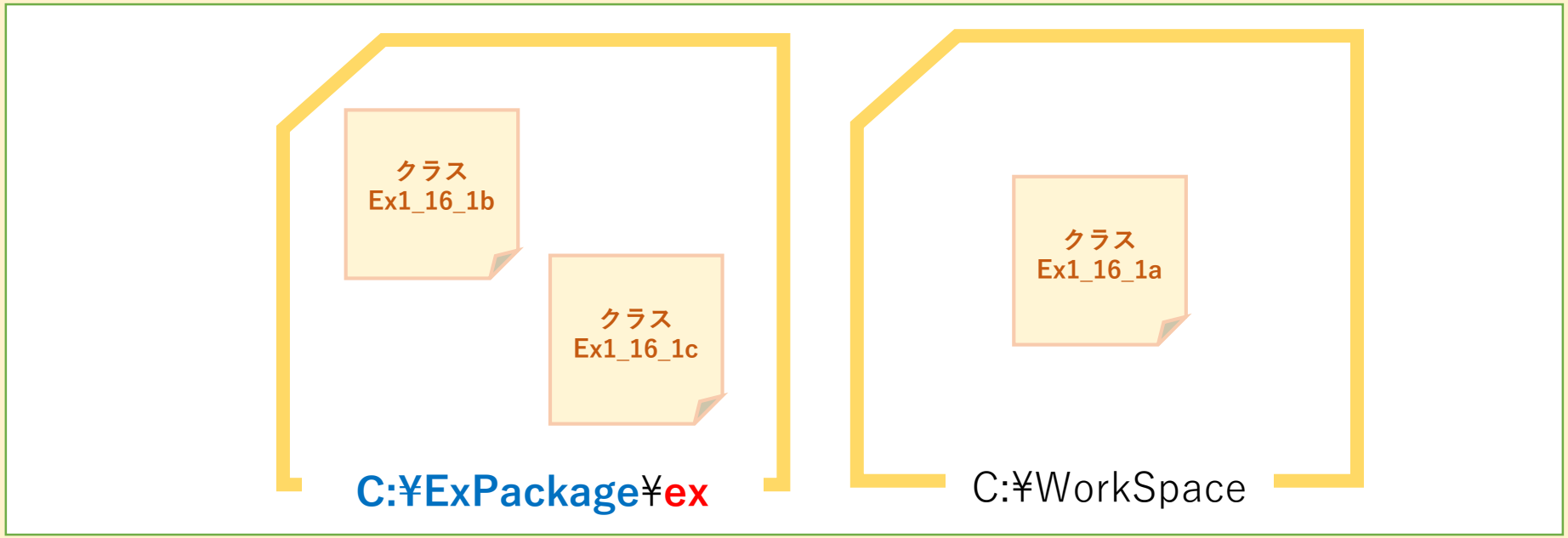
C:¥MyPackage¥sample

```java
//mainメソッドを持たないため、別のクラスからインスタンス化されて使用される

package sample;      //CLASSPATHに『.;C:\MyPackage』の設定が必要

public class Sample1_16_1c {

    public String makeMessage ( int mon , int date ){
        String message  = "今日は" + mon + "月" + date + "日です" ;
        return message ;
    }

}
```

＜演習：Ex1_16_1＞

以下のようにファイルを配置し、動作するようにプログラムを書き換えたい。

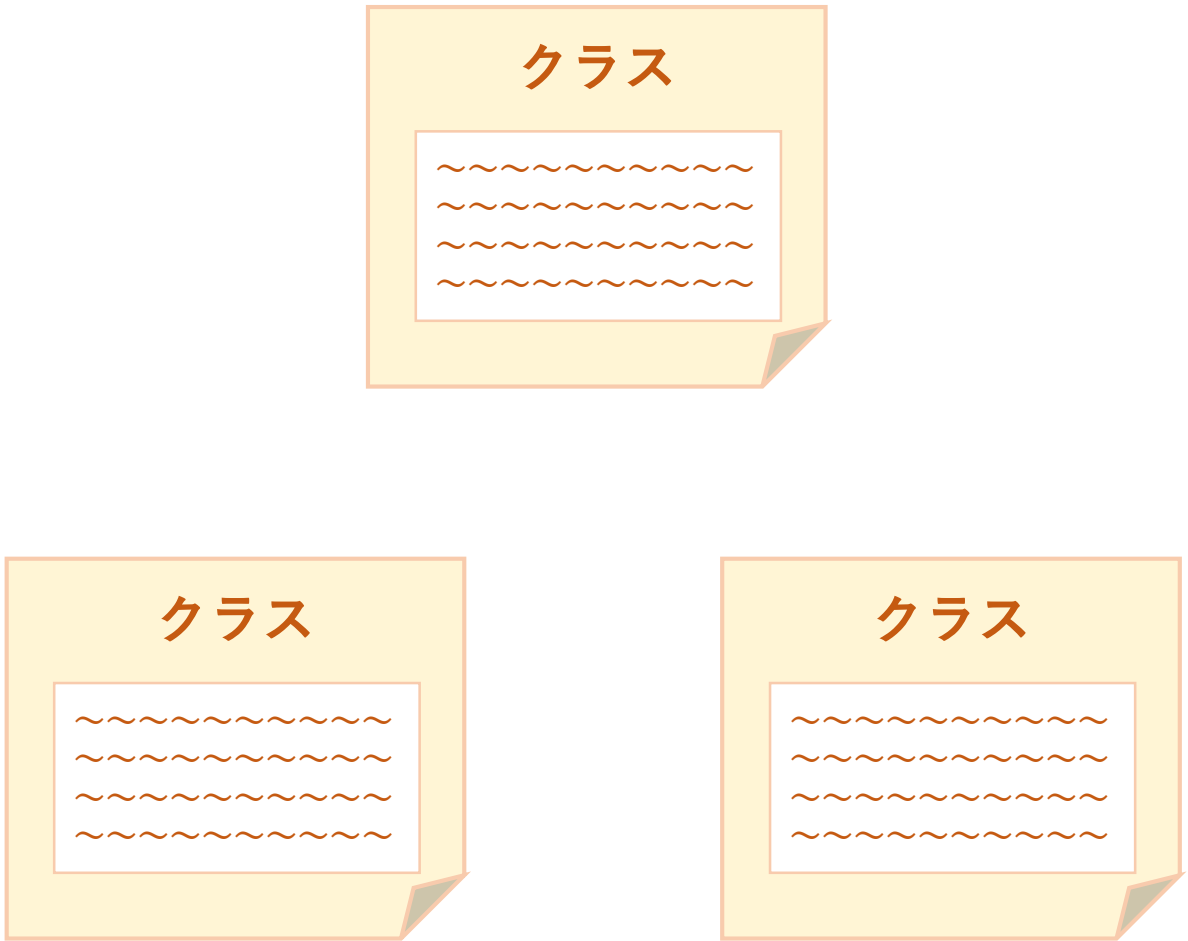
⑴ 「C:¥ExPackage」を環境変数CLASSPATHに追加登録してください。
　　（既に入力済の「.;C:¥MyPackage」は消さずに付け足しましょう！）
⑵ Ex1_16_1a.java に適切なimport文を記述してください。
⑶ Ex1_16_1b.java に適切なパッケージ宣言を記述してください。
⑷ Ex1_16_1c.java に適切なパッケージ宣言を記述してください。

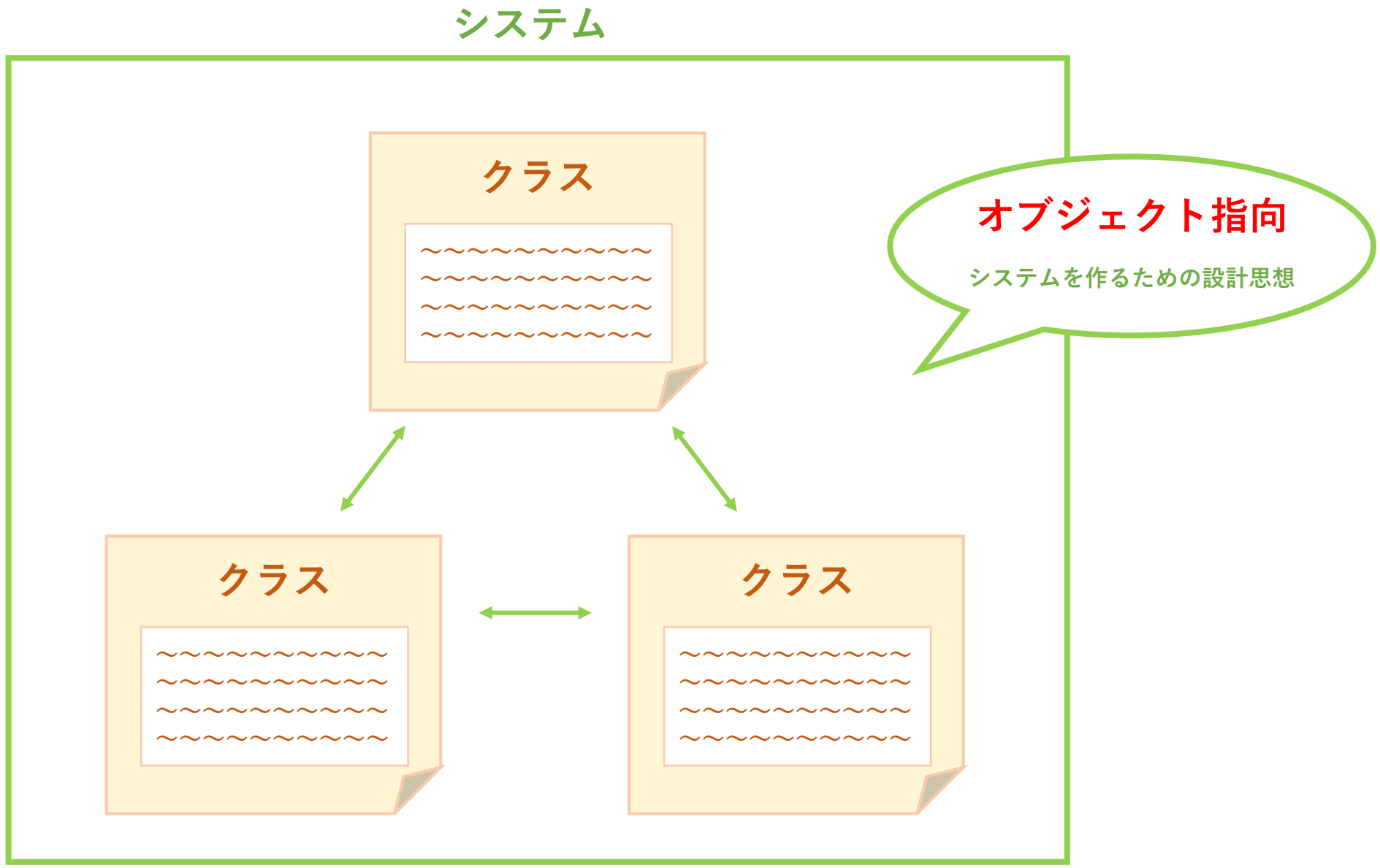



パッケージ・エクスプローラー
JavaPractice
JRE システム・ライブラリー [JavaSE-1
src
(デフォルト・パッケージ)
Ex1_16_1a.java
ex
Ex1_16_1b.java
Ex1_16_1c.java

Javaプログラミング講座～基礎固め編～

修了


クラス
クラス
クラス


システム
クラス
クラス
クラス
オブジェクト指向
システムを作るための設計思想