

● J2Kad19D「内部クラス」(実践編 P.125「内部クラス」)

リスト1を参考に SayHello インターフェイス、Greeting クラス、OuterPerson クラスを作成し、main メソッドに以下の処理を作成せよ。

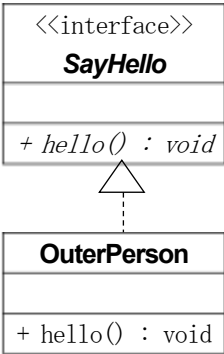
- ① SayHello インターフェイスの参照を使って OuterPerson を生成、Greeting.greet メソッドであいさつさせる。
- ② main メソッド内に内部クラスとして InnerPerson (仕様は OuterPerson と同じ、ただし hello メソッドの「外部クラス」を「内部クラス」に変更すること) を作成し、①と同じようにあいさつさせる。

Greeting クラスの仕様 (J2Kad19D.java に作成)

メソッド	説明
public static void greet(SayHello s)	s の hello メソッドを呼び出す。

OuterPerson クラスの仕様 (J2Kad19D.java に作成)

メソッド	説明
public void hello(SayHello s)	「外部クラス：こんにちは！」と表示する。



リスト1：外部クラスと内部クラス (ファイル「J2Kad19D.java」)

```
public class J2Kad19D {
    public static void main(String[] args) {
        // OuterPerson (外部クラス)
        OuterPerson を生成して Greeting.greet メソッドへ渡す。
        // InnerPerson (内部クラス)
        InnerPerson を定義する。
        InnerPerson を生成して Greeting.greet メソッドへ渡す。
    }
}
```

課題完成時の画面

外部クラス：こんにちは！
内部クラス：こんにちは！

● J2Kad19C 「匿名クラス（無名クラス）」（実践編 P.127 「匿名クラス」）

J2Kad19D で作成した SayHello インターフェイスを使って、以下の処理を作成せよ。

- ① SayHello インターフェイスの参照に匿名クラスを定義して設定、Greeting.greet メソッドにこの参照を渡す。
- ② Greeting.greet メソッドの引数として直接、匿名クラスを定義して渡す。

なお、①の匿名クラスでは「匿名クラス①：こんにちは！」、②の匿名クラスでは「匿名クラス②：こんにちは！」と表示すること。

課題完成時の画面

匿名クラス①：こんにちは！
匿名クラス②：こんにちは！

● J2Kad19B 「ラムダ式」（実践編 P.131 「関数型インタフェース」、P.132 「ラムダ式」）

J2Kad19C の匿名クラスをラムダ式で記述せよ。なお、「匿名クラス」の表示は「ラムダ式」に変更すること。

課題完成時の画面

ラムダ式①：こんにちは！
ラムダ式②：こんにちは！

● J2Kad19A 「ラムダ式の省略形」 (実践編 P.134 「ラムダ式の省略形」)

リスト1 のコード (実践編 P.136、List⑥-5) を入力し、以下の仕様で main メソッドから printout メソッドを呼び出す処理を作成せよ。printout メソッドにはラムダ式を渡すものとし、ラムダ式として実装するコードは「return n + 1;」とする。

- ① printout メソッドの引数にラムダ式 (省略なし) を渡す。
- ② ラムダ式の引数の型を省略して渡す。
- ③ さらにラムダ式の引数を囲むカッコを省略して渡す。
- ④ さらに命令文の中カッコとセミコロンを省略して渡す。

リスト1: ラムダ式の省略形 (ファイル「J2Kad19A.java」)

```
interface SimpleInterface {  
    int doSomething(int n);  
}  
  
public class J2Kad19A {  
    static void printout(SimpleInterface i) {  
        System.out.println(i.doSomething(2));  
    }  
  
    public static void main(String[] args) {  
        printout(①もとのラムダ式);  
        printout(②引数の型を省略);  
        printout(③引数を囲む()を省略);  
        printout(④命令文の {} と; を省略);  
    }  
}
```

課題完成時の画面

```
3  
3  
3  
3
```

「3」が4つ並ぶ
(①から④まで処理自体は同じ)

● J2Kad19S「ライフゲーム①（初期データの表示）」

ライフゲーム（J2Kad19X で作成）の初期データがテキストファイルとして準備されている。initCanvas メソッドを作成し、読み込んだデータを Canvas クラスで表示せよ。

初期データ

ファイル名	描画座標	説明
./data/p00.txt	X : 10、Y : 5	パルサー。周期 3 で変化する。
./data/p01.txt	X : 20、Y : 15	グライダー。左上へ移動していく。
./data/p02.txt	X : 20、Y : 4	ダイハード。130 世代後に死滅する。

initCanvas メソッドの仕様（作成すること）

書式	仕様
public static void initCanvas(Canvas c)	読み込むデータファイル、X 座標、Y 座標をキーボードから入力、ファイルから初期データを読み込み Canvas クラスに設定する。 文字が*のときは true、それ以外の場合は false を設定する。

Canvas クラスの仕様（作成済み）

書式	仕様
public Canvas(int width, int height)	コンストラクタ。横幅 : width、高さ : height のキャンバスを作る。
public void show()	キャンバスを画面に表示する。
public setPoint(int x, int y, boolean dot)	座標(x, y)を dot に設定する。
public boolean getPoint(int x, int y)	座標(x, y)の値を返す。キャンバス外の場合は false を返す。

リスト 1：初期データの表示（ファイル「J2Kad19S.java」）

```
public class J2Kad19S {
    public static final int WIDTH = 40;
    public static final int HEIGHT = 24;

    public static void main(String[] args) {
        Canvas c = new Canvas(WIDTH, HEIGHT);
        initCanvas(c);
        c.show();
    }
    public static void initCanvas(Canvas c) {
        作成すること
    }
}
```


課題完成時の画面 (p00.txt の場合)

読み込むデータファイル名>p00

X 座標>10

Y 座標>5

(J2Kad20S と同じ)

[0] どうしますか？ (0 : 続ける、-1 : 終了) >0

マイナスの値で終了、それ以外は次のパターンを表示する。

入力の際は先頭にこれまでの
入力回数（世代数）も表示する。

[1] どうしますか？ (0 : 続ける、-1 : 終了) >0

[2] どうしますか？ (0 : 続ける、-1 : 終了) >0

(最初のパターンに戻る)

[3] どうしますか？ (0: 続ける、-1: 終了) >-1

p00 の場合、周期 3 で元のパターンに戻る。

p01 の場合、左上へ向かってパターンが動いていく。

p02 の場合、130 回目で全滅する。