# **J2Kad20D「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クラスを追加し、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「外部クラス」→「内部クラス」にすること）を作成し、①と同じようにあいさつさせる。

|  |
| --- |
| <<interface>>  ***SayHello*** |
|  |
| *+ hello() : void* |

|  |
| --- |
| **OuterPerson** |
|  |
| + hello() : void |

**Greetingクラスの仕様（J2Kad20D.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（J2Kad20D.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「J2Kad20D.java」）**

public class J2Kad20D {

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！

# **J2Kad20C「匿名クラス（無名クラス）」**

J2Kad20D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！

# **J2Kad20B「ラムダ式」※J2Kad20Cのmainメソッドをコピーして作成**

J2Kad20C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！

# **J2Kad20A「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「J2Kad20A.java」）**

interface SimpleInterface {

int doSomthing(int n);

}

public class J2Kad20A {

static void printout(SimpleInterface i) {

System.out.println(i.doSomthing(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」が4つ並ぶ

1. から④まで処理自体は同じ）

3

3

3

3

# **J2Kad20S「ライフゲーム①（初期データの表示）」**

ライフゲーム（J2Kad20X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**

|  |  |
| --- | --- |
| 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、それ以外のときはfalseを設定する。 |

**Canvasクラスの仕様**

|  |  |
| --- | --- |
| 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。キャンバス外のときはfalseを返す。 |

**リスト1：初期データの表示（ファイル「J2Kad20S.java」）**

public class J2Kad20S {

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）**

指定した場所にテキストファイルと同じパターンが表示されていたらOK

読み込むデータファイル名＞**p00**

X座標＞**10**

Y座標＞**5**

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・○○○・・・○○○・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・○・・・・○・○・・・・○・・・・・・・・・・・・・・・・

・・・・・・・・・・・○・・・・○・○・・・・○・・・・・・・・・・・・・・・・

・・・・・・・・・・・○・・・・○・○・・・・○・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・○○○・・・○○○・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・○○○・・・○○○・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・○・・・・○・○・・・・○・・・・・・・・・・・・・・・・

・・・・・・・・・・・○・・・・○・○・・・・○・・・・・・・・・・・・・・・・

・・・・・・・・・・・○・・・・○・○・・・・○・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・○○○・・・○○○・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

# **ライフゲーム（←検索）**

自分を囲む8つのセルの状態によって、そのセルに生命が誕生するのか死滅するのかが決まるシミュレーション。状態変化（生or死）のルールは以下の通り。

**状態変化のルール**

|  |  |  |
| --- | --- | --- |
| 着目するセルの状態 | 周囲8マスのセルの生命の数 | 着目するセルの次の状態 |
| 生 | 2または3 | 生存（生） |
| 1以下 | 過疎（死） |
| 4以上 | 過密（死） |
| 死 | 3 | 誕生（生） |
| それ以外 | 死のまま |

# **J2Kad20X「ライフゲーム②（実行！）」**

**課題完成時の画面**を参考にライフゲームを作成せよ。なお、コーディングは各自で考えること（J2Kad20SのinitCanvasメソッドの利用OK）。

**課題完成時の画面（p00.txtの場合）**

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。

入力のときは先頭にこれまでの

入力回数（世代数）も表示する。

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**

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・○・・・・・○・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・○・・・・・○・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・○○・・・○○・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・○○○・・○○・○○・・○○○・・・・・・・・・・・・・・・

・・・・・・・・・・・・○・○・○・○・○・○・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・○○・・・○○・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・○○・・・○○・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・○・○・○・○・○・○・・・・・・・・・・・・・・・・・

・・・・・・・・・・○○○・・○○・○○・・○○○・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・○○・・・○○・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・○・・・・・○・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・○・・・・・○・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

[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。