## ● 授業のポイント

① 内部クラスについて学習します。
② 匿名クラス（無名クラス）について学習します。
③ ラムダ式について学習します。

**※ 今回は課題 A までがラムダ式の練習になります。**
**※ （いつもとは難易度が異なります）物足りない学生はＳやＸを積極的にチャレンジしましょう！**
**※ ラムダ式自体は「こういう書き方がある」ということを頭の片隅に覚えておいてもらうレベルで構いません。**
**※ 実際にこういうコードに直面したときに「あ、授業できいたことがある」→「調べてみよう」となればいいです!**

## ● 学習項目

・「内部クラスとは（実践編 P. 125)」
・「匿名クラス（実践編 P. 127)」
・「関数型インターフェイス（実践編 P. 131)」
・「ラムダ式（実践編 P. 132)」
・「ラムダ式の省略形（実践編 P. 134)」

## ● J2Kad20D「内部クラス」

### 1．内部クラスとは（実践編 P.125）

クラスやメソッドの中でもクラス宣言を行うことができます。これを内部クラスと呼びます。ただし、クラス内で宣言したときはそのクラスでのみ、メソッド内で宣言したときはそのメソッド内でのみ使うことができます。
課題では実践編 P.128、**List⑥-2** の Person クラスを外部クラス（OuterPerson）と内部クラス（InnerPerson）として宣言して動作確認をします。

```
    Greeting.greet(new OuterPerson);          // こちらでも OK
```

## ● J2Kad20C「匿名クラス（無名クラス）」

### １．匿名クラス（実践編 P.127）

クラスのインスタンスを生成するときに同時にサブクラスの宣言を行うこともできます。
このとき宣言したサブクラスには名前は必要ありません
（ただしこのときに生成したインスタンスでしか使えないクラスになります)。
また、メソッドの引数を指定する場所でクラスの宣言を行うことも可能です。
この場合もクラスに名前は必要ありません（ただしその引数でしか使えません)。
このように名前のないクラスを匿名クラス（あるいは無名クラス）と呼びます。

```
// インスタンス生成時にサブクラス宣言、ただし s1 でしか使えない
SayHello s1 = new SayHello() {
    public void hello() { System.out.println("匿名クラス①：こんにちは！"); }
};
Greeting.greet(s1);

// 引数指定時にサブクラス宣言、ただしこの引数でしか使えない
Greeting.greet(
    new SayHello() {
        public void hello() { System.out.println("匿名クラス②：こんにちは！"); }
    }
);
```

## ● J2Kad20B「ラムダ式」

### １．関数型インターフェイス（実践編 P.131）

抽象メソッドを１つしか含まないインターフェイスを関数型インターフェイスと呼びます。課題で扱う SayHello インターフェイスが関数型インターフェイスに該当します

### ２．ラムダ式（実践編 P.132）

J2Kad20C にて、インスタンス生成時にサブクラスを宣言する場合、サブクラスは SayHello を実装したクラスであることは明白です。また SayHello は関数型インターフェイスなのでオーバーライドするメソッドも hello メソッドしかありません。つまりこれらはわざわざ記述しなくても推論できる部分となります（下記、太字部分)。

```
SayHello s1 = new SayHello() { public void hello() { System.out.println("匿名クラス①：こんにちは！"); } };
```

Java ではこれらを削除し、hello メソッドの引数列()と hello メソッドのコード部分{}の間に「->」を挟んで記述することができます。これをラムダ式と呼びます。

```
SayHello s1 = () -> { System.out.println("匿名クラス①：こんにちは！"); };      // ラムダ式
```

ラムダ式の基本的な記述は以下のようになります。
(引数列) -> {処理内容}

## ● J2Kad20A「ラムダ式の省略形」

### １．ラムダ式の省略形（実践編 P.134）

実践編 P.136、**List⑥-5** をもとに P.134～P.135 にあるラムダ式の省略形を実際に入力して動作を確認します。
また可能であれば J2Kad20B へ戻って、こちらも省略形にしてください。

## ● J2Kad20S「ライフゲーム①（初期データの表示）」 &J2Kad20X「ライフゲーム②（実行！）」

### １．ライフゲーム

ライフゲームとは生命の誕生、進化、淘汰などのプロセスを簡易的なモデルで再現したシミュレーションです（Wikipedia より）。課題 S では初期データのロード、課題 X では実際のシミュレーションを作成します。どういうふうに作成するのか、各学生に考えさせてください（たぶんいろんな作り方があります）。

なお、課題 X の解答例では Canvas を 2 つ使っています。更新前のデータ用と更新後のデータ用です。これを交互に切り換えながらデータ更新を行っています。