Woodland Academy

オブジェクト指向その他補足資料

- 列挙型
- 内部クラス

Shape Your Future

目次

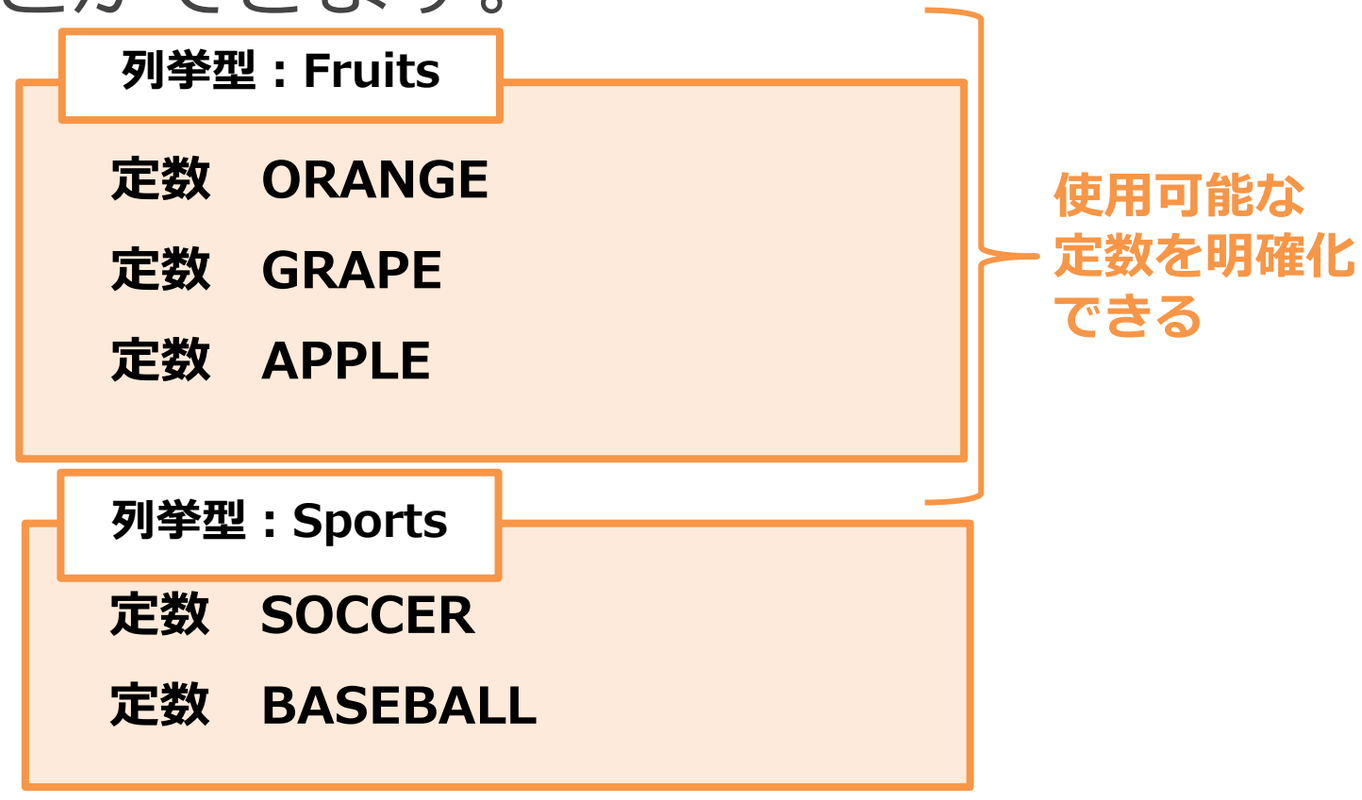
目次

列挙型

- **列挙型**[Enum]とは、**関連する定数をひとつにまとめておくこと**ができる型のことです。列挙型で管理することで、**使用可能な定数を明確化**することができます。

- Enum（列挙型）で定義する定数のことを列挙子と言います。

- Javaの列挙型はクラスですので、フィールドやメソッドを定義することができます。



列挙型で管理する場合

列挙型の定義

- 列挙型を定義するには、**enum** キーワードを使用します：

```
1 public enum Status {
2     RUNNING, POWERED_DOWN, SLEEPING
3 }
```

- ここで、中括弧「{}」には、取りうるすべての列挙子を書きます。列挙子のリストはカンマ「,」で区切ります。

Note

列挙子は、**大文字のスネークケース**で命名すべきです。

列挙型の使用

- 列挙型は他のクラスと同じよう扱うことができます。つまり、「Status」は普通のクラス名として、**変数や引数のタイプ**になり得ます。なお、Status 型の変数が取り得る値は、上記の 3 つだけです：

```
Status pc1Status = Status.RUNNING;
Status pc2Status = Status.POWERED_DOWN;
```

- 列挙された列挙子が、**クラス変数**（static な変数）と同じように使われていることがわかります。

列挙型と switch 文

- 列挙型のもう一つの便利な点は、**switch 文**（↰§ 1.3.1）で直接に扱うことができます：

```
switch (pc1Status) {
    case RUNNING:
        System.out.println("PC1 is running.");
        break;
    case POWERED_DOWN:
        System.out.println("PC1 is not running.");
        break;
    case SLEEPING:
        System.out.println("PC1 is sleeping.");
        break;
}
```

列挙型のインスタンス変数とメソッド

- 列挙型はクラスなので、メンバ変数やメソッド、コンストラクタを持つこともできます。コンストラクタがある場合、定義された各列挙子に対して呼び出す必要があります。

```
1 public enum State {
2     RUNNING(1.0f), POWERED_DOWN(0.0f), SLEEPING(0.2f); // ここの「;」は省略不可
3
4     private float power;
5
6     State(float power) {
7         this.power = power;
8     }
9
10    public float getPower() {
11        return power;
12    }
13 }
```

Try
enums パッケージ

- 列挙型のコンストラクタは自動的に private になります。

Woodland Academy

Q&A

目次

内部クラス

- Java では、あるクラスの中に他のクラスを定義することができます。これは**入れ子クラス**や**内部クラス**[Inner Class]と呼ばれます：

```
1 public class Outer {
2     class Inner { }
3 }
```

- よくある使い方は、いくつかの小さなクラスを、それらを使うクラスに**集約される**ことです。例えば、ダイヤのクラスは、車のクラスだけに使われています。この場合、ダイヤクラスを車クラスの内部クラスとして書いてもいいでしょう：

```
1 public class Car {
2     private class Wheel { }
3 }
```

Woodland Academy

静的・非静的内部クラス

- 変数やメソッドと同様に、内部クラスも静的（static）と非静的の 2 種類があります。静的な内部クラスは直接インスタンス化することができますが、非静的な内部クラスは外部クラスのオブジェクトによってインスタンス化する必要があります。インスタンス化されたオブジェクトはこの外部クラスのオブジェクトに依存します。

- 一般的に、静的な内部クラスは単純な、あるいは誰でも使えるユティリティクラスを実装するために使います。非静的な内部クラスは、外部クラスに依存するクラスを実装するために使います。

- 例えば、クラス内で使用する**列挙型**は、よく内部クラスとして記述されています。

Woodland Academy

静的内部クラス

- 静的内部クラスは、機能的には通常のクラスと変わりなく、使用する主な目的は、異なるクラスを一緒に**まとめる**ことです。static キーワードを使用して定義します：

```
1 public class Car {
2     Wheel[] wheels;
3
4     private static class Wheel { }
5 }
```

- 外部クラス以外の他のクラスを使用するには、**外部クラス名**と内部クラス名を「.」で繋いで使わなければなりません：

```
Car.Wheel wheel = new Car.Wheel();
```

非静的内部クラス

- 非静的内部クラスは、一般的に外部クラスに明示的に**依存する**クラスを定義するために使用されます。例えば、運転手は常に自分の車を運転することができます：

```
1 public class Car {
2     private Driver Driver;
3
4     private class Driver { }
5 }
```

- 他のクラスがこのような内部クラスをインスタンス化したい場合は、外部クラスの**オブジェクト名**と「new」を「.」で繋いで使う必要があります：

```
1 Car car = new Car();
2 Car.Driver driver = car.new Driver();
```

次へ

Woodland Academy

- 非静的内部クラスは、外部クラスのあるオブジェクトに依存しているため、そのオブジェクトのメンバ変数やメソッドを直接利用することができます：

```
1 public class Car {
2     private Driver Driver;
3     private String brand;
4
5     private class Driver {
6         public String getCarBrand() {
7             return brand;
8         }
9     }
10 }
```

Try Car.java

内部クラス：まとめ

Sum Up

内部クラスの構文や使い方は複雑ですが、現段階で覚えておきたいことは 2 つだけです：

1. 内部クラス名の表現方式、すなわち Outer.Inner の形式。なぜなら、後で使う外部ライブには内部クラスがあります。

2. 非静的内部クラスは、外部クラスのインスタンス変数やメソッドにアクセスすることができます。なぜなら、後では特別な非静的内部クラス（§ 3.2.2）を使うことになります。

Woodland Academy

Q&A

まとめ

Sum Up

1.列挙型の定義と使い方。

2.内部クラスの基本的な使い方。

Woodland Academy

Thank you!

From Seeds to Woodland — Shape Your Future.

Shape Your Future