第10回 継承とオーバーライド — 振り返り

この単元の前に、前回までの内容を振り返りましょう。下のスペースに図やメモを書いてもかまいません。

継承とオーバーライド（この単元）— Lesson 10・約2時間

この単元の前に：オブジェクト指向の復習

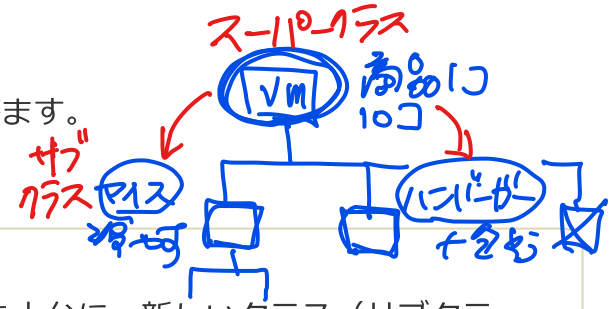
前回は「コンストラクタ」と「カプセル化」を自販機で学びました。その部分を短くおさらいします。

復習：前回までに学んだこと

- **コンストラクタ** — `new クラス名()` でインスタンスが作られるときに一度だけ実行される特別なメソッド。クラス名と同じ名前で、初期値をセットする。
- **カプセル化** — フィールドを `private` にして、getter・setter からだけ操作する。
- **VendMachine** — 飲み物の名前（`name`）と在庫（`stock`）を持ち、`buy()` で購入できる自販機クラスとしてここまで拡張してきた。

この単元で学ぶことのイメージ

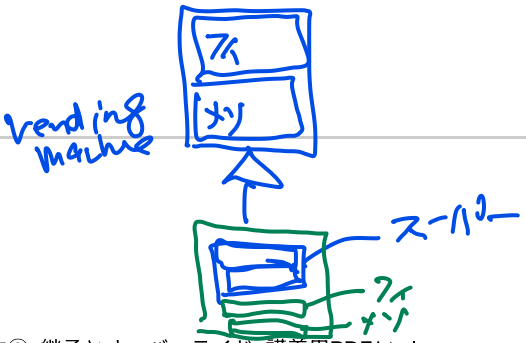
この単元では「継承」と「オーバーライド」を学びます。



Vending machine

- **継承** — 既存のクラス（スーパークラス）を土台に、新しいクラス（サブクラス）を「拡張」して作る仕組み。サブクラスはスーパークラスのフィールド・メソッドをそのまま使える。
- **オーバーライド** — スーパークラスが持つメソッドを、サブクラスで「同じ名前・同じ引数」で書き直すこと。サブクラスの動きをスーパークラスと変えられる。

自販機を「普通の自販機」と「冷飲料専用機」「温飲料専用機」のように種類で広げるイメージで進めます。



【1】具体的な学習目標

この単元を終えたとき、以下のことができればゴールです。

【必須】

- 継承の役割（スーパークラスの機能を引き継いでサブクラスで拡張すること）を説明できる。
- `extends` を使って、自販機クラスを継承したサブクラスを書ける。
- オーバーライドの役割（スーパークラスのメソッドをサブクラスで上書きすること）を説明できる。
- スーパークラスと同じ名前・同じ引数のメソッドをサブクラスで定義し、オーバーライドできる。

【オプション】

- `super` でスーパークラスのコンストラクタやメソッドを呼び出せる。

【2】導入：自販機の「型」を広げる

カバ先生：「ここまでで、自販機クラスにコンストラクタとカプセル化を入れたね。次は『自販機にもいろいろ種類がある』ところをプログラムで表現したい。例えば『冷えた飲み物だけ売る機』と『温かい飲み物だけ売る機』だと、買ったときのメッセージを『冷えてます』『温かくしてお渡しします』のように変えたい。そんなときに使うのが継承とオーバーライドだよ。」

ナナコ：「スーパークラスの自販機を土台にして、『冷飲料用』『温飲料用』のサブクラスを作る、ってことですね。」

ユウタ：「サブクラスで `buy()` の表示だけ変えるのがオーバーライド？」

カバ先生：「その通り。まずは継承の書き方から見ていこう。」

【3】継承とは

継承では、**既存のクラス（スーパークラス）**を土台に、**新しいクラス（サブクラス）**を定義します。サブクラスは `extends スーパークラス名` と書くだけで、スーパークラスのフィールドとメソッドをそのまま使えます。

- **スーパークラス** — 継承の元になるクラス。
- **サブクラス** — スーパークラスを継承したクラス。 `class サブクラス名 extends スーパークラス名 { ... }` と書く。
- サブクラスはスーパークラスの `public` なメソッド・フィールドを引き継ぐ。サブクラスだけで追加のフィールドやメソッドも持てる。

サンプル：VendMachine を継承した ColdVendMachine（冷飲料専用機）

スーパークラスの `VendMachine` は、 `name` ・ `stock` と `buy()` を持っているとします。

```java
// スーパークラス
public class VendMachine {
    private String name;
    private int stock;

    public VendMachine(String name, int stock) {
        this.name = name;
        this.stock = stock;
    }

    public String getName() { return name; }
    public int getStock() { return stock; }

    public void buy() {
        if (stock > 0) {
            System.out.println(name + "を購入しました");
            stock--;
        } else {
            System.out.println("売り切れです");
        }
    }
}

// サブクラス：VendMachine を継承
public class ColdVendMachine extends VendMachine {

    public ColdVendMachine(String name, int stock) {
        super(name, stock);  // スーパークラスのコンストラクタを呼ぶ
    }
}
```

サブクラスでは `super(引数)` でスーパークラスのコンストラクタを必ず呼びます。こうするとスーパークラスの `name` と `stock` が初期化されます。サブクラスはスーパークラスの `buy()` をそのまま使えます。

【4】オーバーライドとは

オーバーライドは、**スーパークラスが持つメソッドを、サブクラスで「同じ名前・同じ引数」で定義し直す**ことです。サブクラスのインスタンスでそのメソッドを呼ぶと、スーパークラスの処理ではなくサブクラスの処理が実行されます。

- スーパークラスのメソッドと**名前・引数の型・並び**を同じにする。
- 戻り値の型もスーパークラスと同じにする（`void` なら `void`）。
- オーバーライドであることを明示するために、`@Override` をメソッドの直前に書く習慣がある（省略可能）。

サンプル：ColdVendMachine で buy() をオーバーライドする

```
public class ColdVendMachine extends VendMachine {

    public ColdVendMachine(String name, int stock) {
        super(name, stock);
    }

    // スーパークラスの buy() をオーバーライド：冷飲料用のメッセージに変える
    @Override
    public void buy() {
        if (getStock() > 0) {
            System.out.println(getName() + "を購入しました。冷えてます。");
            super.buy();
        } else {
            System.out.println("売り切れです");
        }
    }
}
```

このようにサブクラスで `buy()` を書き直すと、`ColdVendMachine` のインスタンスで `buy()` を呼んだとき「〇〇を購入しました。冷えてます。」と表示されます。

【5】super でスーパークラスのメソッドを呼ぶ

オーバーライドしたメソッドの中から、**スーパークラスの同じメソッド**を呼びたいときは `super.メソッド名(引数)` と書きます。在庫を減らす処理はスーパークラスの `buy()` に任せて、表示だけサブクラスで変える例です。

```
// 表示だけサブクラスで変え、在庫処理はスーパークラスに任せる
@Override
public void buy() {
    if (getStock() > 0) {
        System.out.println(getName() + "を購入しました。冷えてます。");
        super.buy();
    } else {
        System.out.println("売り切れです");
    }
}
```

スーパークラスの `buy()` が「〇〇を購入しました」と表示するなら、サブクラスで「冷えてます」だけ追加して在庫は `super.buy()` に任せる形もよくあります。どちらも「スーパークラスの処理を呼ぶ」ために `super` を使います。

> **ユウタ：**「`super` はスーパークラスのコンストラクタを呼ぶときと、スーパークラスのメソッドを呼ぶときの両方で使うんだね。」

【6】練習問題（継承）

【問題1】VendMachine を継承したサブクラスを作る

`VendMachine` を継承した `HotVendMachine`（温飲料専用機）クラスを作成してください。コンストラクタは `HotVendMachine(String name, int stock)` とし、中で `super(name, stock)` を呼んでください。

【問題2】buy() をオーバーライドする

`HotVendMachine` で `buy()` をオーバーライドしてください。在庫が 0 より大きいときは「〇〇を購入しました。温かくしてお渡しします。」と表示して在庫を1減らし、0 以下のときは「売り切れです」と表示するようにしてください。getter を使って名前・在庫を取得します。

【7】まとめのおさらい

- **継承** — `extends スーパークラス名` でサブクラスがスーパークラスの機能を引き継ぐ。サブクラスのコンストラクタでは `super(引数)` でスーパークラスのコンストラクタを呼ぶ。
- **オーバーライド** — スーパークラスと同じ名前・同じ引数のメソッドをサブクラスで定義し直す。サブクラスのインスタンスではそのメソッドを呼ぶとサブクラスの処理が実行される。
- `super.メソッド名(引数)` でスーパークラスのメソッドを呼べる。`@Override` はオーバーライドであることを示す注釈。

> **ナナコ：**「自販機の種類を継承で増やして、buy() の表示をオーバーライドで変える。同じ型で扱いながら動きを変えられそう。」

> **カバ先生：**「次は、抽象クラスやインタフェースで『型の決まり』を学んで、同じ型でいろいろな自販機を扱う多態性に進んでいこう！」