

3-5.オブジェクト指向

■ オブジェクト指向とは・・・

はじめに

オブジェクト指向とは、作りたいもの（オブジェクト）を、一つ一つの部品に分けて作っていくことです。

クルマ作りにたとえて、工場内に6つのチームがあるとします。

オブジェクト指向によらないクルマ作りは、6つのチームそれぞれでクルマを完成させます。

オブジェクト指向によるクルマ作りは、

- ボディを作るチーム
- ドアを作るチーム
- エンジンを作るチーム
- ウィンドウを作るチーム
- タイヤを作るチーム
- 上記各部品を合体させるチーム

のように **部品ごとにチームを分けて** クルマを完成させます。

どちらにもメリット、デメリットはありますが、

Javaはオブジェクト指向型言語といわれるくらいオブジェクト指向と相性がいいので、

ここで考え方を身に着けていきましょう！

オブジェクトを理解する上で避けては通れない用語を3つ覚えましょう！

①クラス

少し前のカリキュラムで出てきましたね。

クラスとは、オブジェクトの **設計書** のようなもので、後述のプロパティやメソッドをひとまとめにしたものです。

例えば、**車の設計書がクラスで、実際に作られた車がオブジェクト** です。

割と抽象的な概念なので、ここでは「**クラスとは設計書である**」とおぼえておきましょう！

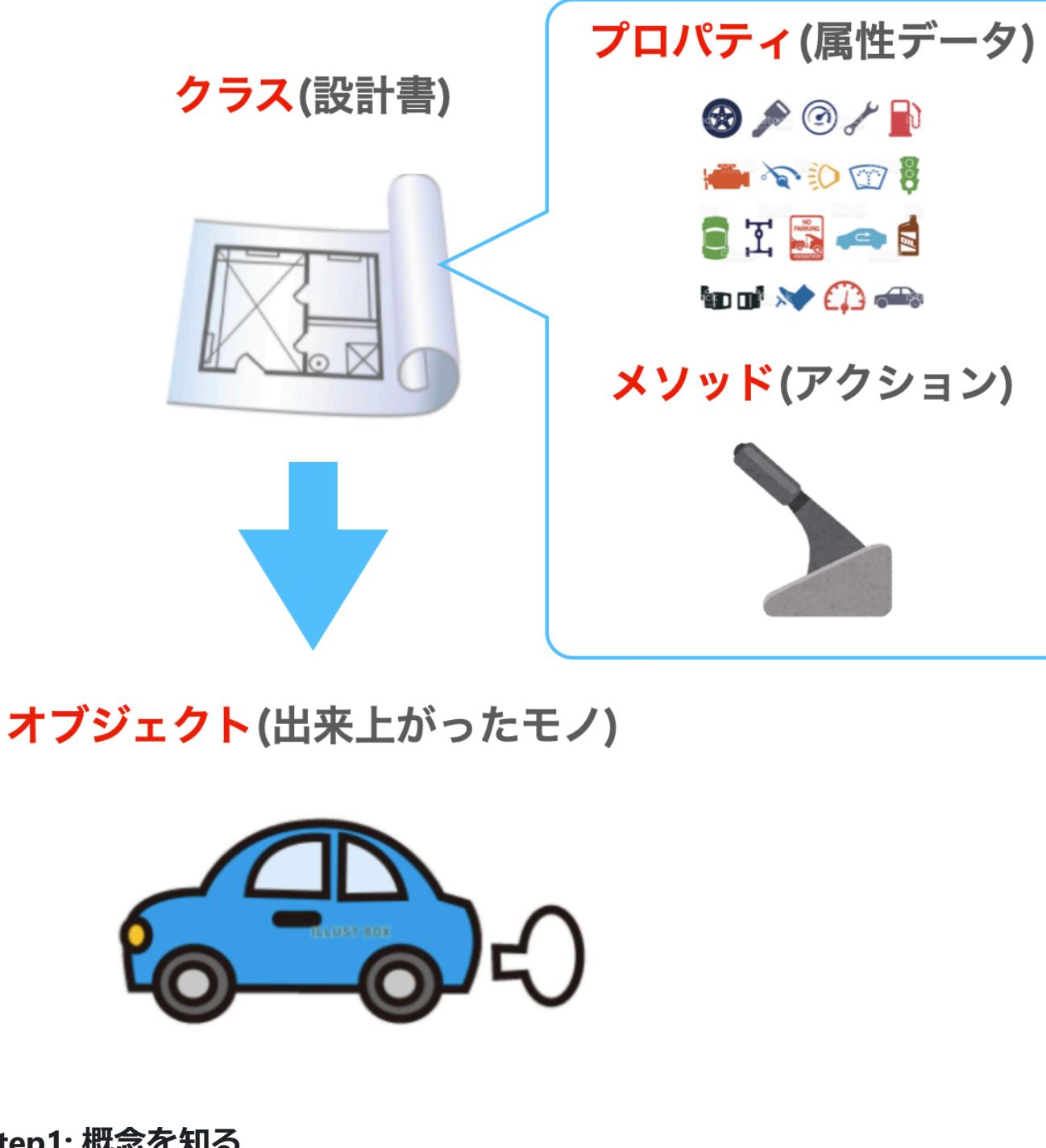
②プロパティ(属性)

オブジェクトが持っている **データ** のことです。

車の例だと、車というオブジェクトは「メーカー」、「色」、「排気量」といったプロパティを持っています。

③メソッド

オブジェクトが持っている処理のことで、車の例だと「走る」、「曲がる」、「止まる」などの **アクション** のことです。



Step1: 概念を知る

わかりやすいようにRPGで考えていきましょう！

RPGには、いろんなキャラクターがいます。



勇者



魔法使い



僧侶



ラスボス

どのキャラクターもみんな元々は人間なので
「人間クラス」を元に（継承）作られています。

人間クラス (Human.java)



Human.java (人間クラス)

```
public class Human {

    private String name = null; // なまえ
    private int gender = 0; // 性別(1:男 2:女)
    private int length = 0; // 身長
    private int weight = 0; // 体重
    private int vitality = 0; // たいりょく
    private int magic = 0; // まりょく

    // コンストラクタ
    public Human() {
    }

    // getter・setter(なまえ)
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    // getter・setter(性別)
    public int getGender() {
        return gender;
    }

    public void setGender(int gender) {
        this.gender = gender;
    }

    // getter・setter(身長)
    public int getLength() {
        return length;
    }

    public void setLength(int length) {
        this.length = length;
    }

    // getter・setter(体重)
    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    // getter・setter(たいりょく)
    public int getVitality() {
        return vitality;
    }

    public void setVitality(int vitality) {
        this.vitality = vitality;
    }

    // getter・setter(まりょく)
    public int getMagic() {
        return magic;
    }

    public void setMagic(int magic) {
        this.magic = magic;
    }

    // はなす
    public void talk(String about) {
        System.out.println(about);
    }

    // たべる
    public void eatFood(String food) {
        int foodType = 0;
        if ("やくそく".equals(food)) {
            foodType = 1;
        } else if ("まぼうのみず".equals(food)) {
            foodType = 2;
        } else {
            foodType = 3;
        }
        digestFood(foodType); // 食べ物を消化する
    }

    // 消化する
    private void digestFood(int foodType) {
        if (foodType == 1) {
            vitality += 10; // たいりょくを10回復
        } else if (foodType == 2) {
            magic += 10; // まりょくを10回復
        } else {
            vitality += 1; // たいりょくを1回復
        }
    }
}
```



勇者クラス (Yuusha.java)

人間クラスを継承したクラスで勇者特有の必殺技メソッドをもつ。

Yusha.java (勇者クラス)

```
public class Yusha extends Human {  
  
    // コンストラクタ  
    public Yusha() {  
        super.setName("ゆうしゃ");  
        super.setGender(1);  
        super.setLength(180);  
        super.setWeight(70);  
        super.setVitality(100);  
        super.setMagic(5);  
    }  
  
    public void specialAttack(Human target) {  
        String name = super.getName();  
        super.setVitality(super.getVitality() - 20);  
        System.out.println(name + " の こうげき");  
        System.out.println(name + " の ひっさつわざが さくれつした!");  
  
        target.setVitality(target.getVitality() - 50);  
        System.out.println(target.getName() + " に 50 のダメージを あたえた");  
  
        super.setVitality(super.getVitality() - 10);  
        System.out.println(name + " の たいりょくば" + super.getVitality() + "になった");  
        System.out.println("");  
    }  
}
```



魔法使いクラス (Wizard.java)

人間クラスを継承したクラスで魔法使い特有の攻撃魔法メソッドをもつ。

Wizard.java (魔法使いクラス)

```
public class Wizard extends Human {
    // コンストラクタ
    public Wizard() {
        super.setName("まほうつかい");
        super.setGender(1);
        super.setLength(170);
        super.setWeight(60);
        super.setVitality(20);
        super.setMagic(50);
    }

    public void magicAttack(Human target) {
        String name = super.getName();
        System.out.println(name + " の こうげき");
        System.out.println(name + " は こうげきじゅもん を となえた!");

        target.setVitality(target.getVitality() - 20);
        System.out.println(target.getName() + " に 20 のダメージを あたえた");

        super.setMagic(super.getMagic() - 10);
        System.out.println(name + " の まりよくは " + super.getMagic() + " になった");
        System.out.println("");
    }
}
```



Cleric.java (僧侶クラス)

人間クラスを継承したクラスで回復魔法のメソッドをもつ。

Cleric.java (僧侶クラス)

```
public class Cleric extends Human {
    // コンストラクタ
    public Cleric() {
        super.setName("そうりょ");
        super.setGender(2);
        super.setLength(160);
        super.setWeight(50);
        super.setVitality(20);
        super.setMagic(70);
    }

    // かいふくじゅもん
    public void healingMagic(Human target) {
        String name = super.getName();
        System.out.println(name + " は かいふくじゅもんを となえた!");

        target.setVitality(target.getVitality() + 20);
        System.out.println(target.getName() + " の たいりょくは " + target.getVitality() + " になった");

        target.setMagic(super.getMagic() - 10);
        System.out.println(name + " の まりよくは " + super.getMagic() + " になった");
        System.out.println("");
    }
}
```



ラスボスクラス (LastBoss.java)

人間クラスを継承しており、ラスボスの邪悪な必殺技メソッドをもつ。

LastBoss.java (ラスボスクラス)

```
public class LastBoss extends Human {
    // コンストラクタ
    public LastBoss() {
        super.setName("ラスボス");
        super.setGender(1);
        super.setLength(210);
        super.setWeight(120);
        super.setVitality(500);
        super.setMagic(40);
    }

    public void specialEvilAttack(Human target) {
        String name = super.getName();
        System.out.println(name + " の こうげき");
        System.out.println(name + " の じやあくな ひっさつわざが さくれつした!");

        target.setVitality(target.getVitality() - 30);
        System.out.println(target.getName() + " は 30 のダメージをうけた");

        super.setVitality(super.getVitality() - 10);
        System.out.println(name + " の たいりょくは" + super.getVitality() + "になった");
        System.out.println("");
    }
}
```

ラスボスとの戦闘シーンを思い描いてください。



メイン処理

プログラムの主処理を記述します。
RPG風のメッセージを表示し、各キャラクターのステータスを表示します。

Main.java

```

public class Main {
    // メイン処理
    public static void main(String[] args) {
        Yusha Yusha = new Yusha(); // 勇者オブジェクトを生成
        Wizard wizard = new Wizard(); // 魔法使いオブジェクトを生成
        Cleric crelic = new Cleric(); // 僧侶オブジェクトを生成
        LastBoss lastBoss = new LastBoss(); // ラスボスオブジェクトを生成

        // 敵があらわれた！
        System.out.println(lastBoss.getName() + " が あらわれた！");

        // バトル開始
        Yusha.specialAttack(lastBoss); // 勇者の攻撃
        wizard.magicAttack(lastBoss); // 魔法使いの攻撃
        lastBoss.specialEvilAttack(Yusha); // ラスボスの攻撃
        crelic.healingMagic(Yusha); // 僧侶が勇者を回復

        // 回復アイテムを使用
        System.out.println(Yusha.getName() + " は やくそう をつかった");
        Yusha.eatFood("やくそう");
        System.out.println(wizard.getName() + " は まほうのみず をつかった");
        wizard.eatFood("まほうのみず");
        System.out.println("");

        // ステータス表示
        showStatus(Yusha);
        showStatus(wizard);
        showStatus(crelic);
        showStatus(lastBoss);

    }

    // ステータス表示メソッド
    private static void showStatus(Human target) {
        System.out.println("- " + target.getName() + " の ステータス---");
        if (target.getGender() == 1) {
            System.out.println("せいべつ : 男");
        } else {
            System.out.println("せいべつ : 女");
        }
        System.out.println("しんちょう : " + target.getLength());
        System.out.println("たいじゅう : " + target.getWeight());
        System.out.println("たいりょく : " + target.getVitality());
        System.out.println("まりょく : " + target.getMagic());
        System.out.println("");
    }
}

```

【実行結果】

```

らすばす が あらわれた！
ゆうしゃ の こうげき
ゆうしゃ の ひっさつわざが さくれつした！
らすばす に 50 のダメージを あたえた
ゆうしゃ の たいりょくは70になった

まほうつかい の こうげき
まほうつかい は こうげきじゅもん を となえた！
らすばす に 20 のダメージを あたえた
まほうつかい の まりょくは 40 になった

らすばす の こうげき
らすばす の じゃあくな ひっさつわざが さくれつした！
ゆうしゃ は 30 のダメージを うけた
らすばす の たいりょくは420になった

そうりょ は かいふくじゅもんを となえた！
ゆうしゃ の たいりょくは 60 になった
そうりょ の まりょくは 70 になった

ゆうしゃ は やくそう をつかった
まほうつかい は まほうのみず をつかった

- ゆうしゃ の ステータス---
せいべつ : 男
しんちょう : 180
たいじゅう : 70
たいりょく : 70
まりょく : 60

- まほうつかい の ステータス---
せいべつ : 男
しんちょう : 170
たいじゅう : 60
たいりょく : 20
まりょく : 50

- そうりょ の ステータス---
せいべつ : 女
しんちょう : 160
たいじゅう : 50
たいりょく : 20
まりょく : 70

- らすばす の ステータス---
せいべつ : 男
しんちょう : 210
たいじゅう : 120
たいりょく : 420
まりょく : 40

```

Step2: 使い方を知る

プログラムを解説していきます！

Humanクラス

Humanクラスには身長、体重、体力などRPGのキャラクターのベースとなるプロパティや、話す `talk`、食べる `eatFood`などのメソッドが設定されています。

各プロパティは公開範囲が `private` なので、データを参照したり変更したりするには、`setName`、`getName`といった `setter`、`getter` と呼ばれるメソッドを使う必要があります。

消化する `digestFood` というメソッドもありますが、これは体の内部のことでの公開する必要もないため、カプセル化といって処理はするものの、結果を表示させることはしていません。

Yusha、Wizard、Crelic、LastBossクラス

・ `extends Human`: 繙承

ここでは **Humanクラスを継承すること** により、Humanクラスが持つ人間の基本動作をそのまま利用しています。必要最低限の「共通する基本情報」を用意しておくと、Human クラスの汎用性が高まります。

・ 独自のメソッドの実装

継承した人間の基本動作に加えて、それぞれのキャラクター固有の必殺技や魔法攻撃のメソッドを持っています。
「継承 + 独自のメソッドやプロパティ」を実装することで、継承したクラスに付加価値を与えることができます。

・ コンストラクタによる初期設定

各キャラクターは **コンストラクタ** により、身長、体重、体力などの初期設定が可能です。
オブジェクトを最初に生成する際に実行されるため、**初期設定** をするのに適しています。

メイン処理

・ プログラムの流れ

1: 各キャラクターのオブジェクトを生成

オブジェクトに命が吹き込まれてRPG風の戦闘が始まります。
(new Humanクラスを継承した各クラス

2: 各キャラクターの攻撃メソッドにターゲットを指定

これにより指定したターゲットの体力が減少します。

例) 僧侶（Crelic クラス）の場合

1. 回復メソッドを呼び出す
2. ターゲットの体力が回復する処理

メソッドのターゲットの指定には、Humanクラスを設定しています。

Humanクラスは全キャラクターが継承しているクラスなので、どのキャラクターでも対象として指定できます。

3: 回復

クラスのメソッドや回復アイテムの使用により、回復するプロパティや分量が変わるように設計されています。

例) Humanクラスの食べるメソッド eatFood

Humanクラスで設定された消化メソッド `digestFood` が呼び出され、回復する仕組みになっています。
Humanクラスの説明でも既出ですが、カプセル化により隠蔽されています。

4: 各キャラクターのステータス

生成した各クラスのオブジェクトごとにプロパティの値は異なります。
また、戦いによるダメージによって各キャラクターのプロパティが変わっているのがわかると思います。
(ここでは「たいりょく」がそれにあたりますね

Step3: オブジェクト指向的にキャラクターを作っていない場合の問題点

コードが見づらくなる

Humanクラスを継承せずにYushaクラス、Wizardクラスなどに毎回Humanメソッドを記述しても結果は変わりません。しかし、コードが非常に長くなり、見づらいコードとなってしまいます。

機能を増やすときの手間が増える

人気RPGなので大型アップデートとして新しく王宮戦士、王女、神官、商人、踊り子、裏ボスのキャラクターを追加することになりました。キャラクターごとにHumanメソッドを記述していた場合、新しいキャラクターには機能追加とHumanメソッドの記述が必要ですが、オブジェクト指向的に記述していれば、機能追加をすれば完了です。

修正漏れが起きやすくなる

人間の基本動作に、宿屋に泊まることで体力、魔力が全回復する「寝る」という機能をつけるのを忘れていました。キャラクターごとにHumanメソッドを記述していた場合、一個一個に「寝る」というメソッドを記述し修正しなければいけませんが、オブジェクト指向的に記述していれば、Humanクラスに「寝る」というメソッドを一回記述すれば修正完了です。

課題

提出課題はありませんので、一通り学習が終わったら次の章に進んで下さい。

最終更新日時: 2022年 08月 21日(日曜日) 15:26