

- クラスを作るってどういうこと？
- 既存のクラスを拡張したクラスを作ろう
- メソッド

★5 クラスの作成 (p.44)

★5.1 クラスを作るってどういうこと？

★5.1.1 かめさんプログラム再考

これまで作ってきたプログラムのうち、かめさんを動かすものは、図1のような構成をしていた。われわれが自分で作るのは Hoge.java とそれをコンパイルした Hoge.class のみで、タートルグラフィックスのためのクラス (TurtleFrame や Turtle など) については、予め用意されたクラスファイル (TurtleFrame.class や Turtle.class など) を利用していた。今回は、他のプログラムから利用できるような新しいクラスを自分で定義する方法を学ぶ。

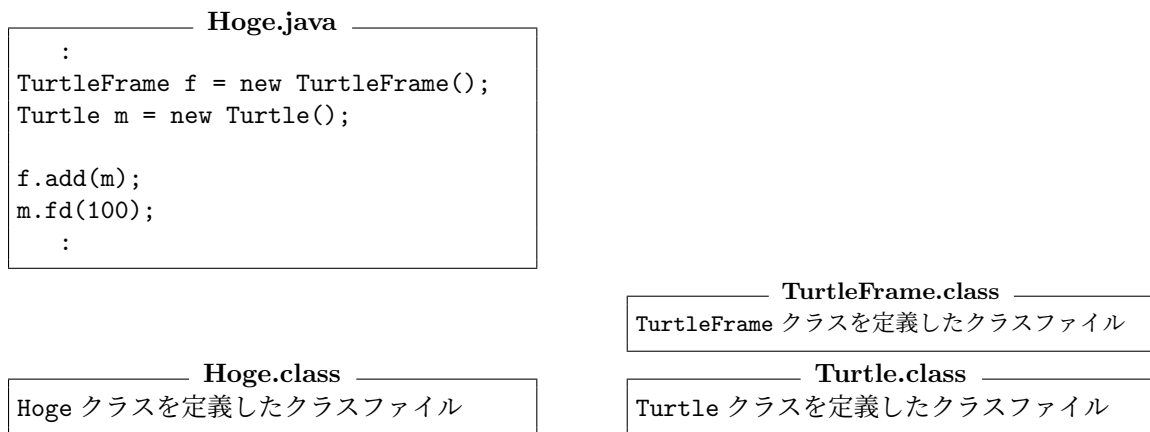


図 1: これまでのかめさんプログラムの構成例

★5.1.2 作る話の前に使う話の復習をしよう

自分でクラスを作る方法を学ぶ準備として、他人が作ったクラスを利用する例で復習しよう。

HW クラスの API 仕様 個人の身長体重のデータを扱うためのクラス

● コンストラクタ

HW() 名前 "ほげお", 身長 170.0[cm], 体重 62.0[kg] のインスタンスを生成.

HW(String n, double h, double w) 名前 n, 身長 h[cm], 体重 w[kg] のインスタンスを生成.

● メソッド

void print() このインスタンスの名前, 身長, 体重を表示する.

double calcBMI() このインスタンスの BMI を計算して返す. $BMI = (\text{体重 [kg]}) / (\text{身長 [cm]} / 100)^2$

static void printHimando(double bmi) bmi が Border1 未満, Border1 以上 Border2 未満, Border2 以上の三つの場合に分けてメッセージを出力. 順に, 「やせてんなあ」, 「ふつー」, 「太ってる…かな？」.

● フィールド

String name 名前. **double height** 身長. **double weight** 体重.

static double Border1 BMI の境界値のうち小さい方. 初期値は 20.

static double Border2 BMI の境界値のうち大きい方. 初期値は 25.

G05Fugayo.java

```
1  /** HW クラスを使うプログラム */
2  public class G05Fugayo{
3      public static void main(String[] args){
4          HW p1 = new HW();
5          p1.print();

6          double bmi = p1.calcBMI();
7          HW.printHimando(bmi);

8          HW p2 = new HW("ふがよ", 234.5, 50.0);
9          p2.print();

10         HW.printHimando(p2.calcBMI());

11         p2.weight = 120.0;
12         p2.print();
13         HW.printHimando(p2.calcBMI());

14         HW.Border1 = 24.0;
15         HW.printHimando(p1.calcBMI());
16         HW.printHimando(p2.calcBMI());
17     }
18 }
```

G05Fugayo の実行結果 (注)

ほげおさんの身長は 170.0[cm], 体重は 62.0[kg] です
BMI は 21.45 ふつー

ふがよさんの身長は 234.5[cm], 体重は 50.0[kg] です
BMI は 9.09 やせてんなあ

ふがよさんの身長は 234.5[cm], 体重は 120.0[kg] です
BMI は 21.82 ふつー

BMI は 21.45 やせてんなあ
BMI は 21.82 やせてんなあ

注) G05Fugayo.class と同じディレクトリ内に HW.class があると想定している。また、この実行結果には、見やすくするために改行が挿入されている

★ 5.1.3 既存のクラスを拡張したクラスを作ろう

次節以降ではじめに考える例は、図 2 に示すような構成のものである (☆ 1)。T51.java と HTurtle.java の二つのソースを作成する。HTurtle.java は新しく作成する HTurtle というクラスの定義を書いたものであり、T51.java はそれを利用するものである。

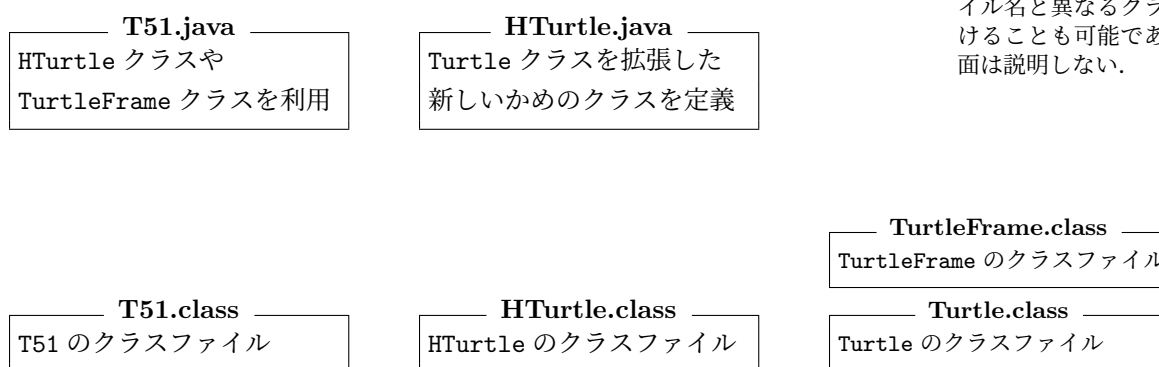


図 2: HTurtle クラスを用いるプログラムの構成例

HTurtle クラスは、今まで用いてきた Turtle クラスに新たな機能を追加して作成する。このように既存のクラスに機能を追加して新しいクラスを定義することを、既存のクラスを**拡張**するという。また、クラス A を拡張してクラス B が定義されているとき、B は A の**サブクラス (子クラス)**であるといい、A は B の**スーパークラス (親クラス)**であるという。HTurtle クラスは Turtle クラスを拡張しており、HTurtle クラスは Turtle クラスのサブクラスである。

クラス A $\xrightarrow{\text{拡張}}$ クラス B $\xrightarrow{\text{拡張}}$ クラス C という関係の場合、A も B も C のスーパークラスであり、B も C も A のサブクラスである。Java には、あらゆるクラスのスーパークラスとして **Object クラス**が存在しており、全てのクラスは Object クラスのサブクラスである (☆ 2)。

あるクラスのオブジェクトは、自分のインスタンスメソッドやインスタンス変数の他に、自身のスーパークラスで定義されたインスタンスメソッドやインスタンス変数も引き継いでもっている。このことを、メソッドや変数をスーパークラスから**継承**するという。したがって、HTurtle クラスは、Turtle クラスとそのスーパークラスの全てのインスタンスメソッド/変数を継承している (☆ 3)。クラスメソッド/クラス変数についても同じ様なことがいえる (☆ 4)。

☆ 1) これらの例では、全てのソースファイルは一つだけクラスを定義しており、ソースファイル名とクラス名が一致している。実際には、一つのソースファイル中に複数のクラスを定義したり、ソースファイル名と異なるクラス名をつけることも可能であるが、当方は説明しない。

☆ 2) 「`extends` クラス名」を省略すると、「`extends Object`」とみなされる。

☆ 3) Java API 仕様のページから `java.lang` パッケージの `Object` クラスを探せば、`Object` クラスのメソッドなど (あらゆるクラスに継承されている)を一覧することができる。

☆ 4) 詳しいことは省略するが、例えば HTurtle は Turtle クラスのサブクラスなので、`HTurtle.speedAll` というクラスメソッドや `HTurtle.withTurtleAll` というクラス変数を使える。

★ 5.2 メソッド

HTurtle クラスの中身を考えていこう。まずは新しいメソッドの追加から。

★ 5.2.1 メソッドの追加

Turtle クラスを拡張した HTurtle クラスを定義し、p.44 の API 仕様が示すような機能をもつ二つのメソッド `polygon` と `house` を作ることを考える。以下の HTurtle.java が HTurtle クラスを定義するプログラムであり、T51.java がそれを用いるプログラムである。

