- 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)

★5.0　クラスを作るってどういうこと？

★5.0.1　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．

Hoge.java

```
   :
TurtleFrame f = new TurtleFrame();
Turtle m = new Turtle();

f.add(m);
m.fd(100);
   :
```

TurtleFrame.class

TurtleFrame クラスを定義したクラスファイル

Hoge.class

Hoge クラスを定義したクラスファイル

Turtle.class

Turtle クラスを定義したクラスファイル

図 1: これまでのかめさんプログラムの構成例

★5.0.2　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 = (体重 [kg])/(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,「やせてんなぁ」,「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 やせてんなぁ

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 やせてんなぁ
BMI は 21.82 やせてんなぁ
```

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 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．

☆ 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，当面は説明しない．

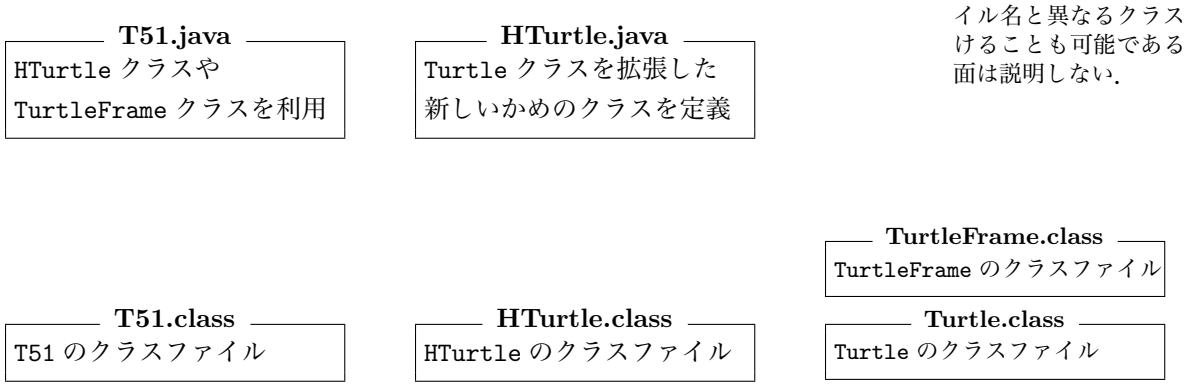

図 2: HTurtle クラスを用いるプログラムの構成例

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)．

☆ 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．

HTurtle クラスは Turtle クラスのインスタンスメソッド／変数を継承しているので，HTurtle クラスを利用する `T51.java` では，HTurtle の二つのインスタンスメソッドとともに，`fd` などの Turtle のインスタンスメソッドも使える．

T51.java

```
 1  import tg.*;
 2
 3  public class T51 {
 4      public static void main(String[] args){
 5          TurtleFrame f = new TurtleFrame();
 6          HTurtle m = new HTurtle();
 7          double size = 50;
 8          f.add(m);
 9          m.house(size);
10          m.up(); m.lt(90); m.fd(50); m.rt(72); m.down();
11          m.polygon(5, size / 2);
12          m.up(); m.moveTo(100,100,0); m.down();
13          m.polygon(10, size / 5);
14      }
15  }
```

HTurtle.java

```
 1  import tg.*;
 2
 3  public class HTurtle extends Turtle {          //Turtle を拡張する



 4      public double polygon(int n, double s){    //polygon メソッドの定義

 5          double a = 360.0/n;                    //曲がる角度を求めておく
 6          for(int j = 0; j < n; j++){            //n 回繰り返す
 7              fd(s);                             //s 前に進んで
 8              rt(a);                             //a 曲がるのを


 9           }
10           return n * s;
11      }


12      public void house(double s){               //house メソッドの定義

13          polygon(4, s);                         //polygon を利用
14          fd(s);
15          rt(30);
16          polygon(3, s);
17          lt(30);
18          bk(s);                                 //もとの場所に戻しておく
19      }
20  }
```