目次

★ 2 オブジェクトの生成とメソッド呼び出し

★2 オブジェクトの生成とメソッド呼び出し

注: この資料の ★ 2.1 のような番号は，教科書の章・節の番号と対応しているわけではありません．

★ 2.1 手続き型プログラミングとオブジェクト指向プログラミング

C 言語のような言語では，データとそれに対する処理をばらばらに書く．データにどのような処理を施すか，その手続きを記述するスタイルとなっている．このようなスタイルのプログラミング言語は**手続き型**であるという．

これに対して，Java は**オブジェクト指向**であるといわれる．オブジェクト指向のプログラミング (☆ 1) では，やりたいことを実現するために必要な「もの」を定義してそれを生成し，それにメッセージを送って何かをやってもらう（あるいは「もの」同士が相互作用する）という形をとる (☆ 2)．「もの」にはデータも手続きもひとまとめになっている．

☆ 1) オブジェクト指向プログラミング: object-oriented programming. 長いので，OOP と略すことも．

☆ 2) 例えば，かめとはどんなものかを定義しておき，その定義に従うかめを何匹か生成し，そのうち一匹に「前に 100 すすめ」というメッセージを送る．

★ 2.2 クラス，オブジェクト，インスタンス

上記の「もの」のことを**オブジェクト**といい，オブジェクトの定義を**クラス**という．また，あるクラスの定義から実際に生成されるオブジェクトの実体のことを**インスタンス**という．以下は，よくある直感的な説明…

object, class, instance

- クラスは設計図，インスタンスはその設計図から作られたもの
- クラスはクッキー型，インスタンスは型抜きしたクッキー

Java プログラミングでは，他人が作ったクラスを利用したり，自分でクラスを定義してそれを利用したりしながらプログラムを作っていく．

★ 2.3 API (Application Programming Interface)

クラスを用いたプログラムを書く人が利用できる機能（クラスのオブジェクトが受け付けるメソッドなど）のことを，そのクラスの **API**(☆ 3) という．p.12,13 には，TurtleFrame クラス，Turtle クラスの API の仕様がまとめられている．

☆ 3) API は，「計算機システム II」の授業でも，OS とアプリケーションプログラム (AP) との仲立ちをするものとして登場したかもしれません．

★ 2.4 オブジェクトの生成とメソッド呼び出し (p.8)

T21.java

```
1  /** 最初のプログラムの例 */
2  import tg.*;
3  public class T21 {
4     public static void main(String[] args){
5        TurtleFrame f;              // 変数 f の型宣言
6        f =  new TurtleFrame();     // TurtleFrame を生成し f に代入
7        Turtle m = new Turtle();    // Turtle を生成し，m の初期値として代入
8        Turtle m1 = new Turtle();   // もう一つ生成し，m1 の初期値として代入
9        f.add(m);                   // f に m を追加
10       f.add(m1);                  // f に m1 を追加
11       m.fd(100.0);                // m よ前に 100 進め
12       m.rt(90.0);                 // m よ右に 90 度回れ
13       m.fd(150.0);                // m よ前に 150 進め
14       m1.rt(90.0);                // m1 よ右に 90 度回れ
15       m1.fd(100.0);               // m1 よ前に 150 進め
16    }
17 }
```

- このプログラムが利用する Turtle クラスと TurtleFrame クラスは，このプログラムとは別の所で定義されている．詳しくは後述.
- TurtleFrame 型や Turtle 型のような，クラスに対応した型のことを，**クラス型**という
- **メソッド** (method) には，上記のような**インスタンスメソッド**の他に**クラスメソッド**というものもある（次回）

★ 2.5 プリミティブ値，コンストラクタ，ライブラリの利用 (p.14)

T22.java

```
1  import tg.*;
2  public class T22 {
3      public static void main(String[] args){
4          double x = 300.0, y = 200.0, d = 100.0;              // double 型の変数を用意
5          TurtleFrame f = new TurtleFrame(700.0, 500.0);  //引数のあるコンストラクタ呼出し
6          Turtle m = new Turtle(x, y, 180.0);
7          Turtle m1 = new Turtle(x+d, y+d, 0.0);
8          javafx.scene.paint.Color c = new javafx.scene.paint.Color(0.8,0.0,0.0,1.0); //赤色
9          m1.setColor(c);       //m1 の色を作成した色オブジェクト) に指定
10         f.add(m);
11         f.add(m1);
12         m.fd(d);
13         m1.fd(d);
14         m.lt(90.0);
15         m1.lt(90.0);
16         d = d / 2;            //d の値を d/2 に変更
17         m.fd(d);
18         m1.fd(d);
19         m1.moveTo(m);
20     }
21 }
```

- 整数の型 int, 浮動小数点の型 double などは，クラス型ではなく**プリミティブ型**という．プリミティブ型の値 (☆ 4) は，オブジェクトではない．
- 前の例 (T21) の 4 行目の TurtleFrame() もコンストラクタ．p12 の API 仕様を読むとわかるように，このクラスには「引数なし」,「引数 2 つ」の 2 種類のコンストラクタが用意されている．このように Java では，**同じ名前だけど引数の数や型が異なる**コンストラクタやメソッドを複数定義することができる．これを**多重定義（オーバーロード）**という．

☆ 4) プリミティブ値という．300 や 3.14 はプリミティブ値である．プリミティブ型/値のことを基本型/値ということもある．

primitive a. 原始的な，基礎的な; construct v. 組み立てる，建造する; overload n. 一般的な用法では，過負荷．プログラミング言語の世界では，多重定義

★ 2.6 値を返すメソッドとインスタンス変数 (p.18)

T23.java

```
 1  import tg.*;
 2  import javafx.scene.paint.*;
 3  public class T23 {
 4      public static void main(String[] args){
 5          double d = 100, x, y, a;
 6          TurtleFrame f = new TurtleFrame();
 7          Turtle m = new Turtle(200, 300, 0);
 8          f.add(m);
 9          m.fd(d);

10          x = m.getX();                              // m の X 座標のとり出し



11          y = m.getY();                              // m の Y 座標のとり出し
12          a = m.getAngle() - 45;                     // m の角度のとり出し
13          Turtle m1 = new Turtle(x, y, a);  // m1 の作成
14          f.add(m1);
15          m1.fd(d);
16          Turtle m2 = m.clone();            //m2 の作成




17          f.add(m2);
18          m.rt(45);
19          m.fd(d);
20          double newscale = m2.tScale * 4;  // m2 の tScale の 4 倍の数
21          m2.tScale = newscale;             // m2 の tScale に代入



22          m2.tColor = new Color(0.0, 1.0, 1.0, 1.0);  // m2 の亀の色を水色に変える



23          m2.fd(d);
24          Point p = f.getMousePosition();



25          m2.moveTo(p.x, p.y);
26      }
27  }
```

- **インスタンス変数**の他に，**クラス変数**というものもある（次回）．インスタンス変数をインスタンスフィールド，クラス変数をクラスフィールドといい，両者まとめてフィールドということもある．

★ 2.7　インポート宣言 (p.17)

T22.java の 8 行目では，色を表すために javafx.scene.paint.Color クラスを利用している．変数 c の宣言でも，初期化のためのコンストラクタ呼び出しでも，クラス名を指定する箇所に javafx.scene.paint.Color と書いている．一方，T23.java の 22 行目でも同様のことをしているが，こちらではクラス名を Color と書くだけで済ませることができている．後者で楽ができている秘密は，2 行目の

```
import javafx.scene.paint.*;
```

という文にある．詳しくは，教科書 p.17,18 参照．同様に，

```
import tg.*;
```

という文は，Turtle や TurtleFrame を tg.Turtle や tg.TurtleFrame と書かずに済ませるために書いている．