目次

★2　オブジェクトの生成とメソッド呼び出し（承前）

「承前」って？ ⇒ 辞書引きましょう

★ 2.7　オブジェクトの参照とヒープ （注:本当の節番号は 2.8 です）

Java の変数の型は，**参照型**（クラス型を含む）と**プリミティブ型**に大別できる．この 2 つの型では,「変数への代入」の意味が異なる．右のプログラムの動作を考えてみよう．

このプログラムで観察できるような参照型とプリミティブ型の違いは，それらの値の記憶のされ方から理解できる．プリミティブ値はその変数用の記憶場所にそのまま格納されるのに対して，オブジェクトはすべて「**ヒープ**」という領域に置かれ，変数には「そのオブジェクトがヒープ上のどこにあるか」を表す情報（これを「**参照**」という）が格納される．そのため，右記で `m1 = m` という代入文を実行すると，変数 `m1` は `m` と同じオブジェクトを「参照している（指している）」ことになる．

プリミティブ型と参照型

```
1  // プリミティブ型の場合
2  int x, y;
3  x = 50;
4  y = x;
5  x += 50;
6  //参照型（オブジェクト）の場合
7  Turtle m, m1;
8  m = new Turtle();  frame.add(m);
9  m1 = m;
10 m.fd(100);
11 m1.bk(100);
```

★ 2.8　インスタンス変数 （注:本当の節番号は 2.9 です）

Turtle クラスのインスタンス（個々のかめたち）は，自分の位置，向き，色や大きさなどの情報を持っている．インスタンス毎に保持されるこのような情報は，**インスタンス変数**と呼ばれる変数に格納されている (☆ 1)．T23 の例（p.18,20）で考えてみよう．

☆ 1) 詳しくは先の授業で説明するが，存在していてもユーザには公開されない（API に示されない）インスタンス変数もある．例えば Turtle クラスの API(p.13) には，インスタンス変数として色 (`tColor`) と大きさ (`tScale`) のみが示されているが，実際にはかめの位置や向きなどを表すインスタンス変数も存在している．

★3 処理の流れ

★3.1 for 文による繰返し (p.24)

C 言語同様に for 文が使える: `for(` 初期化式 `;` 繰返し条件式 `;` ループの更新式 `)` 繰り返す文

- 「条件式」には boolean 型の式を書く（詳細は後の節を参照）.
- 「初期化式」に変数宣言を書くこともできる → 以下の Q 参照

for 文を使った例 T31.java

```
 1  import tg.*;
 2  public class T31 {
 3      public static void main(String[] args){
 4          TurtleFrame f = new TurtleFrame();
 5          Turtle m = new Turtle();
 6          f.add(m);
 7          int i;
 8          for(i = 0; i < 5; i++){
 9              m.fd(100);
10              m.rt(360.0/5);
11          }
12      }
13  }
```

```
(1) i ← 0

(2) i < 5 ?   yes
(4) 実行 ( このとき i は 0 )
(3) i++ ( i ← 1 )

(2) i < 5 ?   yes
(4) 実行 ( このとき i は 1 )
(3) i++ ( i ← 2 )

          :

(2) i < 5 ?   yes
(4) 実行 ( このとき i は 4 )
(3) i++ ( i ← 5 )

(2) i < 5 ?   no
```

Q1. T31 を実行中，`i=2` のときに 10 行目の処理を行ったあと，かめはどこにいる？ 向きは？

Q2. T31 を次のように書き換えるとどうなるか，それぞれ試しなさい (☆ 2).

for ブロックの外で i を宣言

```
 7    int i;
 8    for(i = 0; i < 5; i++){
 9       m.fd(100);
10       m.rt(360.0/5);
11       System.out.println("i = " + i);
12    }
13    System.out.println(
         "for 文終了後の i の値は " + i);
```

for 文中で i を宣言

```
 7    //int i;
 8    for(int i = 0; i < 5; i++){
 9       m.fd(100);
10       m.rt(360.0/5);
11       System.out.println("i = " + i);
12    }
13    System.out.println(
         "for 文終了後の i の値は " + i);
```

Q3. p.27 の練習問題 3.4 をやろう（以下を書きかえよう）.

P32.java の一部

```
for(int i = 0; i < 180; i++){
  m.fd(50);
  m.rt(50);
}
```

☆ 2) 11,13 行目でダブルクォーテーションに囲まれた文字列の後に + 記号が続くという，C 言語では見慣れない書き方が出てきますが，これについては授業中に簡単に説明します．詳しくは次回以降学びます (p.43).

★ 3.2 繰返しの繰返し（ネスト） (p.28)

for 文の「繰り返す文」の所にまた for 文を書けば，繰返しを繰り返すことができる（これを繰返しの**ネスト**という）．for 文に限らず，様々な繰返し文でネストできる（当然，for 文の中に while 文を入れたりしても構わない）．また，ネストは繰返しに限らない（if 文のネストなど）．

Q4. p.28 の T33.java の 9 行目と 10 行目の間に `System.out.println("i = " + i + ", j = " + j);` という文を挿入すると，どんな出力が得られますか．

★ 3.3 while 文による繰返し (p.30)

while 文もあるよ． `while(` 繰返し条件式 `)` 繰り返す文

Q5. p.28 の Sum31.java と同じものを while 文で書いてみよう．

Sum31.java の一部

```
int sum = 0;
for(int i = 1; i <= 10; i++)
  sum += i;
System.out.println(sum);
```

Q6. 上記をもとに while 文の条件を書きかえて「和がはじめて 20 を超えたときの和の値を表示する」ようにしてみよう（ヒント：和が 20 以下の間は繰り返す）．

Q7. 以下の二つはどう違うでしょう？

while の例

```
int i = 10;
while(i < 3){
  System.out.println("ほげ");
  i++;
}
```

do–while の例

```
int i = 10;
do{
  System.out.println("ほげ");
  i++;
}while(i < 3);
```

★ 3.4 boolean 型と論理演算子 (p.30)

C 言語では，論理値（真 (true) と偽 (false)）を表すのに整数値を用い,「0 以外の数は真」を,「0 は偽」を表した．しかし，Java では論理値用の型として **boolean 型**が用意されている．boolean 型の値は true, false の 2 つのみである．

- C 言語と同様に，数値同士の比較などのために次のような演算子が使える．`>`，`<`，`>=`，`<=`，`==`，`!=` （詳細は p.31 参照）
- `==`と`!=`は，boolean 型や，参照型などでも使える．参照型の場合，両辺が同一のオブジェクトを指しているかどうかを調べることになる (☆ 3)．
- 文字列同士が等しいかどうか調べる場合は，String クラスのインスタンスメソッド `equals(String str)` を用いる (☆ 4)．

また，C 言語同様に，論理積，論理和や否定の演算子も存在する（p.31 参照）．

☆ 3) 例えば,
```
Turle a = new Turtle();
Turle b = new Turtle();
```
なら，`a == b` は `false`,
```
Turle a = new Turtle();
Turle b = a;
```
なら，`a == b` は `true`.

☆ 4) 文字列の扱いについては次回以降に学びます.

boolean 型を使った例

```
 4  boolean a, b;  // boolean 型の変数の宣言
 5  int x = 9, y = 1;
 6
 7  a = false;  // boolean 型は true, false の 2 つの値のみ
 8  a = 3;   // こんなことすると...
 9  System.out.println(a);
10
11  // if 文の条件に は boolean 型の値をとる式を書く
12  if(true)  System.out.println("ほげ");
13  if(a)     System.out.println("ほげほげ");
14  if(!a)    System.out.println("ほげほげほげ");
15  if(x > 0) System.out.println("ふが");
16  b = x <= 0;
17  if(b)     System.out.println("ふがふが");
18  b = !(x != 9);
19  if(b)     System.out.println("ふがふがふが");
20
21  // Q8
22  if(x+3 > 0 && x*x < 100) System.out.println("うん，そう");
23  else                     System.out.println("ちゃうちゃう");
24
25  // while 文等も同様　それぞれ 1 行だけコメントをはずしてみよう
26  while(true) System.out.println("HOGE");
27  //while(false) System.out.println("HOGE");
28  //while(1) System.out.println("HOGE");
```

Q8. 上記の Q8 と書かれたコメントの箇所に書かれた条件式（if 文の括弧内）は，以下の 1. を boolean 型の式で表したものになっている．これを参考に，2. 以下を変数 `x,y` を用いた条件式で表しなさい.

1. $x+3$ が 0 より大きくかつ x^2 が 100 未満である
2. x, y のいずれかが 1 である
3. x は 0 以上 10 未満である
4. $0 \leq x < 10$ または $20 \leq x < 30$
5. x は 3 以外の 3 の倍数である

★ 3.5 if 文 (p.32)

if 文の書き方，使い方は C 言語と同様．ただし，条件式は boolean 型.

```
if( 条件式 ) 条件成立時だけ行う文
```

```
if( 条件式 ) 条件成立時に行う文
else 条件不成立時に行う文
```

Q9. 以下を実行するとどんな結果が得られますか.

if–else の例

```
for(int i = 0; i < 9; i++){
  System.out.print(i);
  if(i % 3 == 1)
    System.out.println(" (^^)/");
  else
    System.out.println("");
}
```

注 1: println("") は，改行だけする．引数なしでも同じ.
注 2: Q10 の例では,「条件成立時に行う文」も「条件不成立時に行う文」も一文なので{,} で囲んでいない．しかし，このようにしていると，文を追加する際に{,}も付け足すのを忘れてコンパイルエラーやバグを生みやすい．Q10 の例のように，一文しかなくても{,}で囲んでおく方がよいだろう.

Q10. 以下を実行するとどんな結果が得られますか.

else–if の例 (T37 改)

```
for(int i = 0; i < 12; i++){
  if(i % 4 == 0){
    m.setColor(Color.RED);
  }else if (i % 4 == 1) {
    m.setColor(Color.GREEN);
  }else{
    m.setColor(Color.BLUE);
  }
  m.lt(30);
  m.fd(50);
}
```

```
if( 条件式 A ) 文 a
else if( 条件式 B ) 文 b
else if( 条件式 C ) 文 c
...
else 文 n
```

Q11. 以下の if 文を同じふるまいをするのは (ア),(イ) のどちらか.

else–if の例 2

```
if(i % 4 == 0){
   System.out.println(i + " ほげ");
}else{
   if(i % 4 == 1){
      System.out.println(i + " ふが");
   }else{
      System.out.println(i + " へな");
   }
}
```

(ア)

```
if(i % 4 == 0) System.out.println(i + " ほげ");
if(i % 4 == 1) System.out.println(i + " ふが");
else           System.out.println(i + " へな");
```

(イ)

```
if(i % 4 == 0)      System.out.println(i + " ほげ");
else if(i % 4 == 1) System.out.println(i + " ふが");
else                System.out.println(i + " へな");
```