CS 知的システム演習

Java言語の復習

松吉 俊

Java言語の復習

- クラスの構成
- 配列の利用方法
- Collectionクラスの利用方法

クラスの構成

- 1つのクラスは、3つのパートから構成される

  フィールド: クラスで利用する**変数**を定義する

  メソッド: クラスで利用する**関数**を定義する

  コンストラクタ: クラスのインスタンスを生成するとき
  (new ○○) の処理を定義する

- 例えば、

  MazeModelクラスのフィールドに何がありますか?

  MazeDataクラスのメソッドには何がありますか?

  Robotクラスのフィールドには何かありますか?

配列の宣言: 1次元

- Javaでの配列の宣言は、C++とは異なる

例：1次元配列の宣言と利用

```
// int 型の配列(サイズ100)の宣言
int[] array = new int[100];

// 配列の各要素を初期化する例
for (int i=0; i < array.length; i++)
  array[i] = 0;
```

配列 array のサイズを表す

配列の宣言: 2次元

- Javaでの配列の宣言は、C++とは異なる

例：2次元配列の宣言と利用

```
// int 型の2次元配列の宣言
int[][] array = new int[100][200];

// 配列の各要素を初期化する例
for (int i=0; i < array.length; i++)
  for (int j=0; j < array[i].length; j++)
    array[i][j] = 0;
```

配列 array[i] のサイズを表す

コレクションクラス

- Javaには、オブジェクトの集まりを扱うための
  クラス群が豊富に用意されている

代表的なクラス

- LinkedList　　連結リストクラス
- ArrayList　　可変長配列クラス

配列と異なり、サイズが固定長でないので使いやすい
（必要に応じてサイズが自動的に増える）

LinkedList の使用例

```
import java.util.*;  // コレクションクラスを利用する場合に必要
  :

LinkedList list = new LinkedList(); // 空のリストの生成

list.add(“We”);                  // リストに “We” を追加
list.add(“love”);                // リストに “love” を追加
list.add(“music”);               // リストに “music” を追加

System.out.println(list);        // リストを表示
                                 //  [We, love, music]

list.removeFirst();              // 先頭要素を削除
System.out.println(list);        // リストを表示
                                 //  [love, music]

list.removeLast();               // 末尾要素を削除
System.out.println(list);        // リストを表示
                                 //  [love]
```

利用可能なクラスライブラリ

- API ドキュメントを参照
  - http://docs.oracle.com/javase/jp/7/api/
  - Java言語で利用可能な全てのクラスのドキュメント
  - Javaプログラミングでは非常に役立つ

変数の型

- Javaの変数の型は大きく2種類
  - 基本型
    - int、char、float、double など
  - オブジェクト型
    - LinkedList、ArrayList、Maze、MazeModel など

コレクションクラスの注意点

- コレクションクラスは、
  オブジェクトの集まりを表現するためのクラス
- つまり、基本型を登録できない

```
LinkedList list = new LinkedList(); // 空のリストの生成

list.add(10);          // コンパイルエラー: int はオブジェクト型ではない
list.add(3.14);        // コンパイルエラー: double はオブジェクト型ではない
```

自作のクラスは登録可能 (1)

```
// 座標 (x, y) を表すクラス
class Position {
    // コンストラクタ
    public Position(int x, int y) {
        this.x = x;
        this.y = y;
    }

    // アクセスメソッド
    public int getX() { return x; }
    public int getY() { return y; }

    // オブジェクトの内容を表示するためのメソッド
    public String toString() {
        return "(" + x + “," + y + ")";
    }

    // インスタンスフィールド
    private int x;
    private int y;
}
```

自作のクラスは登録可能 (2)

```
LinkedList list = new LinkedList();  // 空のリストの生成

list.add(new Position(10, 20));      // リストに (10, 20) を追加
list.add(new Position(30, 40));      // リストに (30, 40) を追加
list.add(new Position(50, 60));      // リストに (50, 60) を追加

System.out.println(list);            // リストを表示
                                     //  [(10,20),(30,40),(50,60)]

list.removeFirst();                  // 先頭要素を削除
System.out.println(list);            // リストを表示
                                     //  [(30,40), (50,60)]

list.removeLast();                   // 末尾要素を削除
System.out.println(list);            // リストを表示
                                     //  [(30,40)]

Position p = (Position)list.get(0);  // 0番目の要素を取得(キャストが必要)
System.out.println("p = " + p);      // p = (30,40) と表示される
```

基本型もラッパークラスを利用すれば登録可能

```
LinkedList list = new LinkedList();  // 空のリストの生成

list.add(new Integer(10));           // リストに 10 を追加
list.add(new Integer(20));           // リストに 20 を追加
list.add(new Integer(30));           // リストに 30 を追加
                                     <- int 型変数を1つ保持するだけのラッパークラス
System.out.println(list);            // リストを表示
                                     //  [10, 20, 30]

list.removeFirst();                  // 先頭要素を削除
System.out.println(list);            // リストを表示
                                     //  [20, 30]

list.removeLast();                   // 末尾要素を削除
System.out.println(list);            // リストを表示
                                     //  [20]
// 0番目の要素を取得(キャストが必要)
int n = ((Integer)list.get(0)).intValue();
System.out.println("n = " + n);      // n = 20 と表示される
```

- Special thanks:
  - 山本 泰生先生
  - 鍋島 英知先生