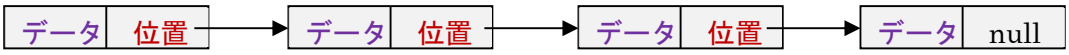
2024 年 1 月 18 日（木）実施

応用プログラム（2） ― リスト構造―

リストとは，関係付けられたデータを順番に並べたものである。ここでは，基本的なデータ構造として連結リストと呼ばれるリスト構造を扱う。連結リストのうち，最も単純な構造は，次の図で表される単方向連結リストである。各オブジェクトはデータと共に，直後に続くオブジェクトへのリンク情報を備えている。リスト構造の特長は，並び順の途中でオブジェクトの追加や削除が容易なことである。



この構造は，Java 言語ではコレクションと呼ばれる，オブジェクトの集合を扱うためのクラスのうち，LinkedList クラスを用いて実現される。LinkedList クラスは，各オブジェクトが前後のオブジェクトへのリンク情報を備える双方向連結リストにも対応している。

LinkedList クラスには引数無しのコンストラクタが定義されていて，次の様にインスタンスを生成する。

```
LinkedList<データ型> 変数名 = new LinkedList<データ型>();
```

例）LinkedList<String> llist = new LinkedList<String>();

Linkage クラス

Eclipse で jimbo の様に『自分の名前』のパッケージを指定して，次の Linkage クラスを作成せよ。（これは直接実行出来ないクラス ⇒ 後の例題及び演習に利用する）

```
package jimbo;

import java.util.LinkedList;

public class Linkage {
    LinkedList<String> llist = new LinkedList<String>();

    Linkage(String[] str0) {
        initList(str0);
    }

    void initList(String[] str) {
        for (int i=0; i<str.length; i++)
            llist.add(str[i]);
    }

    void addN(int n, String strn) {
        llist.add(n, strn);
    }

    void addF(String strf) {
        llist.addFirst(strf);
```

```
    }

    void addL(String strl) {
        llist.addLast(strl);
    }

    void removeN(int n) {
        llist.remove(n);
    }

    void removeF() {
        llist.removeFirst();
    }

    void removeL() {
        llist.removeLast();
    }
}
```

【解説】

1. Linkage クラスでは，コンストラクタから initList メソッドを呼び出して連結リスト llist の初期化を行う。
2. コンストラクタでは文字列の配列を引数として受け取り，それを initList メソッドに受け渡し，LinkedList クラスの add メソッドによって順番に連結リストに付加していく。
3. add メソッドは引数の個数の異なるものが重複して定義(**オーバーロード**)されており，add(n, strn)は，n 番目の位置に文字列 strn を挿入する。なお，n は 0 番から始まる。
4. remove メソッドは連結リストからオブジェクトのリンクを削除するためのもので，remove(n)は n 番目の位置にあるオブジェクトを連結リストから外す。
5. addFirst メソッド，removeFirst メソッドは連結リストの先頭のオブジェクトに対応し，addLast メソッド，removeLast メソッドは連結リストの末尾のオブジェクトに対応する。

例題 1

次のプログラムを入力し，ビルドして，実行せよ。ここで，クラス名は Sample12_1，ソースファイル名は Sample12_1.java とする。

```
package jimbo;

public class Sample12_1 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        String[] country = {"日本", "中国", "韓国", "インド"};

        Linkage linked = new Linkage(country);
        System.out.println(linked.llist);

        linked.addN(3, "ベトナム");
```

```
        System.out.println(linked.llist);

        linked.addN(3, "ネパール");
        System.out.println(linked.llist);

        linked.addF("米国");
        System.out.println(linked.llist);

        linked.removeN(0);
        System.out.println(linked.llist);

        linked.addL("オーストラリア");
        System.out.println(linked.llist);

        linked.removeL();
        System.out.println(linked.llist);

    }

}
```

【解説】llist は Linkage クラスのインスタンス linked のフィールド値を参照する。

例題 2

次のプログラムを入力し，ビルドして，実行せよ。ここで，クラス名は Sample12_2，ソースプログラム名は Sample12_2.java とする。

```
package jimbo;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Sample12_2 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        String[] friend = {"鈴木", "佐藤", "田中", "斉藤"};
        final int NUM = 5;
        String str = null, str2 = null;
        int n = 0;
        Scanner sc = new Scanner(System.in);

        Linkage linked = new Linkage(friend);

        System.out.println("初期のリスト");
        System.out.println(linked.llist);

        for (int i=0; i<NUM; i++) {
            try {
                System.out.println("");
                System.out.print("友人の名前を入力してください：");
                str = sc.next();
```

```
                System.out.print("何番目に加えるかを入力してください：");
                n = sc.nextInt();

                linked.addN(n, str);
            } catch (InputMismatchException exim) {
                str2 = sc.next();
                System.out.println(str2+"は整数ではありません。");
                System.out.println("");
                i--;
            } catch (IndexOutOfBoundsException exiob) {
                System.out.println("要素数を上回りました。"+exiob);
                System.out.println("");
                i--;
            }
        }

        System.out.println("");
        System.out.println("データ追加後のリスト");
        System.out.println(linked.llist);
    }

}
```

【解説】リストにデータを追加する位置は先頭から 0 番，1 番，・・・となるが，最後尾のデータの 2 個先を指定すると例外が起こる。そこで，IndexOutOfBoundsException クラスでその例外をキャッチする。

演習

次のプログラムリストの空欄 1)　　～3)　　に適切な語句を埋めたプログラムを作成し，ビルドして実行せよ。ここで，プログラムのクラス名は Ex12，ソースプログラム名は Ex12.java とする。

```
package jimbo;

import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Ex12 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        String[] 1)     = {"スマートフォン", "タブレット"};
        final int NUM = 5;
        int pos = prod.length;
        String str = null;
        Scanner sc = new Scanner(System.in);
        String fname = "outL.txt";
```

```
        Linkage linked = new Linkage(prod);

        System.out.println("初期のリスト");
        System.out.println(linked.2)     );

        try {
            PrintWriter pw = new PrintWriter(new BufferedWriter(new
                             FileWriter(fname)));

            for (int i=0; i<NUM; i++, pos++) {
                System.out.println((i+1)+"個目の");
                System.out.print("  商品名を入力してください：");
                3)     = sc.next();

                linked.addN(pos, str);
            }

            System.out.println("");
            System.out.println("データ追加後のリストをファイルに書き出します。");

            pw.println(linked.llist);
            pw.close();
        } catch (FileNotFoundException fe) {
            System.out.println(fname +"というファイルが開けません");
        } catch (IOException err) {
            System.out.println("IOException をキャッチ"+err);
        }
    }

}
```

【解説】for 文の再初期化で i++, pos++とあるのは，繰り返し処理の度に i 及び pos のそれぞれを 1 ずつ増加させることを表す。

提出物：

1) 例題 1 及び例題 2 のプログラムのコンソールへの出力結果をコピーして貼り付けたテキストファイル res12.txt をメールに添付する。
2) 演習で空欄に適切な語句を埋めたソースプログラムのファイル Ex12.java をメールに添付する。
3) 演習で出力されたファイル outL.txt をメールに添付する。
4) 第 12 回理解度確認用の質問ファイル Prog1_Questions_12th.txt に解答を記入して，メールに添付する。

＊ メールの件名は『プログラミング 1　第 12 回課題』（鍵括弧は要らない）とする。