プログラミングIII

黒瀬　浩

kurose@neptune.kanazawa-it.ac.jp

OH: 講義の前後，火4限21・405

居室　67・121

ArrayList

配列は追加や削除が動的に行えない

作成　ArrayList<型> 変数名 = new ArrayList<型>();

メソッド

追加　add(型　値)

取得　get(int 要素番号)

削除　remove(int 要素番号)

要素数 size()

リスト5-1抜粋

```
ArrayList<String> months = new ArrayList<String>(); // []
months.add("Jan");        // ["Jan"]
months.add("Feb");        // ["Jan", "Feb"]
months.add("Mar");        // ["Jan", "Feb", "Mar"]
months.remove(1);         // ["Jan", "Mar"]
months.size()      // 2
```

ラッパークラス　基本型を包むクラス

基本型	ラッパークラス
boolean	Boolean
byte	Byte
char	Charcater
short	Short
int	Integer
long	Long
float	Float
double	Double

既に出てきた　Integer.parseInt()　はintラッパークラスのメソッド

List5-2抜粋

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(new Integer(50)); // Integerクラスに50を渡す
Integer interger0 = integerList.get(0);    // 型はInteger
int i0 = integer0.intValue();              // 整数化
```

コレクションフレームワーク

リスト

- 要素の順番を管理する
- 要素の取り出し，追加，削除
- `ArrayList, LikedList`

マップ

- キーと値(Key Value)
- 他言語のハッシュ，辞書と同じ
- `HashMap, LinkedHashMap`

セット

- 要素に重複が起きない
- 他言語のセット，集合と同じ
- `HashSet, TreeSet`

`LinkedList`は，要素から次の要素の位置と前の位置を管理している
前からたどる場合，後ろからたどる場合，要素の追加，要素の削除で高速

HashMapの例

```
HashMap<String, String> map = new HashMap<String, String>();
map.put("key1", "val1");     // 要素の追加
map.put("key2", "val2");
map.entrySet();          // キーと値のペアの一覧取得
// printすると  [key1=val1, key2=val2] 順番は保証されない

map.values();            // 値一覧取得
// printすると  [val1, val2]  順番は保証されない

map.keySet();            // キー一覧取得
// printすると  [key1, key2]  順番は保証されない

map.get("key1");         // キーがkey1の値を取得, キーがなければnull
// printすると  val1
```

値を指定してキー一覧を得たい場合は？
既に存在するキーに対してput()すると？

イテレータ(Iterator) 反復子

一つづつとり出す

```
HashSet<String> set = new HashSet<String>();
set.add("A");
set.add("B");
Iretator<String> it = set.iterator();
while(it.hasNext()){
	String str = it.next();
	System.out.println(str);
}
```

キーボードから読む

```
Scanner sc = new Scanner(System.in);
sc.next();  // イテレータと同じ
```

拡張for文 ひとつづつ取り出して処理する

```
for(型　変数 : コレクション){　　}
```

pythonのfor文

```
for 変数　in コレクション:
    ブロック
```

phpのforeach文

```
foreach ( 変数　as 配列 ){　　}
```

List5-7抜粋改

```
String [] months={"Jan", "Feb", "Mar"}
for(String str : month){
    System.out.println(str);
}
```

キュー，スタック

キュー
　　末尾に追加，最初から取り出し(先入先出し,FIFO)

スタック
　　末尾に追加，末尾から取り出し(後入先出し,LIFO)

リストを使えばキュー，スタックを実装できる

キュー：　　待ち行列，構造探索
スタック：　演算の計算，関数呼び出しの管理，構造探索，章番号付け

グラフ（点と線からなる構造）
　　探索するときに再帰呼び出しやキュー，スタックを使う

高階関数(higher-order function)

関数に関数を渡したい場合がある

```
# pythonの例
import math as M   # 数学パッケージインポート
def f(g, p):       # 関数定義
    return g(p)    # 引数gを関数として実行

print(f(M.sin, M.pi))    # 関数fにsin()とM.piを渡す
print(f(M.cos, M.pi))    # 関数fにcos()とM.piを渡す
```

結果

```
1.2246467991473532e-16   # sin π
-1.0                     # cos π
```

オブジェクト指向では，関数，クラス，変数，メソッド，定数もオブジェクトとして扱いたい（全てがオブジェクト）

多くの言語でmap関数，filter関数, reduce関数には関数と対象を渡す

内部クラス リスト6-1

クラスの中にクラスを定義する

```
class Outer {
    private String message = "*Outer";
    void doSomething(){
        class inner {
            void print(){
                System.out.println("*Inner");
                System.out.println(message); // class外
            }
        }   // doSomething()の中でインスタンスを作成している
        Inner inner = new Inner();
        inner.print();
    }
}
public class InnerClassExample {
    public static void main(String[] args){
        Outer outer = new Outer();
        outer.doSomething();
}
```

匿名クラス　名前のないクラス List6-1から

```
interface SayHello {
   public void hello(); // 抽象メソッド
}
class Greeting {
       // SayHelloインターフェースを実装するクラスを引数として受け取る
   static void greet(SayHello s){  // (2) 渡されたpをsに
       s.hello();           // (3) p.hello()と同じ
   }
}

class Person implements SayHello {
    public void hello(){       // (4) 最終的にここが呼ばれる
        System.out.println("Hello");
    }
}
public class SimpleExampe {
    public static void main(String[] args){
        Person p = new Person();
        Greeting.greet(p);   // (1) Person型のpインスタンスを渡す
    }
}
// クラスを限定せずにそのクラスに応じたメソッドを呼び出す
```

ラムダ式

```
(int n) -> { return n+1; }
```
整数nを受け取りn+1を返す

```
(int a, int b) -> { return a+b; }
```
整数a,bを受け取りa+bを返す

引数の型は省略できる

```
(n) -> { return n+1; }
```
nを受け取りn+1を返す

引数が1つなら()を省略できる

```
n -> { return n+1; }
```
nを受け取りn+1を返す

実施する文が1つなら { return ;}を省略できる

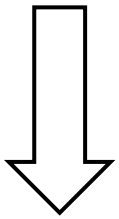
```
n -> n+1;
```
nを受け取りn+1を返す

引数がない場合

```
() -> System.out.println("こんにちは");
```

コレクションフレームワークとラムダ式 list6-6抜粋

```
for(Point p : pointList){
    p.x *= 2;
    p.y *= 2;
}
for(Point p : pointList){
    p.printInf();
}
```



```
pointList.forEach( p -> { p.x *= 2; p.y *= 2; } );
pointList.forEach( p -> p.printInfo() );
```

演習

5章の例題
5-01 ArrayListの操作
5-02 ラッパークラスの例
5-03 オートボクシング，オートアンボクシング
5-04 マップコレクション
5-05 セットコレクション
5-06 イテレータ
5-07 拡張for文
5-08 キュー
5-09 スタック
5-10 ソート

演習

6章の例題
6-01 内部クラス (6-03まで)
6-04 匿名クラス
6-05 ラムダ式の例
6-06 コレクション操作の例
　　forEachでの書き換え
　　ソート