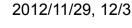
発展プログラミング演習II

9. クラスライブラリの使い方 ~コレクションとアルゴリズム~

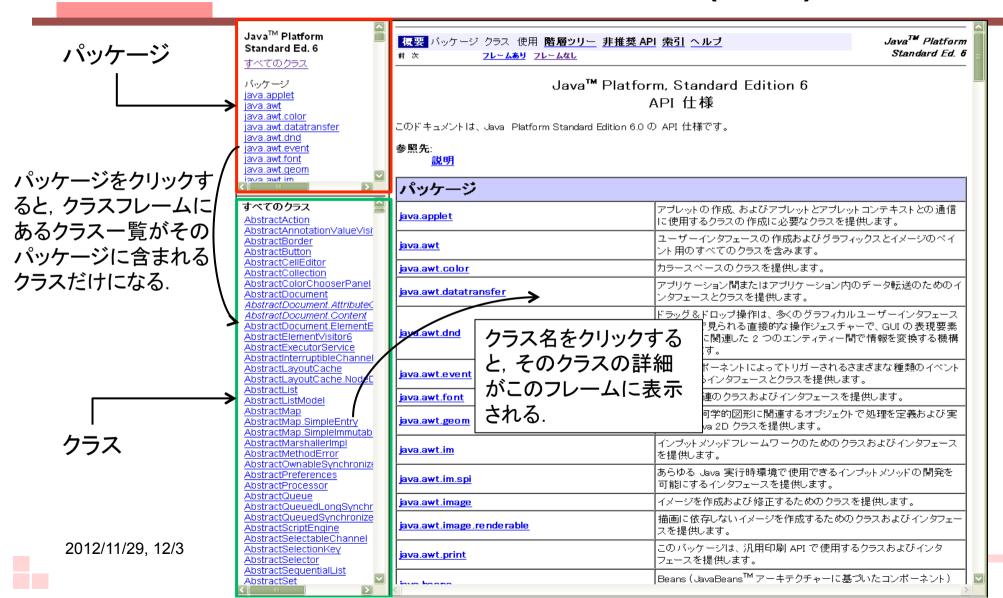
玉田 春昭 水口 充

クラスライブラリについて

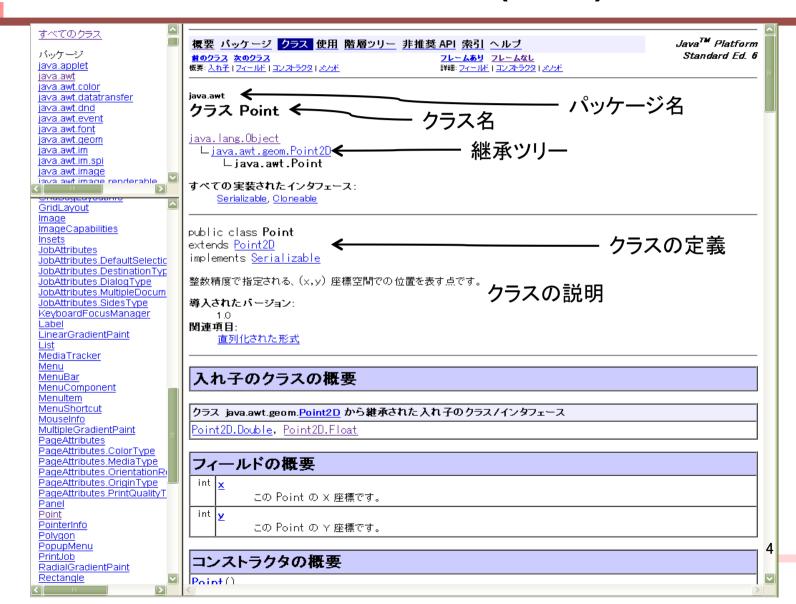
- Javadocの読み方
- コレクションとアルゴリズム
 - コレクション
 - アルゴリズム
- 入出力



Javadocの読み方 (1/3)

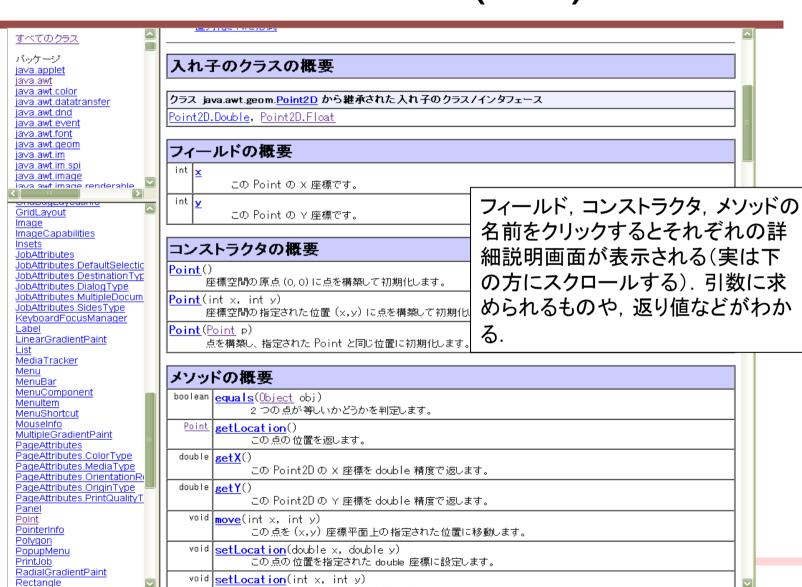


Javadocの読み方 (2/3)



2012/11/29, 12/3

Javadocの読み方 (3/3)



2012/11/29, 12/3

コレクション (1/2)

- Java言語ではデータ構造のためのフレームワークを持っている。
 - 配列より、より高度なデータ構造を利用できる.
 - クラスがあらかじめ用意されている.
 - 色々なデータ構造を統一的に扱える.

コレクション (2/2)

- 集合を表すSet,リスト構造を表すList,キーと値の ペアを扱うMap.
 - Java SE 6からは、これに加えてDeque(デック)が追加されている.
- 実装と構造名が分かれている.
 - 実装+データ構造がクラス名になっている.

		·····································				
		配列	リンクリスト	ツリー構造	ハッシュテーブル	ハッシュテーブル +リンクリスト
構造名	Set			TreeSet	HashSet	LinkedHashSet
	List	ArrayList	LinkedList			
	Мар			TreeMap	HashMap	LinkedHashMap

java.util.Collectionインター フェース

- コレクションフレームワークの中核となるインターフェース.
- コレクションを表す。
 - コレクションとは「要素」であるオブジェクトを束ねるグループのこと。
 - コレクションインターフェースができること.
 - 特定要素の追加(add), 削除(remove), 検索(contains).
 - コレクション同士のマージ(addAll), 全要素の列挙(iterator), 全要素の全削除(clear).
- このインターフェースを使う直接のクラスは存在しない。
 - サブインターフェースを介したクラスしか提供されていない.

- java.util.List**はリスト構造を表すインター** フェース.
 - 配列で実装するArrayListとリンク構造で実装する LinkedListがある.
 - どちらも使い方は一緒.

値を代入する変数の型は抽象的なインターフェースとする.

生成するクラスは具象クラス.

```
List list = new ArrayList();
list.add("first");
list.add("second");
list.add("third");
String value = (String)list.get(1); // "second"が返される.
```

例題9.1 ArrayListを使う

- 右のプログラムを作成 し、実行結果を確かめ よ。
 - Moodleにもプログラムを掲載している.
 mainメソッドがわからない場合はダウンロードして確かめること.
- コンパイル時に出力される警告(warning)は無視して構わない。

```
import java.util.*;
public class ArrayListDemo{
  public void run(){
    List list = new ArrayList();
    list.add("one");
    list.add("two");
    list.add("three");
    list.add("four");
    list.add("five");
    System.out.println(list);
    list.add(1, "TWO");
    System.out.println(list);
    String removedObject = (String)list.remove(2);
    System.out.println(list);
  // mainメソッドは省略
```

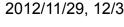
例題9.2 ArrayListの 中身を列挙する

- 右のプログラム
 ArrayListDemo2を 作成せよ。
- 入力できれば、実行結果を確かめよ。

```
import java.util.*;
public class ArrayListDemo2{
  public void run(){
    List list = new ArrayList();
    list.add(new Integer(5));
    list.add(new Float(-14.14f));
    list.add("Hello");
    list.add(new Long(120000000));
    list.add(new Double(-23.45e-11));
    printList(list);
    list.add(1, "String to be inserted");
    printList(list);
    Object removedObject = list.remove(3);
    printList(list);
  private void printList(List list) {
    for (int i = 0; i < list.size(); i++) {
      Object object = list.get(i);
      System.out.print(object + " ");
    System.out.println(list.size());
  // mainメソッドは省略
```

例題9.3 ArrayListとLinkedList

- 例題9.2 (ArrayListDemo2)はArrayListを 使ったプログラムである.
- これをLinkedListを使ったプログラムに書きな おし、実行結果を確かめよ.
- クラス名はLinkedListDemo2とする.
 - ArrayListDemo2をコピーして作成すると簡単.



課題9.1 ArrayListの要素

- 下のプログラムは一部間違いがある。
 - コンパイルエラーが起こらないように修正して、実行結果を確かめよ.
 - 1~10までの合計を計算するプログラム.

```
import java.util.*;
public class ArrayListDemo3_学生証番号{
  public void run() {
    List list = new ArrayList();

  for(int i = 1; i <= 10; i++) {
    list.add(new Integer(i));
  }
  int sum = 0;
  for(int j = 0; j < list.size(); j++) {
    sum += list.get(j);
  }
  System.out.println("Total: " + sum);
  }
  // mainメソッドは省略
}</pre>
```

ヒント

- Listの中には参照型しか入れられない。
- Listに入れている型は何でしょうか.
 - Listから取り出すときは、どのような型で取り出しているでしょうか.
 - この講義資料の9ページ目を参照。
- Listの中身にはObject型として入れられている.
- IntegerからObjectへは自動的に変換してくれるが、逆は自動変換してくれない。
 - 自分でキャストしなければならない.

java.util.Map//ンターフェース

- ・ java.util.Mapはキーと値のペアの集合を扱うためのインターフェース.
 - ハッシュテーブルで実装するHashMapとツリー構造で実装するTreeMapがある.
 - 順序は考慮されない.
 - 追加した順序を保持したいのであれば、LinkedHashMapを用いる.

値を代入する変数の型は抽象的なインターフェースとする.

生成するクラスは具象クラス.

```
Map map = new HashMap();
map.put("first", new Integer(1));
map.put("second", new Integer(2));
map.put("third", new Integer(3));
Integer value3 = (Integer)map.get("third"); // 3
Integer value4 = (Integer)map.get("four"); // null
```

例題9.4 HashMapを使う

- 右のプログラムを作成し、 実行結果を確かめよ。
 - Moodleにもプログラムを 掲載している. mainメソッド がわからない場合はダウ ンロードして確かめること.
- コンパイル時に出力される警告(warning)は無視して構わない。

```
import java.util.*;

public class HashMapDemo{
   public void run(){
     Map map = new HashMap();

   map.put("one", 1); map.put("two", 2);
   map.put("three", 3); map.put("four", 4);
   map.put("five", 5);
   System.out.println(map);

   map.put("one", 10);
   System.out.println(map);

   map.remove("one");
   System.out.println(map);
}
// mainメソッドは省略.
}
```

例題9.5 Mapの内容を列挙する

• 下のプログラムを入力し、実行結果を確かめよ.

```
import java.util.*;
public class HashMapDemo2{
  public void run(){
    Map map = new HashMap();
    map.put("one", 1); map.put("two", 2); map.put("three", 3);
    map.put("four", 4); map.put("five", 5);
    printMap(map);
    map.put("one", 10);
    printMap(map);
    map.remove("one");
    printMap(map);
  private void printMap(Map map) {
    for(Iterator i = map.entrySet().iterator(); i.hasNext(); ){
      Map.Entry entry = (Map.Entry)i.next();
      Object key = entry.getKey();
      Object value = entry.getValue();
      System.out.println(key + " => " + value);
  // mainメソッドは省略
```

課題9.2, 9.3

- 例題9.5 (HashMapDemo2)は追加した順序が無視される. 追加した順序を保持するようにプログラムを変更せよ.
 - ヒント: LinkedHashMapを用いる.
 - クラス名は「LinkedHashMapDemo 学生証番号」とする.
- **例題9.5** (HashMapDemo2)は順序がバラバラである. キーが辞書順に並ぶように、プログラムを変更せよ.
 - ヒント: TreeMapを用いる.
 - クラス名は「TreeMapDemo_学生証番号」とする.

課題9.4 データベースの作成(1/3)

- 名前と電話番号のペアを管理する簡易データベースを作成せよ。
 - 名前, 電話番号の追加, 更新, 検索, 削除, 一覧が可能.
 - 仕様
 - add 名前 電話番号
 - データベースに値を追加する.
 - list
 - 登録された*名前と電話番号*の一覧を表示する。
 - find 名前
 - データベースに*名前*が存在すれば*名前と電話番号*を表示する.
 - データベースに*名前*が存在しなければ何も表示しない.
 - remove 名前
 - データベースから*名前*のデータを削除する.
 - データベースに*名前*が存在しなければ何もしない.
 - update *名前 電話番号*
 - データベースに名前が存在すれば、電話番号を更新する.
 - データベースに*名前*が存在しなければ何もしない.
 - quit
 - 終了する.
 - わからなければ、APIドキュメントを参照すること。

課題9.4 データベースの作成 (2/3)

実行例

```
$ java Database 111111
> list
> add tamada 090-1111-1111
> find minakuchi
> add minakuchi 090-2222-2222
> list
minakuchi 090-2222-2222
tamada 090-1111-1111
> find minakuchi
minakuchi 090-2222-2222
> remove tamada
> list
minakuchi 090-2222-2222
> quit
```

```
$ java Database_111111
> list
> add tamada 090-1111-1111
> update minakuchi 090-2222-1111
> list
tamada 090-1111-1111
> update tamada 090-2222-1111
> list
tamada 090-2222-1111
> list
tamada 090-2222-1111
> remove tamada
> list
> quit
```

- ・赤字部分が入力部分. このような実行結果になるようプログラムを作成すること.
- ・必ず文字の分割を行うこと(add→名前→電話番号のような入力方法はダメ).

課題9.4 データベースの作成 (3/3)

- 課題のヒント
 - Mapを用いる.
 - 入力において、文法エラーは起きないものとする。
 - 標準入力から1行読み込む方法.
 - String line = System.console().readLine();
 - ただし、環境によっては、System.console()メソッドがない場合、 System.console()メソッドがnullを返す場合がある。
 - その場合は、MoodleにあるSimpleConsoleを使うこと.
 - 使い方は以下の通り。
 - String line = SimpleConsole.console().readLine();
 - 1行をスペースで区切る方法
 - String[] strings = line.split(" ");

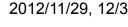
課題9.1, 9.2, 9.3, 9.4

クラス名など

- 課題9.1:ArrayListDemo3 学生証番号
- 課題9.2:LinkedHashMapDemo 学生証番号
- 課題9.3:TreeMapDemo 学生証番号
- 課題9.4: Database 学生証番号
 - 6桁の学生証番号. 最初にgはいらない.
- ソースファイルをMoodleに提出すること。
 - 課題9.4は「Database_学生証番号」のみ提出すること.
 - SimpleConsoleは不要.
- 〆切: 2012年12月6日(木) 9:00まで.

課題のヒント

- 課題9.1:わからなければ以下のことを行うこと.
 - mainメソッドを書く.
 - コンパイルエラーを起こしている個所をコメントアウトする.
- 課題9.2, 課題9.3
 - 例題9.3 (p.12)を参照のこと.
- 課題9.4



ジェネリクス (Generics)

- 汎用的なクラスやインターフェースを特定の型に対応付ける機能。
 - JDK 1.5から導入された.
- List<String>
 - Stringのみを格納できるListの意味.
 - 汎用のListがString 専用になった。
 - Listから値を取り出すと き、キャストが不要になっ た.
 - ジェネリクスを用いない 場合、キャストが必要、

ジェネリクスありの場合

```
List<String> list = new ArrayList<String>();
list.add("abc");

// String型以外を格納しようとしたのでコンパイルエラー
list.add(1);

// listの0番目の値(abc)を取得.

// String型用のlistなので、キャスト不要.

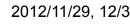
String string = list.get(0);
```

ジェネリクスなしの場合

```
List list = new ArrayList();
list.add("abc");
// ジェネリクスがないので、エラーなし.
list.add(1);
// listの0番目の値(abc)を取得.
// ジェネリクスがないので、キャストが必要.
// キャストがないので、コンパイルエラー.
String string = list.get(0);
```

アルゴリズム (1/2)

- 良く使うアルゴリズムもAPIとして用意されている。
 - 乱数, ソート, 探索
 - java.utilパッケージを参照.
 - 暗号, ハッシュ関数
 - java.security, javax.cryptパッケージを参照
 - 圧縮(zip)
 - java.util.zipパッケージを参照



アルゴリズム (2/2)

- Java言語では以下のアルゴリズムも用意されている。
 - 乱数
 - ・アルゴリズムに線形合同法を採用している.
 - ソート
 - アルゴリズムとして、修正マージソートを採用、安定なソート。
 - 二分探索
 - ソート済みのListに対して二分探索を行う。

乱数

- 疑似乱数生成器としてjava.util.Randomが 用意されている。
 - 線形合同法で生成されている.
 - C言語のrand () 関数と同じアルゴリズム.
 - $x_{n+1} = (a x_n + b) \mod m$

使い方その1

```
Random rand = new Random();
int value = rand.nextInt();
```

使い方その2

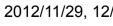
```
Random rand = new Random(System.currentTimeMillis());
double value = rand.nextDouble();
```

例題9.6 乱数

- java.util.Randomクラスを用いて、100回乱数を発生させ、 それを標準出力に出力せよ(System.out.printlnを使って 画面に出力せよ)。
 - 以下の空白部分を埋めよ.
 - Random#nextInt (RandomクラスのnextIntメソッドの意味)を 使って乱数を得る.

java.util.Random

- intやdoubleの乱数が生成できる。
 - Random#nextInt(), Random#nextDouble()な تبل
- 詳細はJavadocを参照のこと.



例題9.7 シードを設定

- 例題9.6 (RandomDemo)を変更し、シードを自分の学生証番号にせよ。
 - シードとは、種とも呼ばれ、乱数発生の元となる。
 - シードが同じであれば同じ乱数列が生成される.
 - 何度か実行し、同じ乱数列が生成されることを確認せよ.
 - 学生証番号が0から始まる場合は、頭に適当な数字を つけること。

ソート

- Collection (List)をソートするには java.util.Collectionsクラスのstaticメ ソッドsortを用いる.
 - アルゴリズムはマージソート.
- 配列をソートするにはjava.util.Arraysクラ スのstaticメソッドsortを用いる.
 - アルゴリズムはObject型のソートはCollections と同じくマージソート, 基本型の配列に対してはクイッ クソート.

2012/11/29, 12/3

例題 9.8 ソート

- ・ 以下の処理を行うプログラムを作成せよ.
 - クラス名はColletionSortDemoとする.
 - 1. ArrayList**のインスタンスを作成し**, List型**の変**数list**に代入せよ**.
 - 2. list**に100個の**int型**の乱数を追加せよ**.
 - 3. list**の中身を画面(標準出力)に表示せよ**.
 - **4.** list**の中身をソートせよ**.
 - 5. list**の中身を画面(標準出力)に表示せよ**.

java.util.Comparator インターフェース

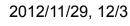
- オブジェクトの比較器
 - 一同じクラスのオブジェクトの大小関係を比較するためのインターフェース。
 - 実装しなければならないメソッド
 - int compare (Object object1, Object object2)
 - object1<object2ならば負数, object1>object2ならば正数, object1==object2ならば0を返す.
 - boolean equals (Object object)
 - 通常のequalsメソッドと同じ.
 - 実装しておくと、パフォーマンス向上に役立つ場合がある.

Comparatorの実装

- 比較器のサンプル.
 - 上較器を適切に実装することで、どのようにもソートできる。

例題9.9 絶対値でソート

- IntegerComaprator(p.33)を修正して、絶対値でソートするようにせよ、クラス名は AbsoluteComparatorとする.
 - 変数の絶対値は Math.abs (値) で求められる.
- 例題9.8 (CollectionSortDemo)を AbsoluteComparatorを使ってソートするよう変更せよ.
 - クラス名はCollectionSortDemo2とする.
- ・ヒント
 - Collections#sort**にソートする集合と**AbsoluteComparator**のオブジェクトを渡す**.



例題9.10 中央値の探索

- 例題9.8 (CollectionSortDemo)の最後に、得られた乱数列(乱数のリスト、配列)の中央値が含まれるか探索する。
 - クラス名はCollectionSearchDemoとする.
 - Javadocを良く読むこと.
 - **-**手順
 - 乱数列から中央値を導き出し、表示する.
 - 中央値とは最大値と最小値を足して2で割った値.
 - 乱数列から中央値を探索する.
 - Collections#binarySearch(list, key)を用いる.
 - » 見つかればインデックスを返し、見つからなければ負の数.
 - 見つかれば、「found!!」と表示する. 見つからなければ「not found」と表示する.

課題9.5 データベースの改良 (1/3)

- 課題9.4のデータベースに以下の機能を追加せよ.
 - -listで表示される一覧を名前順で昇順にソートする.
 - listtelコマンドを追加する.
 - 登録されている情報を一覧で表示する. ただし、電話番号を 昇順にソートされた順に出力される.
 - clearコマンドを追加する.
 - 登録されている全ての情報を削除する.

課題9.5 データベースの改良 (2/3)

- クラス名など
 - クラス名: Database 学生証番号
 - •6桁の学生証番号.最初にgはいらない.
 - ソースファイルをmoodleの「課題9.5 データベースの 改良」に提出。
- 〆切: 2012年12月10日(月) 16:45まで.
 - 今回の課題9.5が提出されており、課題9.4の範囲が 完成していれば、課題9.4も提出されたものとみなす。

課題9.5 データベースの改良 (3/3)

- Mapのソートの仕方
 - キーのソート:SortedMapの実装クラスを用いる.
 - 前回の講義資料を参考のこと.
 - 値のソート:以下のComparatorを用いる.
 - Mapに格納されているMap.Entry型を一度全てArrayListなどに入れる.
 - Map#entrySetを用いるとMapに格納されている値の集合を得られる.
 - Collections.sortメソッドを以下のComparatorとともに呼び出し、ソートする.
- 値のクリア:Map#clearを呼び出す.

```
import java.util.*;
public class TelephoneComparator implements Comparator{
  public int compare(Object object1, Object object2){
    Map.Entry entry1 = (Map.Entry)object1;
    Map.Entry entry2 = (Map.Entry)object2;
    String telephone1 = (String)entry1.getValue();
    String telephone2 = (String)entry2.getValue();
    return telephone1.compareTo(telephone2);
  }
  public boolean equals(Object object1, Object object2){
    return object1.equals(object2);
  }
}
```

応用課題

- 課題9.5のDatabaseにload, save機能を追加せよ.
 - 保存するフォーマットはなんでも良い。
 - 保存するファイルは決め打ちで良い.
- 更に応用課題
 - 現在は、処理の分岐にif文を使っている.
 - if文, switch文を使わず,多態性を用いたプログラムにせよ.
 - 1つのコマンドに対して1つのクラスを作成する.
 - コマンド名をキーとして、それぞれのクラスをMapに入れる.
 - ユーザの入力をスペースで分割し、最初の要素をキーとして、Mapから コマンドオブジェクトを取り出す.
- これができれば、全てのソースファイルをzipでまとめて、課題9.5の提出場所に提出すること。