ImageJ Plugin 作成いろいろ

大阪大学産業科学研究所　新井由之

2012 年 9 月 27 日

目次

1　はじめに

資料では、ImageJ の API や TIPS, 統合開発環境 (IED) である Eclipse のセットアップ方法について簡単に紹介します。著作権は私に帰属します。

2　よく使う ImageJ API

2.1　PluginFilter, PluginFrame, Plugin, PluginTool

ImageJ のプラグインを作ろうと思って ImageJ のメニューから [Plugins]-[New] を選ぶと、上記 4 つの Plugin があり、どれをベースに作ればいいかわからない、ということがあります（実際には IDE 上で作ることがほとんどなので、ここから Plugin を作ることは稀でしょうが)。よく使うのは次の 2 つです。

2.1.1　PluginFilter

ImageJ のもっとも簡単な Plugin テンプレート。インターフェースとして PluginFilter を装したクラスを作ることで、現在アクティブな画像に対して画像処理を行うことが可能です。

ソースコード 1　Plugin Filter

```
// 各種クラスを利用するための import 群（おまじない）
import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.filter.*;

// Plugin として Filter_Plugin クラスを PlugInFilter インターフェースを実装する形で作成
public class Filter_Plugin implements PlugInFilter {
	ImagePlus imp; // ImagePlus(後述)にアクセスするためのメンバ変数

	public int setup(String arg, ImagePlus imp) {
// 画像を処理する前の動作を記述
		this.imp = imp; // メンバ変数 imp にこのプラグイン実行時に受け取った imp を定義
		return DOES_ALL; // どういった画像タイプに対して処理をするか決める。
	}

// 実際の処理部分
	public void run(ImageProcessor ip) { // 現在の画像の ImageProcessor(後述)を受け取る
		ip.invert(); // 画像反転
		imp.updateAndDraw(); // 画像更新
		IJ.wait(500); // 500 ms 待つ
		ip.invert(); // 画像を(再び)反転
		imp.updateAndDraw();
	}

}
```

簡単な処理なら PluginFilter で充分です。

2.1.2 Plugin

Plugin クラスを継承したクラスをプラグインです。自由度が高いのですが、画像の情報（ImagePlus) 等を自分で取得する必要があります。

■ PluginFrame は Frame（フレーム、ウィンドウみたいなもの）を使う事ができるのですが、昔のスタイルのフレームになります。今の OS のウィンドウライクな表示をしたい場合は JFrame クラスを使うのがいいです。PluginTool は最近追加されたクラスのようで、私はよく知りません。マウスのクリック・ドラッグに応答するメソッドがあるので、そういうのを使いたい場合には便利なのかもしれません。

2.2 ImagePlus, ImageProcessor

ImageJ の Plugin を作ろうとして最初に対面する関門です（というほどおおげさではないかもしれませんが)。

ImagePlus は「画像そのもの」を扱います。「画像そのもの」とは、例えばスタックやスライス（この違いも若干混乱しますが）など包含する全てのデータです。一方、ImageProcessor はその中から取り出した画像データを扱います。

2.2.1 ImagePlus

ImagePlus は説明したように画像そのものを扱います。従って、包含するオブジェクトを取り出すメソッドがよく使われます。

ソースコード 2　ImagePlus のよく使うメソッド

```
ImagePlus imp; // よく imp で表されます。
ImageProcessor ip = imp.getProcessor(); // ImageProcessor を得ます。
ImageCanvas ic = imp.getCanvas(); // ImageCanvas を得ます
ImageStatistics is=imp.getStatistics(CENTER_OF_MASS+MIN_MAX+AREA) //
    ImageStatistics を得ます。どんなパラメータを得るかは ij.measure.
    Measurements にあるクラスメンバ変数をセットします）
ImagePlus.addImageListener(listener) // ImageListener クラス
    listener をリスナーとして追加します（後述)。
Roi userRoi = imp.getRoi(); // Roi データを取得します。
imp.setRoi(userRoi); // Roi データを登録します。
Overlay ol = imp.getOverlay(); // Overlay を得ます（後述)。
imp.setOverlay(ol); // Overlay オブジェクトを登録します。
```

2.2.2 ImageProcessor

ImageProcessor は画像データ値にアクセスする際に使われます。

ソースコード 3　ImageProsessor のよく使うメソッド

```
ImageProcessor ip = imp.getProcessor(); // ImageProcessor を得ます。
FloatProcessor fip = (FloatProcessor) ip.convertToFloat(); // 画素データに
    float としてアクセスします
float[] pixVal = (float[])fip.getPixels(); // 画素データ値を
    float 配列 pixVal として取得します。
```

2.3 ImageCanvas

以前は複数の ROI を表示するために使いましたが、その役割は Ovarlay クラスに移行しました。今はマウスやキーボードの情報の読み取りに使います。

```
ImageCanvas ic = imp.getCanvas(); //ImagePlus から ImageCanvas オブジェクトを取得します
        ic.addMouseListener(new icMouseAdapter());
        ic.addKeyListener(new icKeyAdapter());
```

ここで、icMouseAdaptor, icKeyAdapter はそれぞれ java.awt.event.MouseAdapter, java.awt.event.KeyAdapter クラスを継承したクラスを定義しています。必要なメソッド（たとえば public void mouseReleased(MouseEvent e) や、public void keyPressed(KeyEvent e) を overide して使います。

2.4 ImageStatistics

各種解析を行うためのクラスです。ROI における情報を取得するのに使います。使用するクラスにたいして

```
public class hogeclass extend Thread implements Measurements
```

Measurements を実装しておくと、AREA や CENTROID などのメンバ変数をそのまま使えます。

```
imp.setRoi(wandRoi); // wandRoi（後述）を imp にセットします。
ImageStatistics is=imp.getStatistics(AREA+CENTROID);// ImageStatistics として現在の
    ROI から AREA と CENTROID を取得するオブジェクト is を作成します。
double area = is.area;  // 現在のROI の面積（AREA)を求めます。
double xc = is.xCentroid;  //  現在の ROI のセントロイド中心座標を求めます。
double yc = is.yCentroid;
```

2.5 IJ

IJ クラスには ImageJ のプラグインを作成するうえで便利なクラスメソッドが多数用意されています。

```
IJ.log("Message"); // Log ウィンドウにテキストメッセージを表示します
IJ.error("Error"); // エラーメッセージダイアログを表示します。
IJ.versionLessThan("1.43g") // ImageJ のバージョンをチェックします。この場合バージョンが 1.43
    g より小さいと true を返します。
IJ.showStatus("All␣green"); // ImageJ の Status 欄にメッセージを表示します。
IJ.showProgress(x); // プログレスバーをセットしますx の値は 0 から 1 をとります。
```

2.6 ROI,overlay,wand

2.6.1 Roi,wand

ROI (Region Of Interest) は施そうとする画像対象を選択するためによく使われます。矩形や円形・多角形など目的に応じて使い分けます。

wand（ワンド）は Photoshop などではマジックハンドとも言われます。ImageJ では閾値範囲内の連続した画像を取り出す場合に使われます。

ソースコード 4　Roi Wand

```
Roi userRoi =new Roi(x,y,w,h); // ROI として幅w、高さhの矩形 ROI オブジェクトを作成します。
Wand wand = new Wand(ip); // ImageProcessor ip 上に wand オブジェクトを作成します。
wand.autoOutline(wdx, wdy, lt, ht);// 作成したwand オブジェクトを使って座標(x,y)、閾値範囲
    lt から ht の範囲の画像データの輪郭を抽出します。
Roi wandRoi = new PolygonRoi(wand.xpoints,wand.ypoints,wand.npoints,Roi.POLYGON);
// wand データをもとに多角形 ROI(PolygonROI)を定義します。
       imp.setRoi(wandRoi);  // 作成したROI を ImagePlus imp にセットします
       ImageStatistics is=imp.getStatistics(AREA+CENTROID);
//セットした ROI から ImageStistics is を取得します。
```

2.6.2　Overlay

Overlay は、複数 ROI データを ImagePlus に登録する際に使われます（以前は ImageCanvas が使われました)。

```
Overlay ol = imp.getOverlay(); // Overlay を得ます
for(Roi r:roilist) //
    roilist には roi データが複数登録されている List インターフェースを実装するクラス（ArrayList 等
    )とします
       ol.add(r); // For 文を使うことで、Roi を roilist から次々と取り出すことができます
imp.setOverlay(ol); // Roi を ImagePlus にセットします
ol.clear(); // Overlay をクリアします。
```

3　よく使う Java API

3.1　List,ArraiyList

配列は宣言時に指定した数の要素で固定されますが、動的に変化するような要素などを扱う場合、List インターフェースを実装するクラスを利用するのが便利です。

```
private List<FPoint> dplist = new ArrayList<FPoint>(100);;
private List<List<FPoint>> alldplist = new ArrayList<List<FPoint>>();
```

ここでは FPoint を要素クラスとする List として dplist を ArrayList として作成しています。作成するときの引数は最初からその要素分のメモリを確保してくれます（若干動作が速い)。要素の数の増減が可能です。また、List の List といった入れ子構造を作ることも可能です。

```
dplist.add(detP); //dplist に FPoint クラスである detP を登録します。
alldplist.add(0, new ArrayList<FPoint>(dplist)); // 要素番号 0にdplist を「追加」します。
dplist.size(); // 登録されている要素数を返します。
```

3.2　Thread

For ループなどで繰り返し処理を行うと、そのままでは ImageJ の画像の更新等が行われず、実行中はひたすらじっと待つしかなくなります。Thread を利用すれば、画像の更新等の処理を並列して行うことができるようになります。Thread を利用するためには、Thread を継承したクラスを作成すればよいです。

```
public class DetectParticle extends Thread implements Measurements
```

```
2 public void run() {
  //ここにスレッドとして実行する処理を記述する
  }
```

ここでは、DetectParticle という Thread を継承し Measurements を実装したクラスを作成します。Thread を継承したクラスは run() メソッドを定義する必要があります。

```
1 DetectedParticle dp = new DetectParticle();
  dp.start();
```

スレッドを実行するためには作成したオブジェクトに対し start() メソッドを使用します。そうすれば run() に書かれた内容が実行されます。

3.2.1 synchronized

ところで、スレッドを For ループの中で実行すると、最初の処理が終わらないうちに次のスレッドが並列に走ってしまい、望んだ動作をしないことがあります。それを避けるために、あるメソッドを実行中は次のスレッドが走らないようにする必要があります。

```
public synchronized List<FPoint> dpOneSlice(int slice) {
... }
```

synchronized はそのための命令です。run メソッドの中で呼び出すメソッドに対し synchronized をメソッド名の前につけておけば、スレッド実行中もそのメソッドが終わるまで動作を待ってくれます。

4 TIPS など

4.1 現在アクティブな画像の ImagePlus オブジェクトを取得する

ソースコード 5 WindowManager

```
ImagePlus imp = WindowManager.getCurrentImage();
```

また、作成したプラグインを「Window」メニューに表示されるようにするには

ソースコード 6 WindowManager2

```
WindowManager.addWindow(this);
```

とします。

4.2 ２重にプラグインを開くのを防ぐ

ソースコード 7 frame

```
private static Frame frame; // static なクラスとして frame を plugin のメンバ変数として登録
...
if (frame != null){
4       IJ.error("PTA␣is␣already␣implemented");
        return;
        }
frame = this; // frame に自分自身を登録
```

こうしておけば２重に開くのを防ぐことができます。

4.3　画像を開く、閉じる、アップデートされることを検出する

ImageListerner を ImagPlus に追加します。

ソースコード 8　ImageListener

```
ImagePlus.addImageListener(new ImageListener() {
        public synchronized void imageClosed(ImagePlus arg0) {
        // 画像が閉じた時の動作を記述します
        }
        public synchronized void imageOpened(ImagePlus arg0) {
        // 画像が開かれた時の動作を記述します
        }
        public synchronized void imageUpdated(ImagePlus arg0) {
        // 画像が更新された時の動作を記述します
        });
```

addImageListener はクラスメソッドです。各メソッドの引数 arg0 に、その動作が起きた時の ImagePlus のオブジェクトが渡されます。

4.4　Threshold のクリア

閾値がセットされていることを調べたり、セットされている閾値をクリアします。

ソースコード 9　Threshold

```
ip.setThreshold(-808080.0D, ip.getMax(), ImageProcessor.RED_LUT); //
    Threshold のセットに使います（Threshold の下限値に-808080.0
    D をセットすると、Threshold をリセットできます。
ip.getMinThreshold(); ip.getMaxThreshold(); //Threshold の最大値、最小値を得ます。下限値が
    -808080.0D の場合、Threshold はセットされていないことを示します
```

5　統合開発環境 (IDE)　Eclipse のセットアップ

ImageJ の plugin を書くためには Java で記述する必要があります。ImageJ や Fiji には標準のスクリプトエディターが備わっていますが、テキストエディターの域をでておらず、本格的に Plugin を書く場合には不向きです。ここでは、Java 開発環境としてよく知られている Eclipse 上で plugin を作る方法を記述します。

5.1　Eclipse のダウンロード

Eclipse は下記のサイトよりダウンロードできます。

`http://www.eclipse.org/downloads/`

Eclipse には開発環境にしたがってさまざまなバージョンがありますが、ここでは「Eclipse IDE for Java Developers 」を選びます。使用環境を選び、ダウンロードし、解凍したフォルダを適当な場所（Documents フォルダ等）に置きます。

フォルダの中にある「eclipse」を実行します。最初に作業場所として「workspace」の場所を聞かれるので、これまた適当な場所を選びます。作成したプロジェクトはすべてこの workspace フォルダに保存されます。

5.2 ImageJ プロジェクトのセットアップ

workspace フォルダをセットアップ後、Eclipse の起動画面が現れます。アイコンがいくつか現れますが、その中の「Go to workbench」を選びます。

以下一連の作業を記述します。

- 一旦 Eclipse から離れ、ImageJ のサイトから ImageJ のソースコードをダウンロードします。
- `http://rsbweb.nih.gov/ij/download.html`
- 解凍したファイルは適当なフォルダに置きます。
- Eclipse に戻り、[File] - [New] - [Java Project] を選びます。ウィンドウが出てきますが、Project name に適当な名前（たとえば IJ）を入力し、下の [Next] ボタンを押して次に進みます。
- でてくる「Java Settings」で下部にある「Details」から、「Link additional source」を選びます。
- 「Source Folder」ウィンドウの「Browse...」ボタンを押して、先ほどダウンロードした ImageJ のソースファイルを解凍したフォルダ内の「source」フォルダを選びます。すると、Folder name:が自動的に source になります。Finish を押して終了します。
- Finish を押して終了します

これで、ImageJ のソースコードを Eclipse のプロジェクトとして登録することができました。

5.3 Eclipse での Plugin 開発例

では実際に ImageJ の Plugin を作ってみます。

- ImageJ のプロジェクトを作った場合と同じように、[File] - [New] - [Java Project] からプロジェクトを作成します（例：testPlugin)

Next を押し、「Java Settings」ウィンドウに進みます。

- 上のタブの「Projects」を選び、右のボタンから「Add...」を押し、「Select projects to add」から先ほど作った ImageJ のプロジェクトを選択し OK します。
- 他に追加したい外部ライブラリがある場合、タブの「Libraries」から「Add External JARs...」から外部ライブラリを選びます。
- Finish で終わります。
- メインウィンドウの Package Explorer から先ほど作ったプロジェクト (testPlugin) を右クリックし、コンテキストメニューから [New]-[Class] を選びます。
- 「Java Class」ウィンドウの中で Package, Name を入力します。Package は空欄でも問題ありませんが、他の Plugin とのコンフリクトを避けるために一意な名前を付けたほうがよいです。Name はなんでもいいですが、名前の最後に「アンダースコア」をつけないと、ImageJ の plugin メニューに表示されません。
- Interface 項目の右端にある「Add...」ボタンを押します。
- 「Choose interfaces:」に「PlugInFilter」と入れると下に当てはまる項目がでてきます（途中まで入力すれば、候補がでてきます)。
- 下の項目から、「PlugInFilter - ij.plugin.filter」を選び、「Add」、[OK」を選び、「Finish」で終わります。

- すると、PluginFilter を implements したコードが自動的に作成されます。

Eclipse ではコード補完機能や、import の自動追加が行われます。また、ImageJ のソースを取り込んでいるので、ImageJ のコマンドにカーソルを重ねると説明がでますし、右クリックして「Open Declaration」を選べば、ソースコードに直接飛ぶことができます。

6 JNI

JNI は Java Native Interface の略であり、Windows や Mac 依存的な外部プログラムを利用する際に使います。JVM (Java Virtual Machine) の動作は基本的に高速であり、JNI の利用はプラットフォーム依存的なプログラムになってしまうのでできるだけ避けるべきです。どうしても Java 上で走らせるには速度が不十分であったり、すでにあるライブラリ等を利用したい場合にのみ使います。

6.1 ライブラリーの使用

JNI ライブラリーを使う場合、使いたいクラスで以下のように記述します。

```
public native static double[] fit2DGauss(double[] y, double[] x, int sizex, int sizey,
    int[] info);
      // Load fit2DGauss Library
      static {
            System.loadLibrary("fit2DGauss");
      }
```

まず、使用したい外部関数を宣言します。ここでは、２次元ガウス分布でフィッティングするための関数 fit2DGauss を宣言しています。つぎに、ロードするライブラリを読み込む、System.loadLibrary を実行します。static で囲まれているので一度読み込んだら２度は読み込まれません。ライブラリの名前は Winodws の場合***.dll、MacOSX の場合 lib***.jnilib とします。***の部分を loadLibary の引数として渡します。

6.2 javah

この状態でいったんコンパイルし、class ファイルを作成します（ここでは pta/PTA.class)。作成した class ファイルに対し、コマンドプロンプト上で

```
javah -jni pta
```

を実行すると、pta_PTA.h が自動生成されます。

ソースコード 10　pta PTA.h

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class pta_PTA */

#ifndef _Included_pta_PTA
#define _Included_pta_PTA
#ifdef __cplusplus
extern "C" {
#endif
#undef pta_PTA_serialVersionUID
#define pta_PTA_serialVersionUID 1LL
```

```
/*
 * Class: pta_PTA
 * Method: fit2DGauss
 * Signature: ([D[DII[I)[D
 */
JNIEXPORT jdoubleArray JNICALL Java_pta_PTA_fit2DGauss
  (JNIEnv *, jclass, jdoubleArray, jdoubleArray, jint, jint, jintArray);

#ifdef __cplusplus
}
#endif
#endif
```

このファイルをいじってはいけません。このうち、JNIEXPORT からの宣言文を取り出し、ライブラリ本体の記述に使います。fit2DGauss.cpp の一部を記述します。

ソースコード 11　fit2DGauss.cpp の一部

```
// ヘッダファイルpta_PTA.hからコピペしてきます。引数名を適当につけます。
JNIEXPORT jdoubleArray JNICALL Java_pta_PTA_fit2DGauss(JNIEnv *env,
jclass jcla, jdoubleArray jdobFionaData,
jdoubleArray jdobParam, jint sx, jint sy, jintArray jinfoParam) {
// 配列を扱うために、まず引数として渡される配列の要素数を知る必要があります。
//env->GetArrayLength(引数として渡された配列名)で数を調べます。
    jsizeは配列要素数を調べるための型です。
jsize paramLength = env->GetArrayLength(jdobParam); //
jsize fionaDataLength = env->GetArrayLength(jdobFionaData); //

// 次に、配列要素にアクセスするためのポインタを張ります
//env->GetDoubleArrayElements(配列名,NULL)です(double型の場合)。別の型の場合は対応する型を
    使います。
jdouble *jfdata = env->GetDoubleArrayElements(jdobFionaData,NULL); // to access
    jdobData by jdata
jdouble *jparam = env->GetDoubleArrayElements(jdobParam,NULL);
jint *jinfo = env->GetIntArrayElements(jinfoParam,NULL);

// 戻り値用の配列の宣言です。
jdoubleArray retParam = env->NewDoubleArray(paramLength); //to return param

//C++なのでnewで配列を確保します。cの場合はmallocを使います。
fdata = new double[static_cast<int>(fionaDataLength)];
double *param = new double[static_cast<int>(paramLength)];
```

コードが記述できたらコンパイルします。Windows だと VisualC++2010Express、MacOSX だと XCode を使うことができます。その際、jni.h をインクルードする必要があります。

6.2.1　Windows の場合

MicrosoftVisualC++2010Express を利用する（無料）がよいと思います（MinGW 等でも可能)。以下簡単に手順を記述します。

- 新しいプロジェクト... を選びます
- テンプレートから CLR →クラスライブラリを選びます
- ソリューションエクスプローラからプロジェクトを選び右クリック→プロパティを選びます

- 構成プロパティ→ VC++ ディレクトリを選び、インクルードディレクトリから jdk の invlude, include/win32 を追加します (jni.h をインクルードするため)
- C/C++ →コード生成→ランタイムライブラリの「マルチスレッド DLL（/MD)」を「マルチスレッド (/MT)」に変更します。こうしないと、DLL の実行に VC++2010 が必要（正確には MXVCR100.DLL) になってしまいます（配布をきにしなければデフォルトでかまいません）
- プロパティの設定を閉じます。
- ヘッダファイルとして先ほど javah で作ったファイルを追加（上のアイコンにある「既存の項目の追加」）します。必要なライブラリ等も追加します。

6.2.2 MacOSX(Xcode) の場合)

Xcode の場合もそれほどやることはかわりません（jni.h のインクルード等)。簡単に記述します。

- Create a new Xcode project を選びます。
- OS X → Framework & Library から「C/C++ Library」を選びます
- プロジェクト名を入れてプロジェクトを作成します
- Build Settings の Architectures を「Standard (32/64-bit Intel)」に変更します
- Base SDK をできるだけ低いバージョン（互換性確保の為）を選びます　（このあたりは配布とか気にしなければデフォルトでかまいません）
- Packaging にある Excecutable Extension を「jnilib」に、Executable Prefix を「lib」に変更します。
- 必要なヘッダー類（jni.h, jni_md.h 等）はドラッグして左のメニューに入れればよいです。jni.h の場所は/System/Library/Frameworks/JavaVM.framework/Versions/A/Headers/にあります。

このあたりの内容がちんぷんかんぷんの場合はネット等で情報を得て下さい。

参考文献

[1] Java api. `http://docs.oracle.com/javase/jp/6/api/`
Java API の一覧です。ここも時々お世話になります.

[2] Visual studio 2010 express と cuda4.0 でプログラムを実行してみる その 1 インストール. `http://feather.cocolog-nifty.com/weblog/2011/07/visual-studio-2.html`
VC++2012Express で CUDA を設定する方法のサイトだが、その前に VC++2012Express の 64bit コンパイル環境を構築する方法を紹介しています（結構ややこしい)。DLL を３２ビットバージョンだけでなく、６４ビットバージョンでも提供する場合に必須になります.

[3] Tatsuo Ikura. Java drive. `http://www.javadrive.jp/tutorial/`
Java の GUI に関してケースバイケースでやり方が詳細されています。テーブルの扱い方法は特に勉強になりました.

[4] Wayne Rasband. Imagej api. `http://rsbweb.nih.gov/ij/developer/api/index.html`
言わずと知れた ImageJ の API 集。プラグインを作成する際にはこのサイトに大いにお世話になります.

[5] Wayne Rasband. Imagej code. `http://rsbweb.nih.gov/ij/developer/source/index.html`
ImageJ のソースコードはすべて開示されているので、動作がよくわからない API を調べるときにソー

スコードに直接あたると解決することがあります.

[6] ジョシュアブロック (著), 柴田芳樹 (訳). *Effective Java*. ピアソン・エデュケーション, 第 2 版, 2008.
コメント:まさに「Effective」な Java プログラムを書く際の技術が豊富に紹介されている。内容が結構難しいのである程度自由に Java プログラムを書けるようになってから、ぱらぱらと読むのが良いです.

[7] 柴田望洋. 明解 Java によるアルゴリズムとデータ構造. ソフトバンククリエイティブ, 第 1 版, 2007.
コメント:プログラミングをする時に役立つアルゴリズム本。この本でなくても多言語のアルゴリズム本でも基本は同じなので自分にとって使いやすい本を選ぶといいと思います.

[8] 大村忠史, 池田成樹. Java GUI プログラミング Java SE 6 対応 [Vol.1]. 株式会社カットシステム, 第 1 版, 2007.
コメント:GUI を学ぶのに最適な書。フレーム内の部品の配置やテーブル、リスナーについてなどわかりやすく書いてあります.

[9] 柴田望洋. 明解 Java 入門編. ソフトバンククリエイティブ, 第 1 版, 2007.
コメント:Java を 1 から学ぶのに最適。Web でも勉強できるが、やはり 1 冊入門書を手元に置いておくと役立ちます.

[10] 八木裕乃, 明壁敦子. 徹底攻略 Java2 プログラマ問題集 Platform5.0 対応. 株式会社ソキウス・ジャパン, 第 1 版, 2006.
コメント:問題集だが、Java について広範囲に効率よく学ぶことができる。３回くらいやって８割以上とれるようになれば充分.