オブジェクト指向言語 Java

博士(情報理工学) 小林　貴訓
博士(工学) Htoo Htoo 共著
工学博士 大澤　裕

コロナ社

ま　え　が　き

このテキストは，プログラム言語 Java について解説しています．Java はオブジェクト指向という考え方をベースとしてつくられた言語です．オブジェクト指向言語では，プログラムの部品化を容易に行え，再利用しやすくなります．最近の多くのプログラミング言語は，Java と同様にオブジェクト指向言語です．したがって，Java を習得すれば C#や Visual Basic .NET，C++など，他のオブジェクト指向言語を学ぶときに，理解が容易になります．

プログラミングは難しい作業ではありません．一度基本を覚えてしまえば，使用する言語が変わっても，容易に順応できるようになります．しかし，最初のプログラミング言語を覚えるときには若干の障壁を感じるかもしれません．その障壁を乗り越えるためには訓練が必要です．プログラミングを初めて学ぶ人にとっての訓練とは，このテキストのような教材に載っているプログラムを自分で実際にタイプし，コンパイルし，実行してみるという，一見単純作業の積み重ねが重要と考えます．自分でタイプすることにより，慣用句的な表現が記憶に残ります．また，プログラムのタイプミスが生じるかもしれません．その結果，コンパイル時にエラーが出力されます．もし，コンパイルエラーが生じたら新しいことを学ぶチャンスと考えてください．コンパイルエラーメッセージから原因を推測し，テキストのプログラムと見比べてみてください．エラーが発見できないときもあきらめず，友達と検討してください．最後には教員に聞き，必ず原因を理解してください．このような地道な訓練がプログラミング言語の基礎レベルの学修に非常に有益です．

Java をはじめ，多くのプログラミング言語のテキストにはソースファイルが CD や DVD で付録として付けられています．すでにプログラミングの経験がある人が新しい言語を学ぶときには，それらは有効と思いますが，ビギナーのうちは必ず自分でタイプすることが重要です．

大規模なプログラミングにおいては，Eclipse や NetBeans のような統合開発環境（IDE）を使います．それらにより，タイプが容易になったり，タイプミスの発見が早くなり，またデバッグも容易になります．しかし，初学者のうちは，これらは使わず是非テキストエディタとコマンドプロンプトでプログラミングをしてください．統合開発環境はプログラミングの基礎を習得した後で使えば効率化の面でたいへん有益です．しかし初学者のうちは，考えて，調べて，聞くという一見非効率な作業が，習得には有益と考えます．

インターネット上には Java に関する情報があふれています．コンパイルエラーの意味や理

由がわからないときは，検索エンジンにコンパイルエラーメッセージをコピー，ペーストして尋ねることにより解決することもよくあります．プログラミング言語の習得に，インターネットも活用してみてください．

2016 年 9 月

大澤　裕

† 本書に掲載の Java プログラムソースコードは，コロナ社 Web ページの本書の紹介ページ
`http://www.coronasha.co.jp/np/isbn/9784339028652/`
にアーカイブデータをアップしております．必要に応じて，ダウンロードの上，ご活用ください．アーカイブデータ解凍の際のパスワードは以下のとおりです．
OBjJaVa028652CorOnA
また，各章末にある章末問題の解答も併せて上記 Web ページに用意しておりますので，必要に応じてご利用ください．

目　　次

1. とりあえず Java を使ってみる

2. Java の基礎

3. クラスと Java プログラムの基本

4. クラスの拡張

5. 抽象クラスとインタフェース

6. パッケージと例外処理

7. GUI プログラム

8. さまざまなコントロール

9. 図形の描画

10. ファイルの入出力

11. クラスライブラリー

12. マルチスレッド

7 GUI プログラム

Java Programming

前章までで扱ったプログラムは，コンソールに対して文字の入出力を行うものでした．このようなアプリケーションプログラムは **CUI**（console user interface）プログラムと呼ばれます．一方，本章から 9 章まででは **GUI**（graphical user interface）プログラムの作成法について述べます．まず，本章では GUI アプリケーションの基本について述べます．8 章では，GUI を構成する部品（コントロールと呼ばれます）と，その使い方について述べます．9 章では，グラフィックスの表示法やマウスを用いた座標の取得法について述べます．

7.1 JavaFX による簡単なプログラム

GUI アプリケーションを作成するためには，GUI ライブラリーを使います．この GUI ライブラリーには，AWT，Swing，そして本書で扱う JavaFX などがあります．JavaFX は Java8 から標準のライフラリーとなりました．しかし，以前の Java では AWT と Swing がよく使われてきました．そこで現在広く使われているアプリケーションでも AWT や Swing が使われています，しかし今後作成されるアプリケーションは JavaFX で書かれるものが増えると考えられますので，本書では **JavaFX** について解説します．

最初に，JavaFX の機能を用いた GUI アプリケーションの例をプログラム 7.1 に示します．

[プログラム 7.1: ch7/WindowApp.java]

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.layout.StackPane;
4  import javafx.stage.Stage;
5
6  public class WindowApp extends Application {
7      @Override
8      public void start(Stage stage) throws Exception {
9          StackPane root = new StackPane();
10         Scene scene = new Scene(root, 300, 250);
11         stage.setTitle("WindowApp");
12         stage.setScene(scene);
13         stage.show();
14     }
15
```

```
16        public static void main(String[] args) {
17            launch(args);
18        }
19    }
```

このプログラムをつぎのようにコンパイルして，実行します．

[実行例]

```
% javac WindowApp.java
% javaw WindowApp
```

実行例の 2 行目で，いままでの java コマンドの代わりに javaw というコマンドを使っています．これは，いままでどおり java でもよいのですが，java コマンドの場合，プログラムが終了するまで，コマンドプロンプトが出されません．GUI プログラムでは，ユーザと対話しながら処理を行うため，プログラムが起動してからその終わりまでの時間が長いのが普通です．一方，java コマンドで起動した場合，プログラムが終了するまでコンソールが占領されてしまいます．javaw コマンドでプログラムを起動した場合には，コンソールにすぐにつぎのコマンドを受け付けるプロンプトが出されますので，GUI プログラムではこちらが好まれます．

このプログラムの表示例を**図 7.1** に示します．ここに見られるように，つくられたウィンドウの中にはなにも入っていません．章が進むにつれ，コントロールと呼ばれる GUI 部品がウィンドウ内に配置されていきます．まずは，このなにもないウィンドウのつくられ方について説明します．

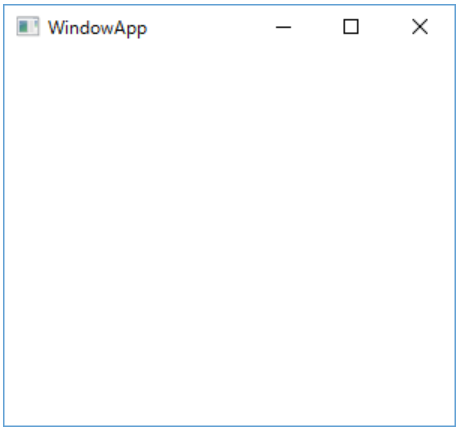

図 7.1 WindowApp の表示例

JavaFX を用いた GUI プログラムも，javaw コマンドに与えられたクラス（この例では WindowApp）の main メソッド（16 行目）から実行が始まります．しかし，main メソッドに書かれているのは launch(args) の 1 行のみです．このメソッドは，main と同じクラス内にある start メソッドを起動します．実質的には，JavaFX で書かれた GUI プログラムは，この start メソッドから実行が開始すると考えておけばよいでしょう．

プログラム 7.1 の 6 行目で，クラス定義の先頭に public 修飾子が付けられています．この public は必要です．理由は，Application クラスにおいて，このクラスを参照しているためです．

さて，プログラムに書かれたクラス `WindowApp` は `Application` クラスを拡張してつくられています．このクラスは抽象クラスであり，`start` メソッドの実装が要求されます．この `start` メソッドの中に複数の行が書かれていますが，ウィンドウを表示するために必要なのは，13 行目の `stage.show()` だけです．ためしに，`start` メソッドから `stage.show()` だけを残してすべて消してしまうと，大きななにもないウィンドウが表示されることになります．

残りの行を説明する前に，JavaFX の GUI プログラムの構成について簡単に説明しておきます．物理的に表示されるウィンドウに対応するのは，`Stage` クラス（`javafx.stage.Stage`）です．このクラスはトップレベルのコンテナと呼ばれます．コンテナとは，GUI を構成するさまざまな部品を入れる入れ物のことです．この Stage クラスは，ウィンドウのサイズや背景の色などを指定することができませんので，それらを行うために `Scene`（`javafx.scene.Scene`）というクラスを使います．さらに，この Scene 内にさまざまな部品を配置するために，Pane というクラスを用います．このクラスはレイアウト，またはコンテナとも呼ばれます．後で扱う Button や Label などの GUI 部品（コントロール）は，Pane を用いて適切な位置に配置します．Pane には多数の種類があり，本章の後半で説明します．ここでは，単純な StackPane（`javafx.scene.layout.StackPane`）というクラスを用いています．

以上で述べた処理を，プログラム 7.1 に対応づけます．まず，9 行目は `StackPane` のオブジェクトをつくり，`root` という変数に代入しています．10 行目では，`Scene` クラスのオブジェクトをつくり変数 `scene` に代入しています．コンストラクタの最初の引数には，StackPane のオブジェクトを，つぎの二つの引数には，ウィンドウの横と縦のサイズを指定します．ここの指定では，横方向に 300 画素，縦方向に 250 画素の大きさのウィンドウがつくられます．11 行目では `start` メソッドに引数として渡された `stage` オブジェクトに対して，タイトルバーに表示するメッセージを指定しています．ここで指定した文字列が，ウィンドウのタイトルバーに表示されます．12 行目では先ほどの `scene` を `stage` に登録しています．最後に 13 行目の `stage.show()` でウィンドウが画面に表示されます．

以上で述べたことを**図 7.2** に示します．Stage がウィンドウの土台にあり，その上に Scene と Pane が順に配置されます．後で述べる Button や Label などの GUI 部品は Pane の上に配置されます．

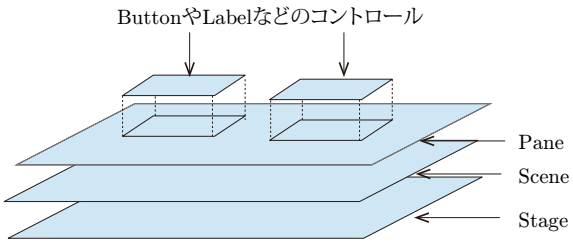

図 7.2　JavaFX の構成

7.3　イベント処理の基本

Java では，画面にボタンやプルダウンメニューなどのコントロールを配置した GUI を容易に作成することができます．例えばボタンでは，それがマウスでクリックされたときにイベントが発生し，そのイベントに対してあらかじめ登録されている処理が実行されます．このような処理を記述したものをイベントハンドラと呼びます．

Java でのイベント処理は，委任イベントモデル（delegation event model）という方法で行われます．イベントの種類には，マウスの移動やボタンの押下，キーボード上のキーの押下，GUI 上に配置されたボタンの選択，メニューの選択などたくさんの種類が考えられます．委任イベントモデルとは，イベントが発生したとき実行させたいメソッドをイベントを発生させるコントロールに登録しておき，実際のイベント発生により自動的にそれらのメソッド（イベントハンドラ）が呼び出される方式です（**図 7.4**）．

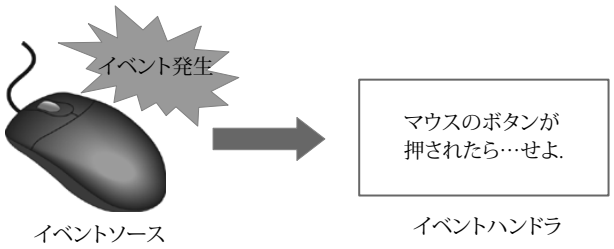

図 7.4　委任イベントモデル

ボタンやマウスのように，**イベント**を発生させるものを**イベントソース**と呼びます．そのイベントは，イベントの種類に応じて記述される**イベントハンドラ**と呼ばれるメソッドで処理されます．したがって，委任イベントモデルでは

1. 個々のイベントを処理するイベントハンドラを定義・宣言しておく
2. そのイベントソースにハンドラを登録する

という手順がとられます．

さて，プログラム 7.2 の例に戻ります．ボタンが押されたとき，`ActionEvent` というイベントが発生します．そこで，ボタンには `ActionEvent` に対する動作を記述したメソッドを登録することになります．これを行うためには，EventHandler（`javafx.event.EventHandler`）インタフェースの `handle` メソッドをオーバーライドします．ここで行う内容は，ラベルに対して `Hello World!` という文字列を設定することです．

プログラム 7.2 では匿名クラスを用いていますが，これを通常のクラスで書き換えるとプログラム 7.3 のようになります．

```
3  import javafx.event.EventHandler;
4  import javafx.scene.Scene;
5  import javafx.scene.control.Button;
6  import javafx.scene.control.Label;
7  import javafx.scene.layout.VBox;
8  import javafx.stage.Stage;
9
10 public class GUISample2 extends Application {
11
12     @Override
13     public void start(Stage stage) {
14         Label lb = new Label();
15         lb.setPrefWidth(250); lb.setPrefHeight(50);
16         Button btn = new Button();
17         btn.setPrefWidth(250); btn.setPrefHeight(50);
18         btn.setText("Click Me");
19         btn.setOnAction(event -> lb.setText("Hello World!"));
20         VBox root = new VBox();
21         root.getChildren().addAll(lb, btn);
22         Scene scene = new Scene(root, 250, 100);
23         stage.setTitle("GUISample");
24         stage.setScene(scene);
25         stage.show();
26     }
27 }
```

7.4 レイアウトの方式

JavaFX による GUI アプリケーションには，画面をつくる Stage があり，その Stage には Scene が登録されることを述べました．また，GUI に配置するコンポーネントは Pane というものを用いて配置し，それを Scene に登録することにより，画面上に表れることになります．

さまざまなコンポーネントの配置はシーングラフ（scene graph）で管理されています．シーングラフは木構造をしています．つまり，一つの根ノード（root）をもち，根ノードと中間ノードには Pane が配置され，葉ノードにボタンなどのコントロールが配置されたグラフです．この木は次数は不定です．つまり，各ノードのもつ子ノードの数は定まっていません．言い換えると，各 Pane に登録されるコントロール（や別の Pane）の数を自由に決めることができます．

Pane への子ノードの登録は，プログラム 7.2 ではつぎのように行いました．

```
VBox root = new VBox();
root.getChildren().addAll(lb, btn);
```

VBox 型の変数名が root になっているのは，それがシーングラフの根ノードになっていることを示しています．どの Pane でも共通の操作は，Pane の `getChildren()` メソッドで子ノードの参照を得て，そこに `addAll` メソッドで引数に指定されているコントロールや別の Pane を登録することです．この操作を組み合わせることによりシーングラフがつくられます．

いままでの例で，プログラム 7.1 ではレイアウトとして StackPane を，プログラム 7.2 では VBox を使いました．以下では，よく使われるレイアウトについて説明します．ただし，プログラムの構造を単純なものにするためにコントロールにはすべて Button を用います．

7.4.1 HBox

HBox によるレイアウトは，コントロールを横一列に並べます．レイアウトに HBox を用いている他には，このプログラム 7.5 には特に新しいものはありません．11 行目から 13 行目で四つの Button をつくり，14 行目で HBox レイアウトをつくり，15 行目では四つのボタン（配列に入れられています）を登録しています．16 行目で Scene を作成する際に，大きさを指定していませんが，大きさが省略された場合には適当なサイズでウィンドウがつくられます．もちろん scene の大きさを指定してもかまいません．このプログラムの実行画面を**図 7.5** に示します．

[プログラム 7.5: ch7/HBoxSample.java]

```
1   import javafx.application.Application;
2   import javafx.scene.Scene;
3   import javafx.scene.control.Button;
4   import javafx.scene.layout.HBox;
5   import javafx.stage.Stage;
6
7   public class HBoxSample extends Application {
8       @Override
9       public void start(Stage stage) {
10          stage.setTitle("HBoxSample");
11          Button[] button=new Button[4];
12          for(int i=0; i<4; i++)
13              button[i]=new Button("button-"+i);
14          HBox root=new HBox();
15          root.getChildren().addAll(button);
16          stage.setScene(new Scene(root));
17          stage.show();
18      }
19  }
```

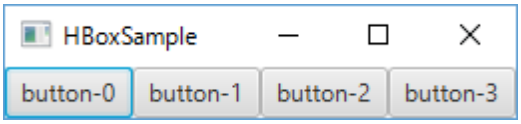

図 7.5　HBox を用いたレイアウト

7.4.2 BorderPane

ボーダーレイアウト（BorderLayout）は，コンポーネントを配置するコンテナ上で，上（TOP），下（BOTTOM），左（LEFT），右（RIGHT），および中央（CENTER）の位置を指定して配置する方式です．ただし，これらの 5 種類をすべて指定する必要はなく，必要なもののみを指定します．ボーダーレイアウトを用いた簡単な例をプログラム 7.6 に示します．

[プログラム 7.6: ch7/BorderPaneSample.java]

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.scene.layout.BorderPane;
5  import javafx.stage.Stage;
6
7  public class BorderPaneSample extends Application {
8      @Override
9      public void start(Stage stage) throws Exception {
10         Button buttonC=new Button("Center");
11         buttonC.setPrefWidth(200); buttonC.setPrefHeight(200);
12         Button buttonT=new Button("Top");
13         buttonT.setPrefWidth(300); buttonT.setPrefHeight(50);
14         Button buttonB=new Button("Bottom");
15         buttonB.setPrefWidth(300); buttonB.setPrefHeight(50);
16         Button buttonL=new Button("Left");
17         buttonL.setPrefWidth(50); buttonL.setPrefHeight(200);
18         Button buttonR=new Button("Right");
19         buttonR.setPrefWidth(50); buttonR.setPrefHeight(200);
20         BorderPane root=new BorderPane();
21         root.setCenter(buttonC);
22         root.setTop(buttonT);
23         root.setBottom(buttonB);
24         root.setLeft(buttonL);
25         root.setRight(buttonR);
26         Scene scene=new Scene(root,300,300);
27         stage.setTitle("BorderPaneSample");
28         stage.setScene(scene);
29         stage.show();
30     }
31 }
```

プログラム 7.6 の実行画面を**図 7.6** に示します.

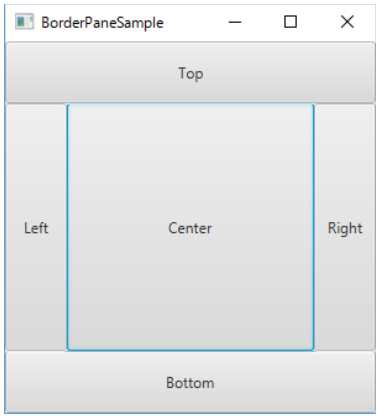

図 7.6 BorderPaneSample の実行例

このプログラムでは，10 行目で Button のインスタンスをつくり，11 行目でそのボタンのサイズを指定しています．これを後 4 回繰り返しています．20 行目で BorderPane のインスタンスをつくっています．21 行目から 25 行目までは，作成された Button を BorderPane に表示位置を指定して登録しています.

```
i= 1, val=  9
i= 0, val= 10
```

10.7 ファイルチューザ

7 章から 10 章で多数の JavaFX のコントロールについて説明しましたが，ファイル操作時に頻繁に使われるコントロールについて説明していませんでした．アプリケーションプログラムでは，ファイルを開き，そのファイルの内容を読み込んだり，ファイルにデータを書き出したりします．ここで，本章で行ったようにファイル名をプログラム中に記述したり，コマンドラインの引数で受け取ることも可能ですが，多くの Windows プログラムにおけるファイル操作では**ファイルチューザ**（**FileChooser**）と呼ばれる画面が表れ，フォルダやファイルの選択を GUI により容易に行えます．以下では，この FileChooser（javafx.stage.FileChooser）の Java での利用について説明します．

以下で説明するプログラム 10.12 の実行画面を**図 10.2** に示します．プログラムを起動すると左側上の画面が表れます．中央の「ファイル選択」というボタンを押下すると，右側のファイルチューザが表れます．ここでファイルを選択し，「開く」ボタンを押すと，ファイル名などの情報がアプリケーションに渡ります．このプログラムでは，ボタンの下にラベルを配置し，そこに選択されたファイル名を表示しています．

[プログラム 10.12: ch10/FileChooserSample.java]

```
1  import java.io.File;
2  import javafx.application.Application;
3  import javafx.scene.Scene;
4  import javafx.scene.control.Button;
5  import javafx.scene.control.Label;
6  import javafx.scene.layout.VBox;
7  import javafx.stage.FileChooser;
8  import javafx.stage.Stage;
9
10 public class FileChooserSample extends Application {
11     Label lb=new Label();
12     @Override
13     public void start(Stage stage) {
14         Button btn = new Button("ファイル選択");
15         btn.setOnAction(event -> fileOpen(stage));
16         VBox root = new VBox();
17         root.getChildren().addAll(btn, lb);
18         Scene scene = new Scene(root, 200, 100);
19
20         stage.setTitle("FileChooser");
21         stage.setScene(scene);
22         stage.show();
23     }
24
25     void fileOpen(Stage stage) {
26         FileChooser fc=new FileChooser();
27         fc.setInitialDirectory(new File("C:\\home\\abc"));
```

```
28            File file=fc.showOpenDialog(stage);
29            if(file != null && file.isFile()) {
30                lb.setText(file.getName());
31            }
32        }
33    }
```

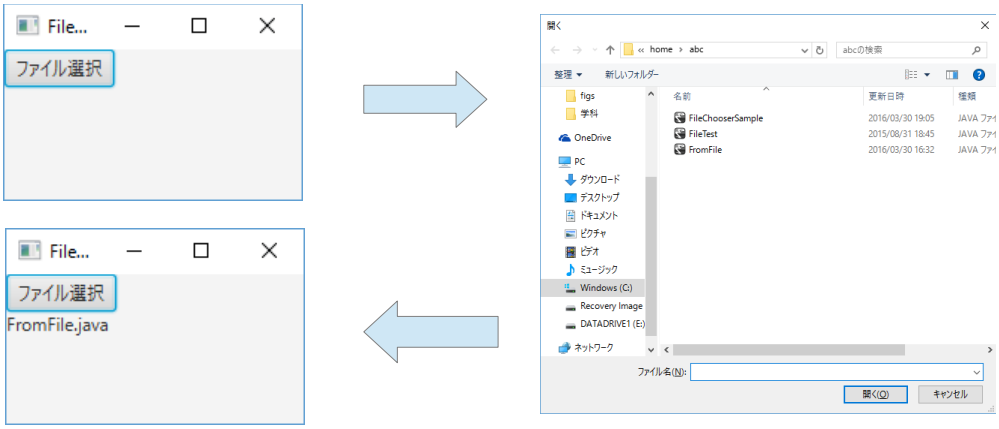

図 10.2　FileChooser の例

15 行目に示すように，ボタンが押されると，25 行目以下の fileOpen が実行されます．26 行目で FileChooser のオブジェクトをつくっています．27 行目では，ファイルチューザに最初に表示されるフォルダを設定しています．この設定が行われていないとき，Windows10 では PC の画面がファイルチューザに表示されます．

28 行目の処理でファイルチューザが画面に表れます（図 10.2）．その画面から目的とするファイルを選び「開く」ボタンを押すと，選ばれたファイルのオブジェクトが file に返されます．29 行目では，戻り値が null でなく，かつ選ばれたものがファイル（他にフォルダがあります）であれば，そのファイル名を得て，ラベルに表示しています．ファイルチューザで「取り消し」ボタンが押された場合，file には null が返されます．

プログラム 10.12 では，ファイルチューザに指定したフォルダ内のすべての種類のファイルが表示されます．しかし，アプリケーションごとに対象とするファイルの種類が決まっており，表示されるファイルも種類もアプリケーションが対象とするものに絞り込みたい場合があります．そのためには，ファイルチューザに ExtensionFilter（javafx.shape.FileChooser.ExtensionFilter）を設定します．

いま，フォルダ内の java ソースファイル（拡張子は.java）と，テキストファイル（拡張子は.txt）のみを表示したいとき，26 行目の下に，つぎの文を追加します．

```
fc.getExtensionFilters().addAll(
    new ExtensionFilter("Text Files","*.txt");
    new ExtensionFilter("Java Source","*.java"));
```

この追加を行った後の表示画面を**図 10.3** に示します．ExtensionFiler のコンストラクタの第 1 引数は右下の楕円で囲んだ部分に表示される文字列です．ファイルの種類を判別しやすい文字列を指定します．また，第 2 引数には*につづけてそのファイルの拡張子を指定します．楕円で囲んだ部分は，ドロップダウンリストになっており，上で述べたように指定したファイルの種類を選択できます．このドロップダウンリストに表示される順番は，addAll メソッドに記述した順番になります．

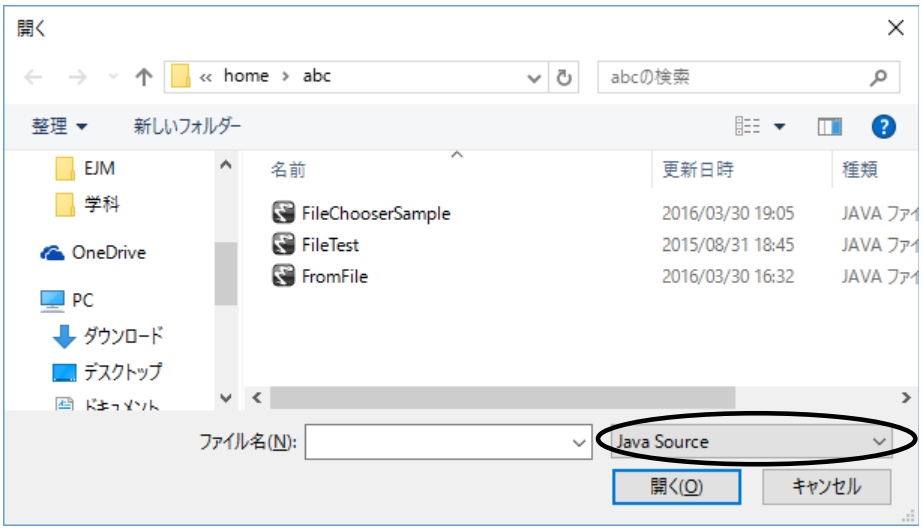

図 10.3　拡張子による絞り込み

章　末　問　題

【1】 以下の文章のうち，誤っている文章を選び，番号で答えなさい．

(1) `System.out.printf` というメソッドは，`import System.*;` という宣言をファイルの先頭に記述することにより，`out.printf` で呼び出すことができる．

(2) `System.in.read()` メソッドは，例外を発生する可能性があるため，try-catch 文の中に記述するか，それを呼び出すメソッドが例外を throw しなければならない．

(3) Scanner クラスで入力を行うとき，整数データの読込みは hasNextInt() メソッドを用いて行う．

(4) ファイル名の変更は，java.nio.Files クラスを用いて行える．

(5) バイナリーデータの出力には java.io.DataOutputStream を用いる．

(6) ランダムアクセスファイルにおいて，ファイル中での読込み位置を指定するためには，getFilePointer メソッドを用いる．

【2】 つぎのプログラムは，java.io.PrintWriter クラスを用いて，カレントフォルダの `text.txt` という名前のファイルに 1 行の文字列を書き込み，ファイルを閉じる一連の流れが記述されている．空白の (A)〜(C) を埋めてプログラムを完成させなさい．

索　　引

―― 著 者 略 歴 ――

小林　貴訓（こばやし　よしのり）
2000 年　電気通信大学大学院修士課程修了（情報システム運用学専攻）
2000 年　三菱電機株式会社
2007 年　東京大学大学院博士課程修了（電子情報学専攻）
　　博士（情報理工学）
2007 年　埼玉大学助教
2014 年　埼玉大学准教授
　　現在に至る

Htoo Htoo（とう　とう）
2004 年　ヤンゴンコンピュータ大学修士課程修了（計算機科学専攻）
2004 年　ヤンゴンコンピュータ大学助教
～08 年
2013 年　埼玉大学大学院博士後期課程修了（数理電子情報専攻）
　　博士（工学）
2013 年　埼玉大学助教
　　現在に至る

大澤　　裕（おおさわ　ゆたか）
1978 年　信州大学大学院修士課程修了（電子工学専攻）
1982 年　東京大学生産技術研究所助手
1985 年　工学博士（東京大学）
1989 年　埼玉大学助手
1990 年　埼玉大学助教授
1998 年　埼玉大学教授
　　現在に至る

オブジェクト指向言語 Java

Object Oriented Programming Language Java

2016 年 11 月 2 日　初版第 1 刷発行　★

検印省略

著　者　小林貴訓
　　Htoo Htoo
　　大澤　裕
発行者　株式会社　コロナ社
　　代表者　牛来真也
印刷所　三美印刷株式会社

112–0011　東京都文京区千石 4–46–10
発行所　株式会社　コロナ社
CORONA PUBLISHING CO., LTD.
Tokyo Japan
振替 00140–8–14844・電話(03)3941–3131(代)
ホームページ http://www.coronasha.co.jp

ISBN 978–4–339–02865–2　（金）　（製本：愛千製本所）

Printed in Japan