NetBeansではじめるJava

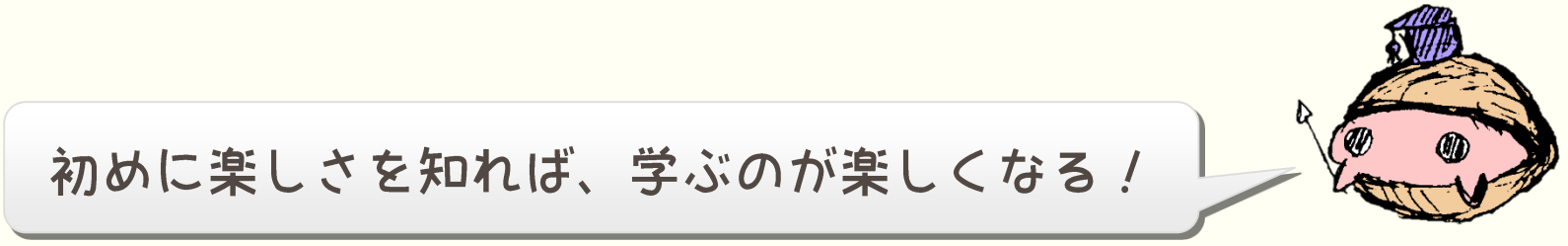
研究効率UP編

加納 徹

Java講習会のねらい

- Javaの文法をしっかり学ぶ
- アルゴリズムの知識を身に付ける
- オブジェクト指向について学ぶ

- **ソフトウェア開発を身近に感じる**
- **データ処理技術の習得で研究効率UP**



Java講習会の流れ

1. 開発環境の構築
2. 基本的なプログラミング
3. ソフトウェアの作り方
4. テキストファイルの入出力
5. 画像ファイルの入出力

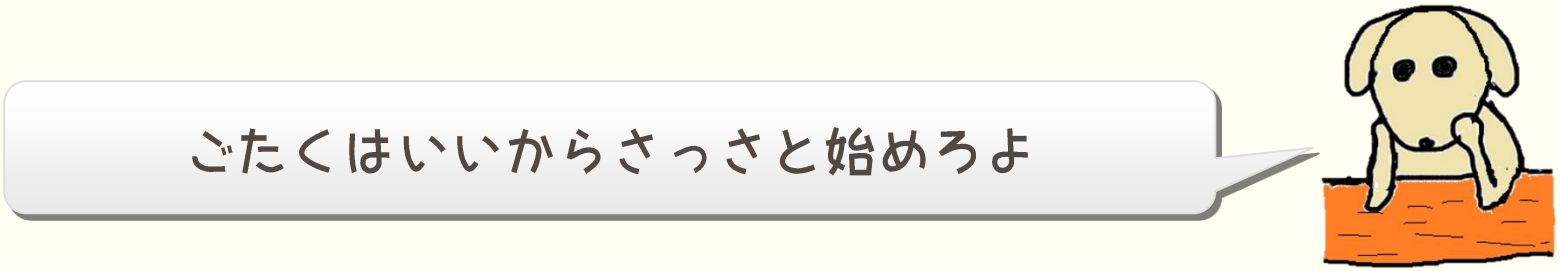
1. 開発環境の構築

Javaの開発環境

1. JDK(Java SE Development Kit)
   - Javaでプログラミングする際に必要な開発キット

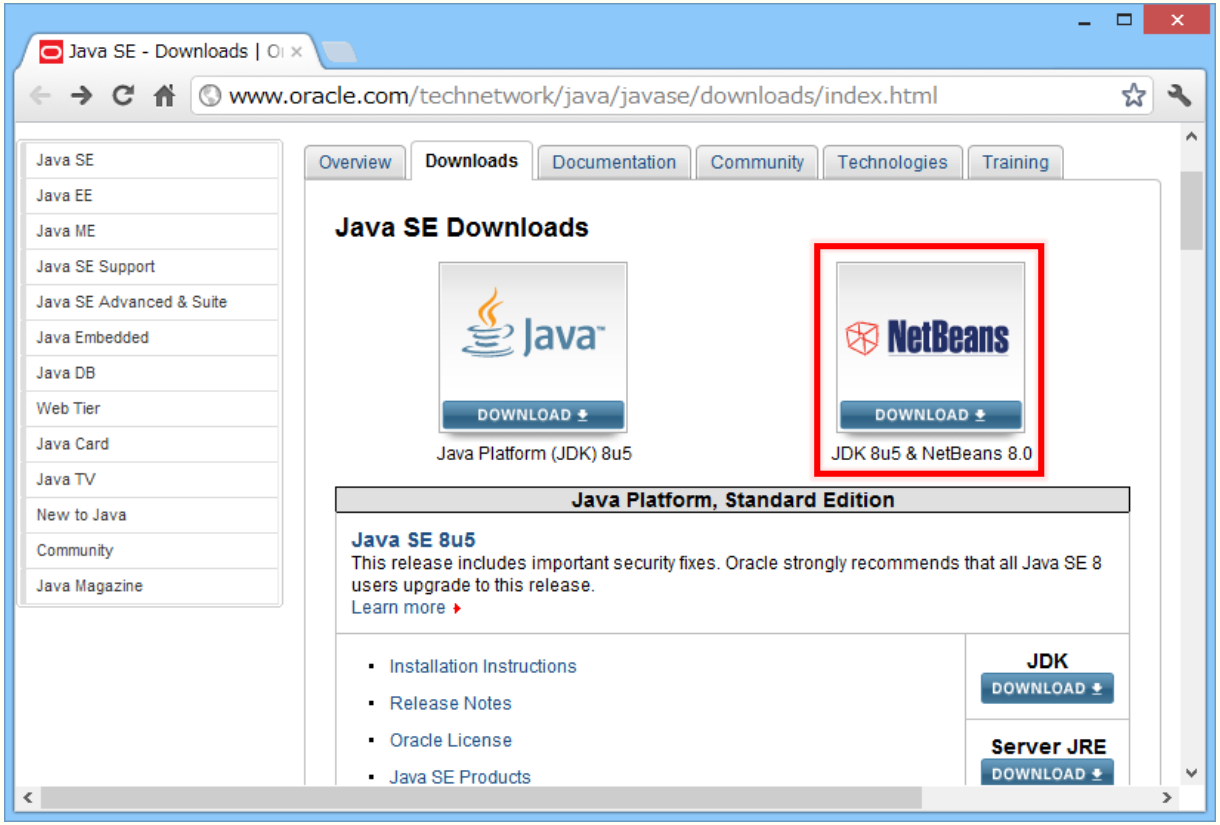
2. NetBeans
   - Java公式の統合開発環境（IDE）
   - 優れたGUIエディタを搭載

3. Javaドキュメント
   - Javaの便利な取扱い説明書



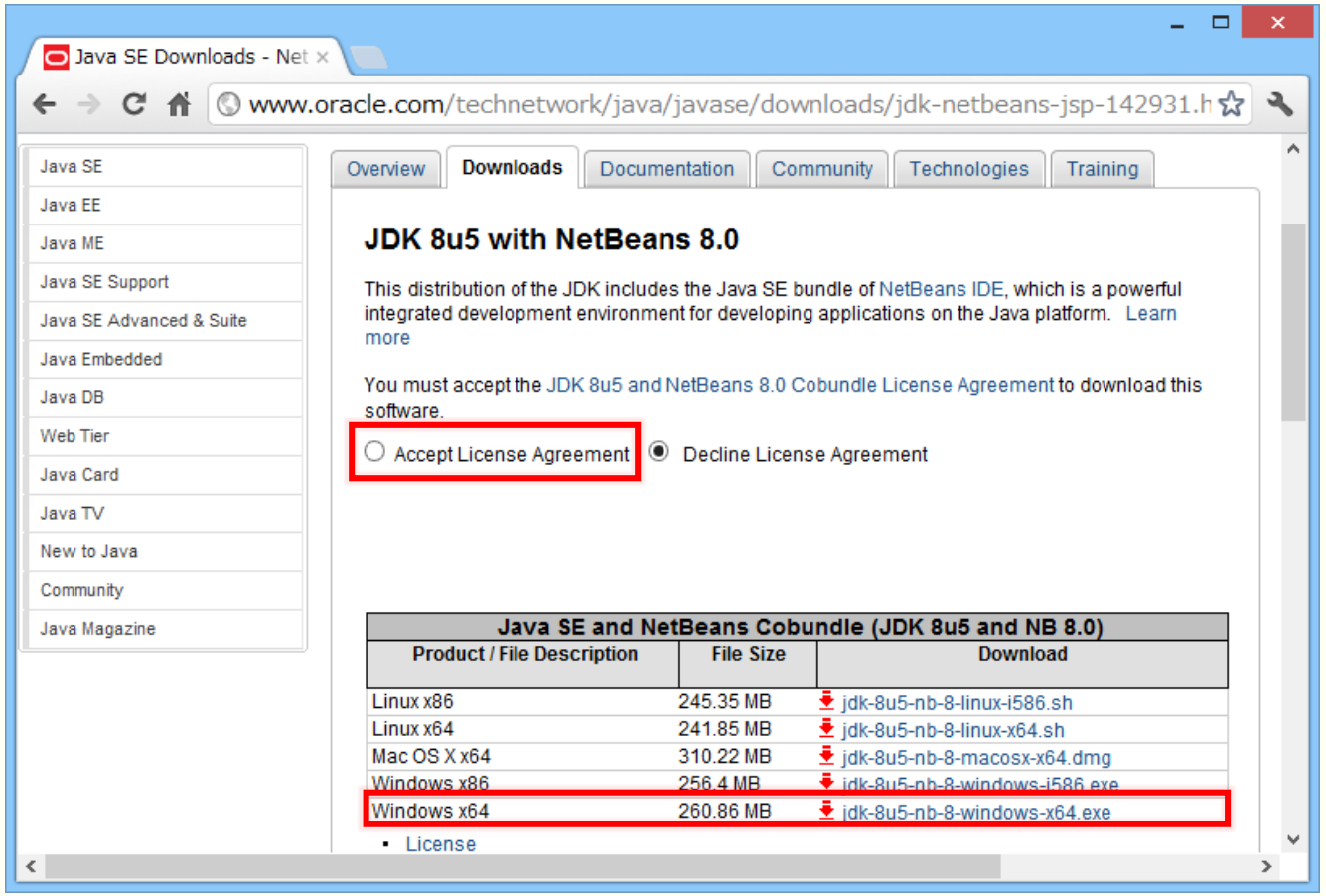
JDK/NetBeansのインストール①

1. 「http://www.oracle.com/technetwork/java/javase/downloads/index.html」にアクセス
2. 「JDK 8u5 & Netbeans 8.0」をクリック



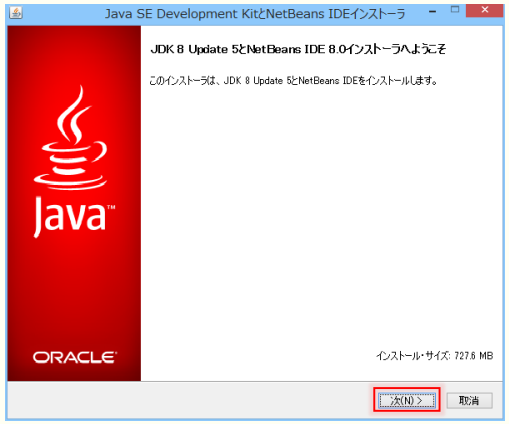
JDK/NetBeansのインストール②

3. 「Accept License Agreement」をチェック
4. 各自の環境に合ったものをダウンロード



JDK/NetBeansのインストール③

1. 「次へ」を押す



2. そのまま「次へ」を押す

3. 「Junitをインストール」をチェックして「次へ」を押す

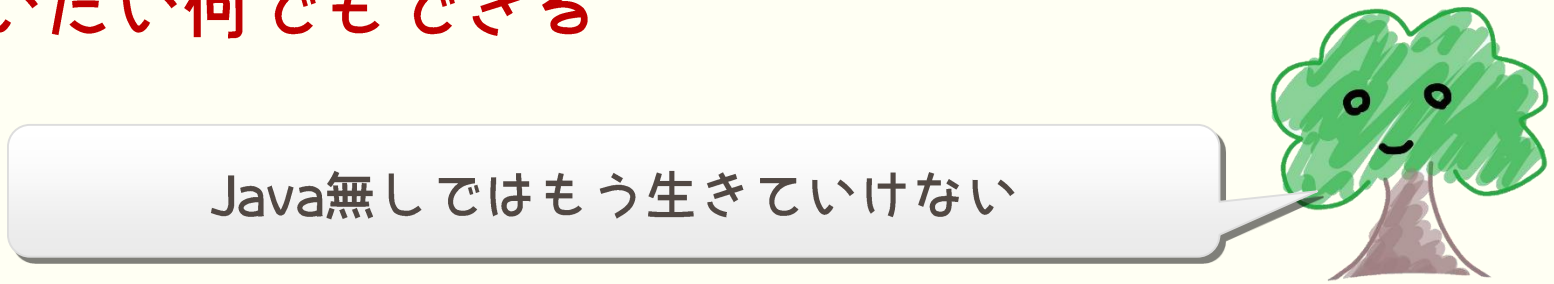
4. そのまま「次へ」を押す

5. 「更新の確認」のチェックを外して「次へ」を押す

6. 「データ提供」のチェックを外して「終了」を押す

Javaの特徴

- プログラミング言語
- C言語から派生（なので文法が似てる）
- ソフトウェアが作りやすい
- マルチプラットフォーム（どんなPCでも動く）
- オブジェクト指向
- だいたい何でもできる



C言語から引き継いだ文法

変数の型
- int型
- float型
- double型
- char型

制御文
- if文
- switch文
- for文
- while文

演算子
- 算術演算子
- 比較演算子
- 論理演算子

結構似た感じのソースコードが書けるよ

用語の簡単な説明

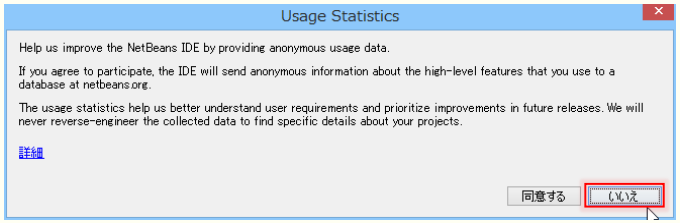
- メソッド
  - C言語の関数に相当
  - main関数と同じようにmainメソッドがある
- クラス
  - C言語の構造体にメンバとして関数を加えたようなもの
  - インスタンス化（実体化）しなければ使えない
  - mainクラスのmainメソッドが一番初めに実行される
- メンバ変数（フィールド）
  - C言語のグローバル変数に相当
  - クラス内のどこからでも参照できる変数
- ローカル変数
  - C言語のローカル変数に相当
  - 特定のメソッドの内部でのみ有効

NetBeansの起動

1. アイコンをクリック

2. すこし待つ

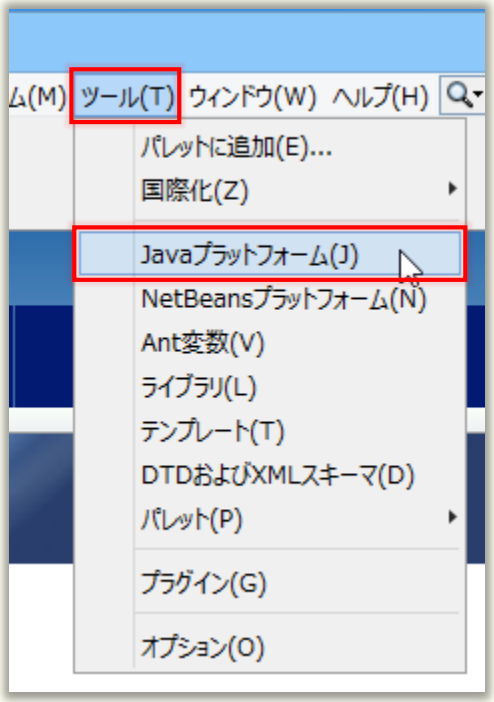
3. データ提供は「いいえ」を選択



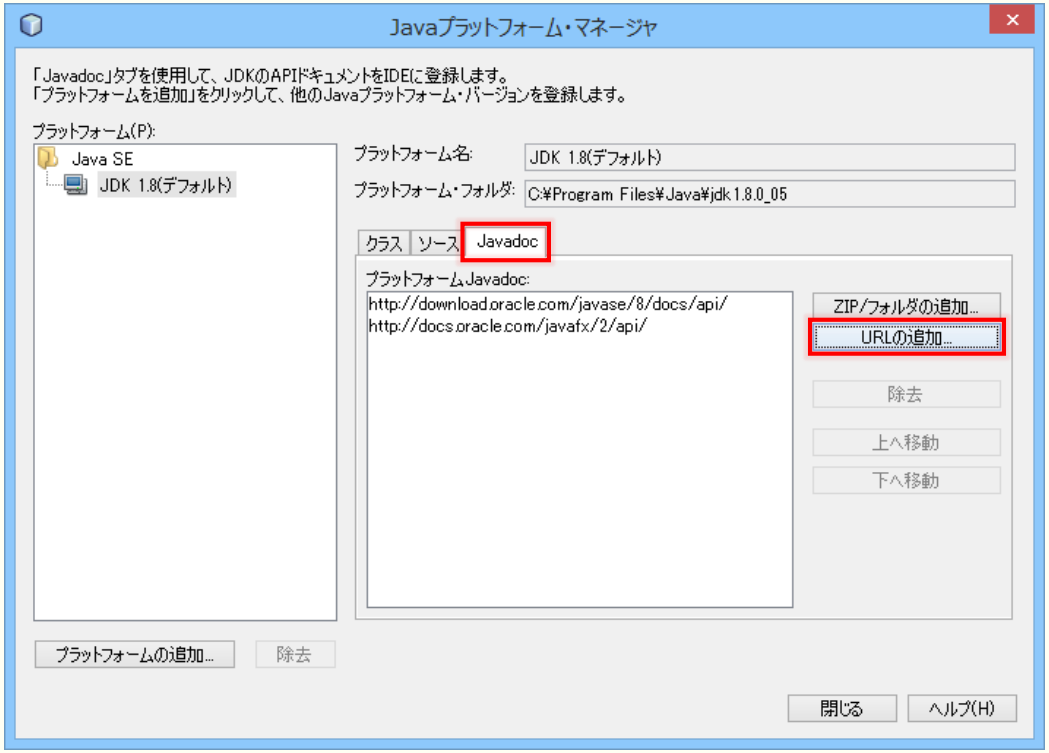
4. NetBeangの起動！

JavaDocの組み込み

1. [ツール]→[Javaプラットフォーム]と進む



2. [Javadoc]タブを選択し、[URLの追加]ボタンを押す

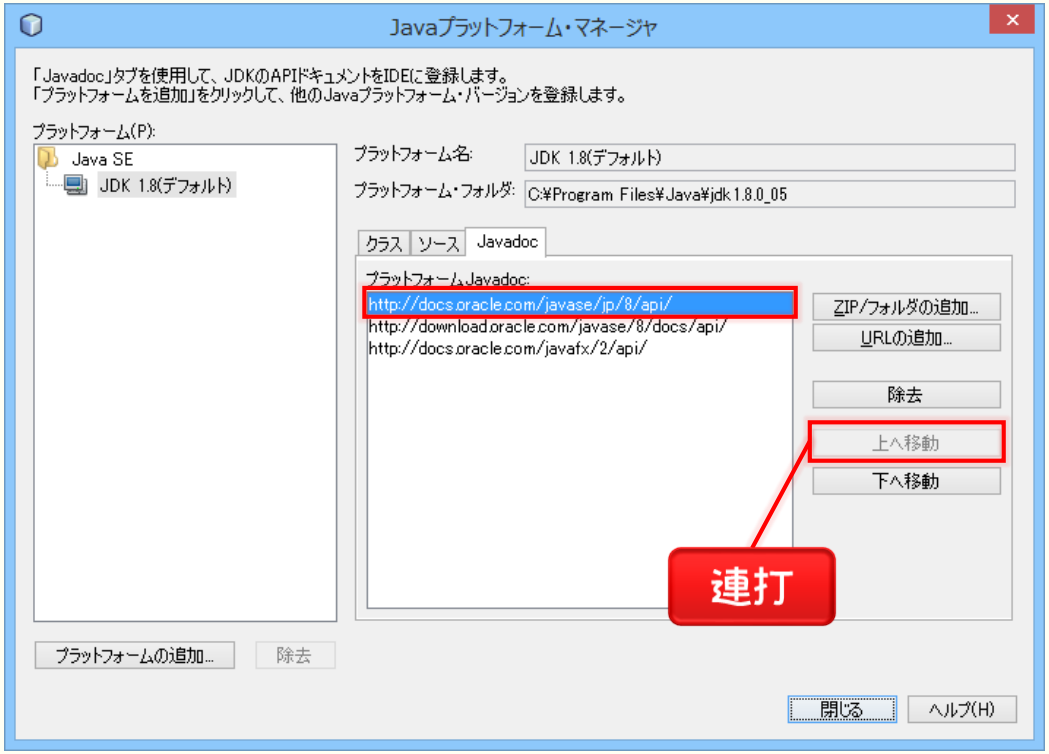


JavaDocの組み込み

3.
「http://docs.oracle.com/javase/jp/8/api/」
を入力し、[URLの追加]ボタンを押す

英語が得意なら
この操作は必要ナシ

4. [上へ移動]ボタンを押して、
一番上まで移動させる

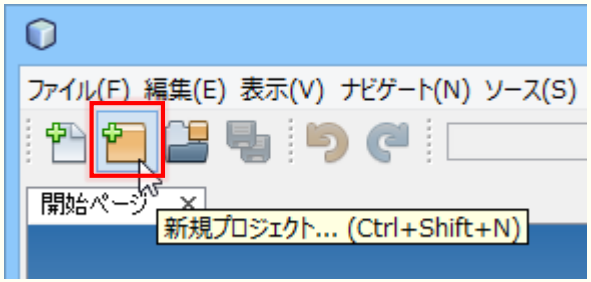


2. 基本的なプログラミング

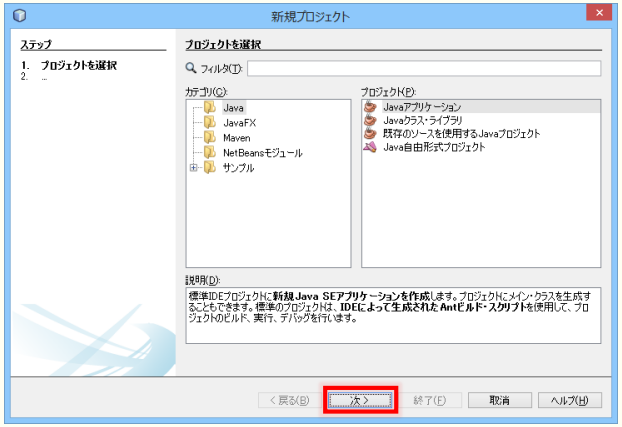
新規プロジェクト
「Lecture1」を作ろう

プロジェクトの作成

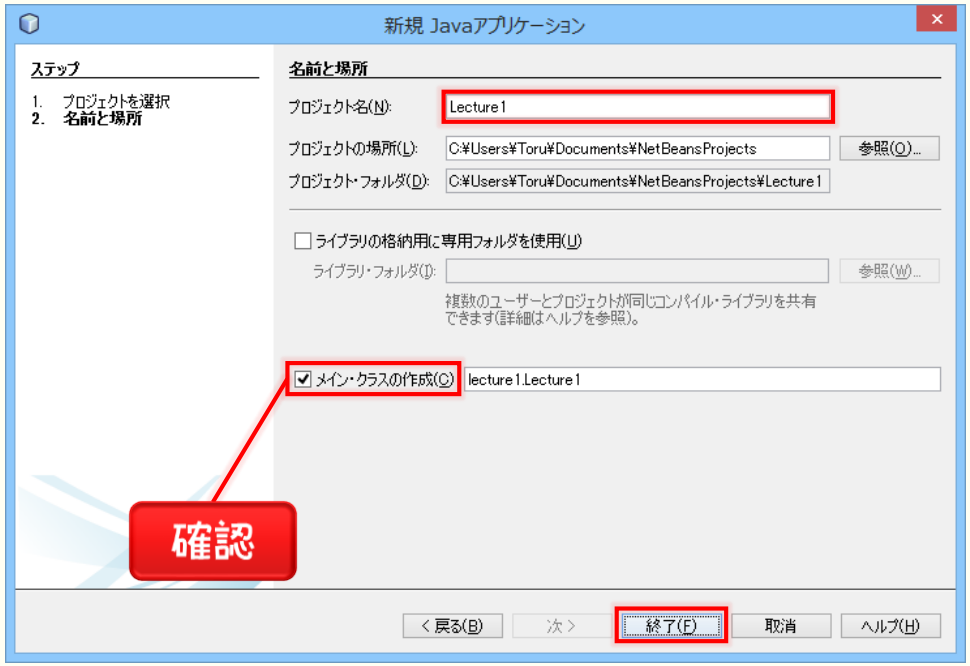
1. [新規プロジェクト...]
ボタンを押す



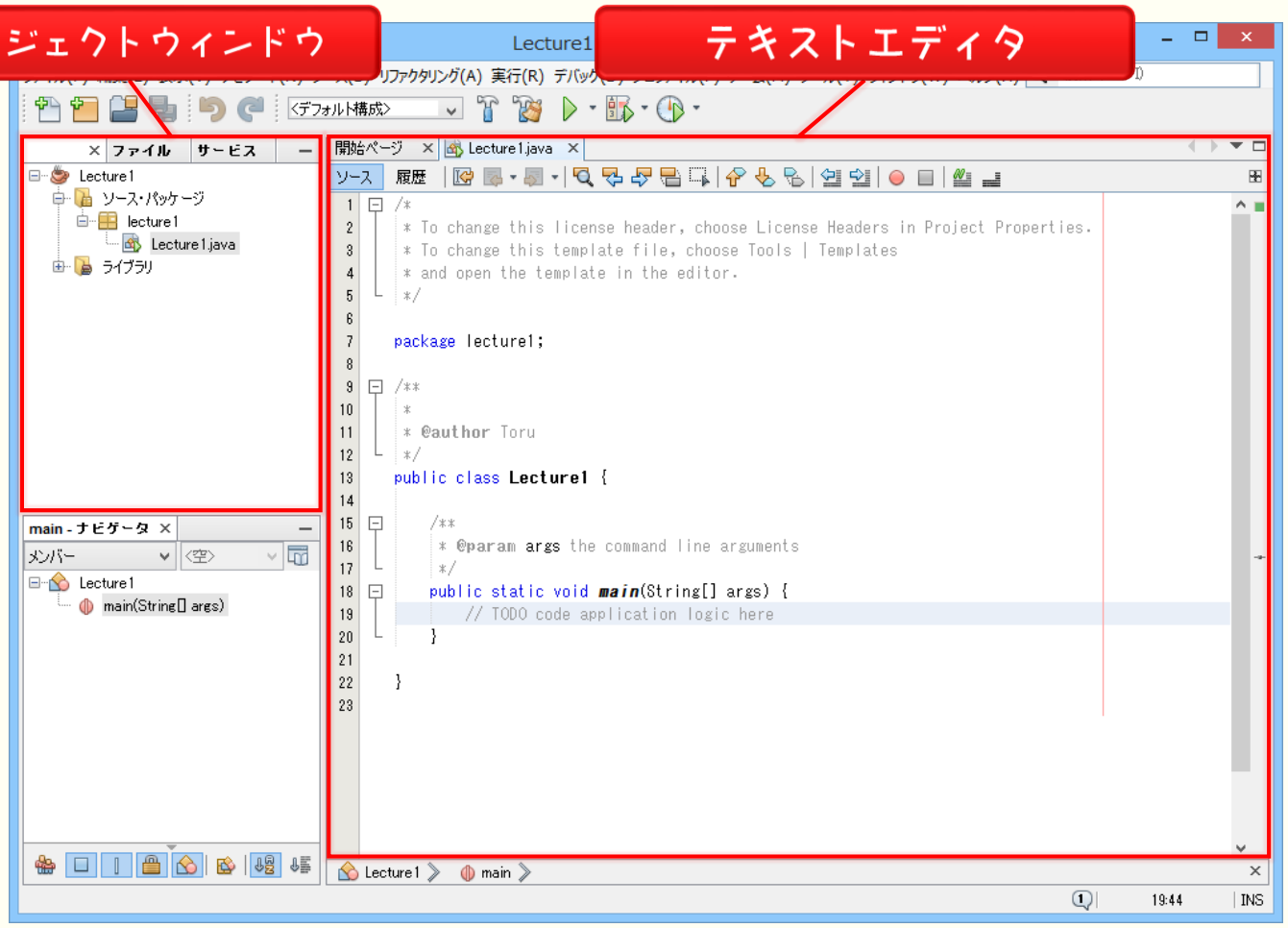
2. そのまま[次へ]ボタンを押す



3. プロジェクト名（Lecture1）をつけ、
[終了]ボタンを押す



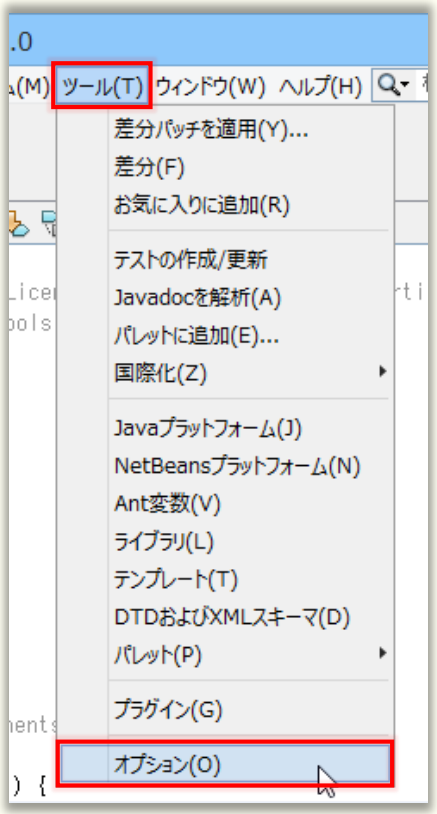
プロジェクトの作成



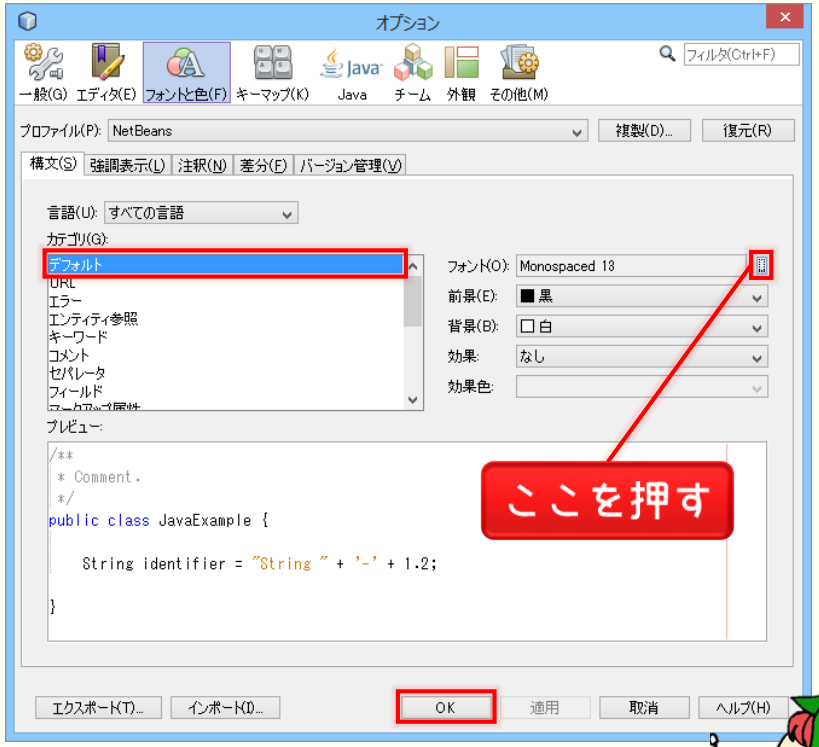
/*　・・・　*/で囲まれた部分と、//ではじまる行は
プログラムを読みやすくするためのコメント

フォント変更のすすめ

1. [ツール]→[オプション]と進む



2. フォントを変更して[OK]ボタンを押す



フォントは「メイリオ」のサイズ「18」がおすすめ

プロジェクト作成のすすめ

時々左上にこんなものを表示します。

新規プロジェクト
「Lecture1」を作ろう

新しく作ったほうが
手っ取り早いとき

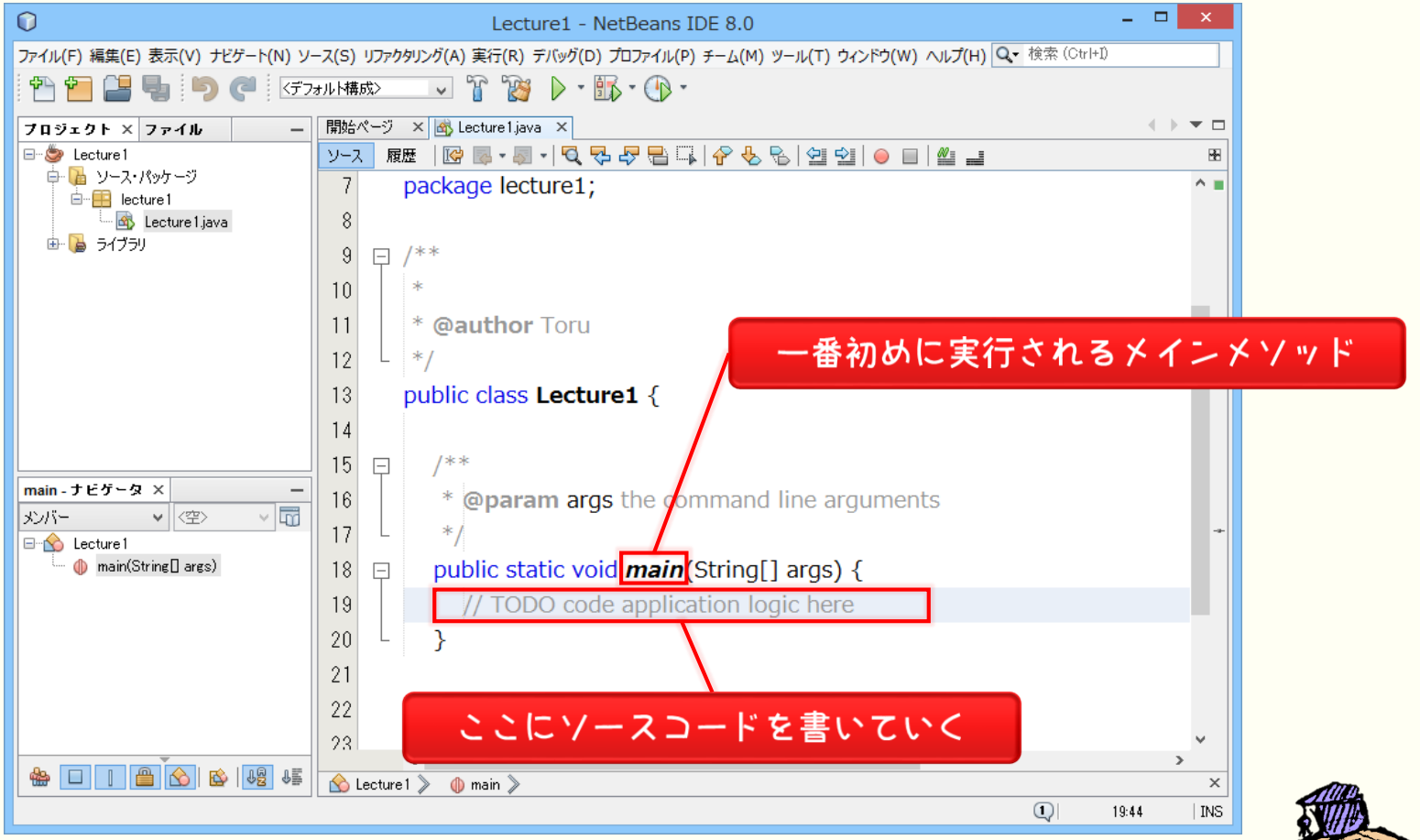
既存プロジェクト
「Lecture1」を使おう

あるものを使ったほうが
手っ取り早いとき

覚えたか？

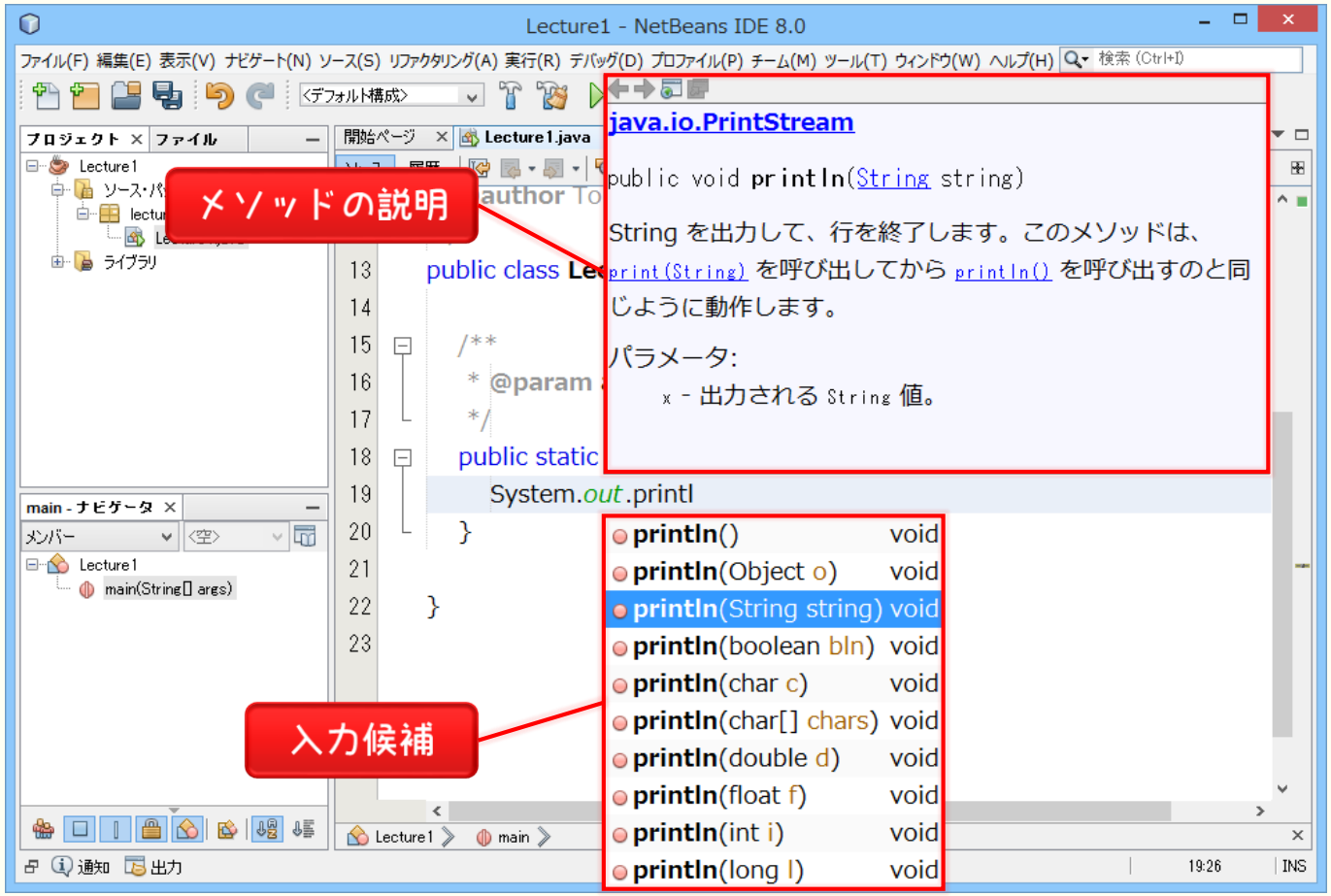
既存プロジェクト
「Lecture1」を使おう

Hello NetBeans!



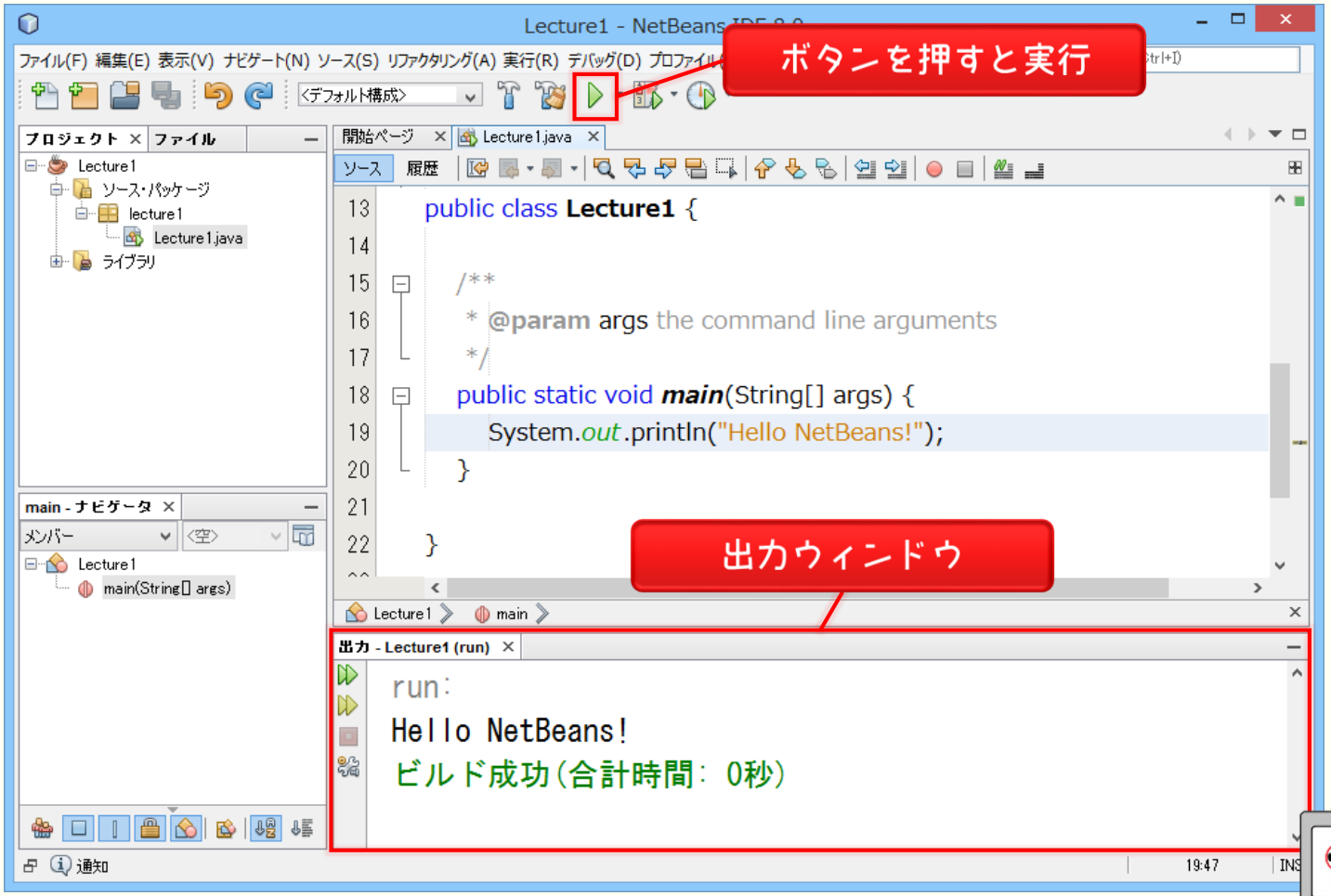
「System.out.println(“Hello NetBeans!”)」と書いてみよう

Hello NetBeans!



[Ctrl] + [スペース]で強制的に候補を出せる

Hello NetBeans!



「Hello NetBeans!」と表示されたら成功！

2.1 データの入れ物「変数」

データの入れ物「変数」

変数とは

［ データ型 ］［ 変数名 ］;

の形で宣言される、数値や文字を入れておく入れ物です。

データの型	種類	サイズ	扱える範囲
short	整数	2 バイト	−32786~32767
int	整数	4 バイト	−2147483648~2147483647
long	整数	8 バイト	−9223372036854775808~9223372036854775807
float	浮動小数点数	4 バイト	有効数字7桁 $\pm 10^{-38}$~10^{38}
double	浮動小数点数	8 バイト	有効数字15桁 $\pm 10^{-308}$~10^{308}
char	文字定数	2 バイト	Unicode文字（一文字）
String	文字列	不定	大体どんだけでも
boolean	論理値	1 バイト	true, false

既存プロジェクト
「Lecture1」を使おう

変数の使い方

```
public static void main(String[] args) {
    int a;
    double b;
    String s;

    a = 42;
    b = 3.1415926535;
    s = "Hello, NetBeans 8.0";

    System.out.println("a の値は " + a);
    System.out.println("b の値は " + b);
    System.out.println("s の値は " + s);
}
```

変数の宣言

値の代入

値の表示

int a = 42; のように、宣言と代入は同時にできる

新規プロジェクト
「Lecture2」を作ろう

色々な計算

```
public static void main(String[] args) {
        int a = 16, b = 4;
        double x = 2.5, y = 8.3;

        // 四則演算
        System.out.println("a = " + a + ", b = " + b);
        System.out.println("a + b = " + (a + b));
        System.out.println("a - b = " + (a - b));
        System.out.println("a * b = " + (a * b));
        System.out.println("a / b = " + (a / b));
        System.out.println();

        // Mathクラス
        System.out.println("x = " + x + ", y = " + y);
        System.out.println("x ^ y = " + Math.pow(x, y));
        System.out.println("sin(x - y) = " + Math.sin(x - y));
        System.out.println("log(x + y) = " + Math.log(x + y));
}
```

「+」で変数や文字を繋げられる

様々な計算が可能なMathクラス

「Math.」と打ち込んで候補を観察してみよう

新規プロジェクト
「Lecture3」を作ろう

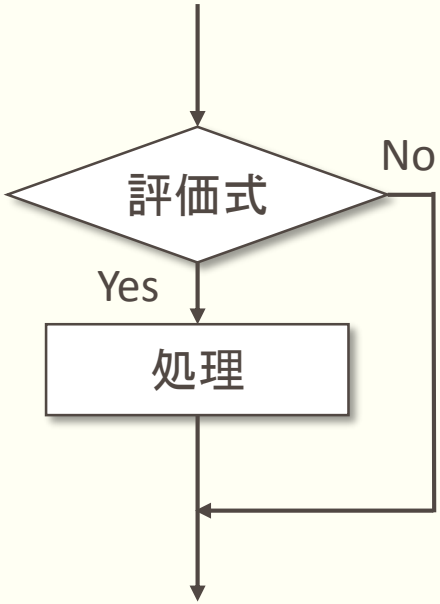
if文

if（**評価式**）{ **処理** }

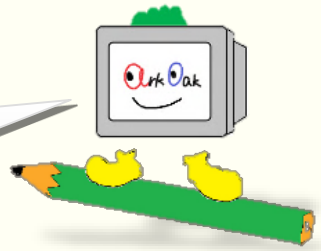
の形で記述され、評価式の結果が正しければ処理が実行されます。

```
public static void main(String[] args) {
    int a = 0;

    if (a == 0) {
        System.out.println("a は 0 だよ");
    } else {
        System.out.println("a は 0 ではないよ");
    }
}
```



プログラムの分岐を作る超重要な文法！



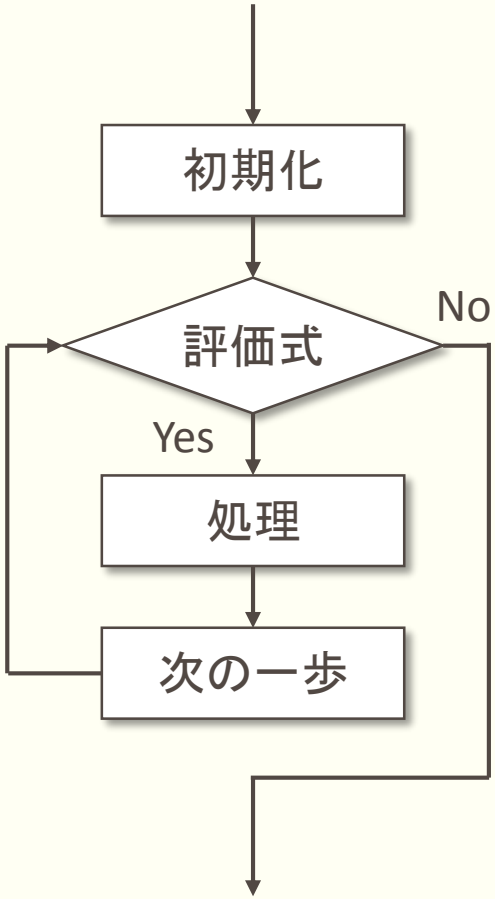
新規プロジェクト
「Lecture4」を作ろう

for文

for（初期化；評価式；次の一歩）｛ 処理 ｝

の形で記述され、次の流れで実行されます。

1. 初期化を行う。
2. 評価式が正しければ、処理を実行する（違えば終了）。
3. 処理が終わったら、次の一歩を行う。
4. 手順2に戻る。



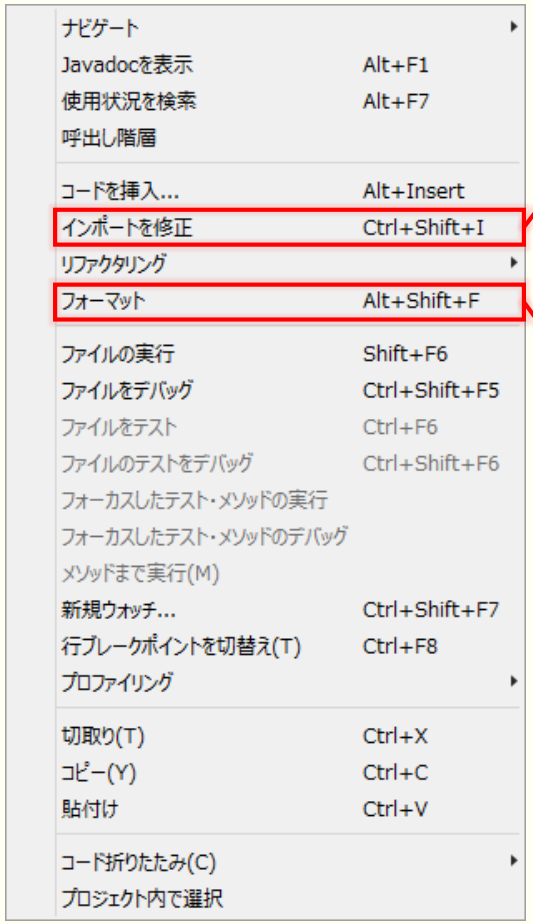

```
public static void main(String[] args) {
        int sum = 0;
        for (int i = 1; i <= 10; i++) {
                sum = sum + i;
        }
        System.out.println("1から10までの和は, " + sum);
}
```

ちょっと一息

わんわんおーく

テキストエディタのどこかで右クリックしてみよう

アー君

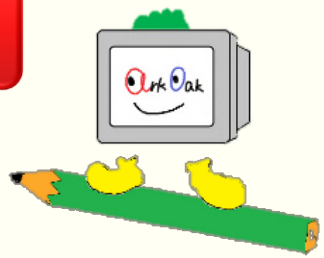


自動的に必要なライブラリをインポートしてくれる機能

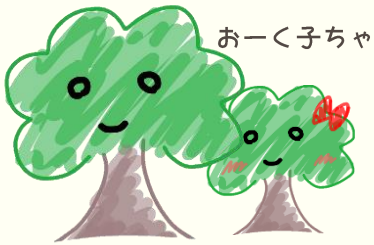
自動的にソースコードを整えてくれる機能

プログラミング小僧

樫木筆夫



おーく君



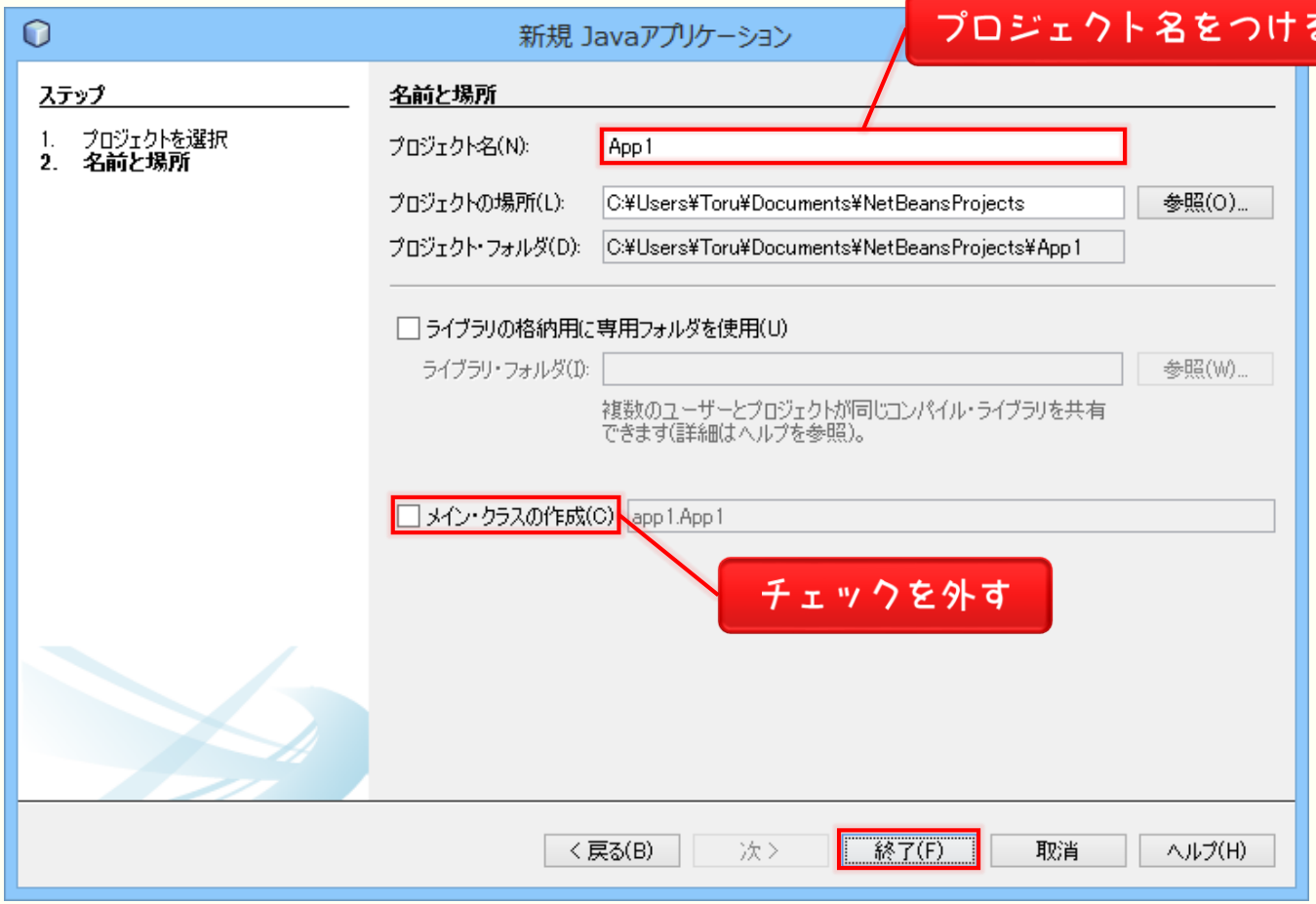
あーくちゃん

Mr.O

3. ソフトウェアの作り方

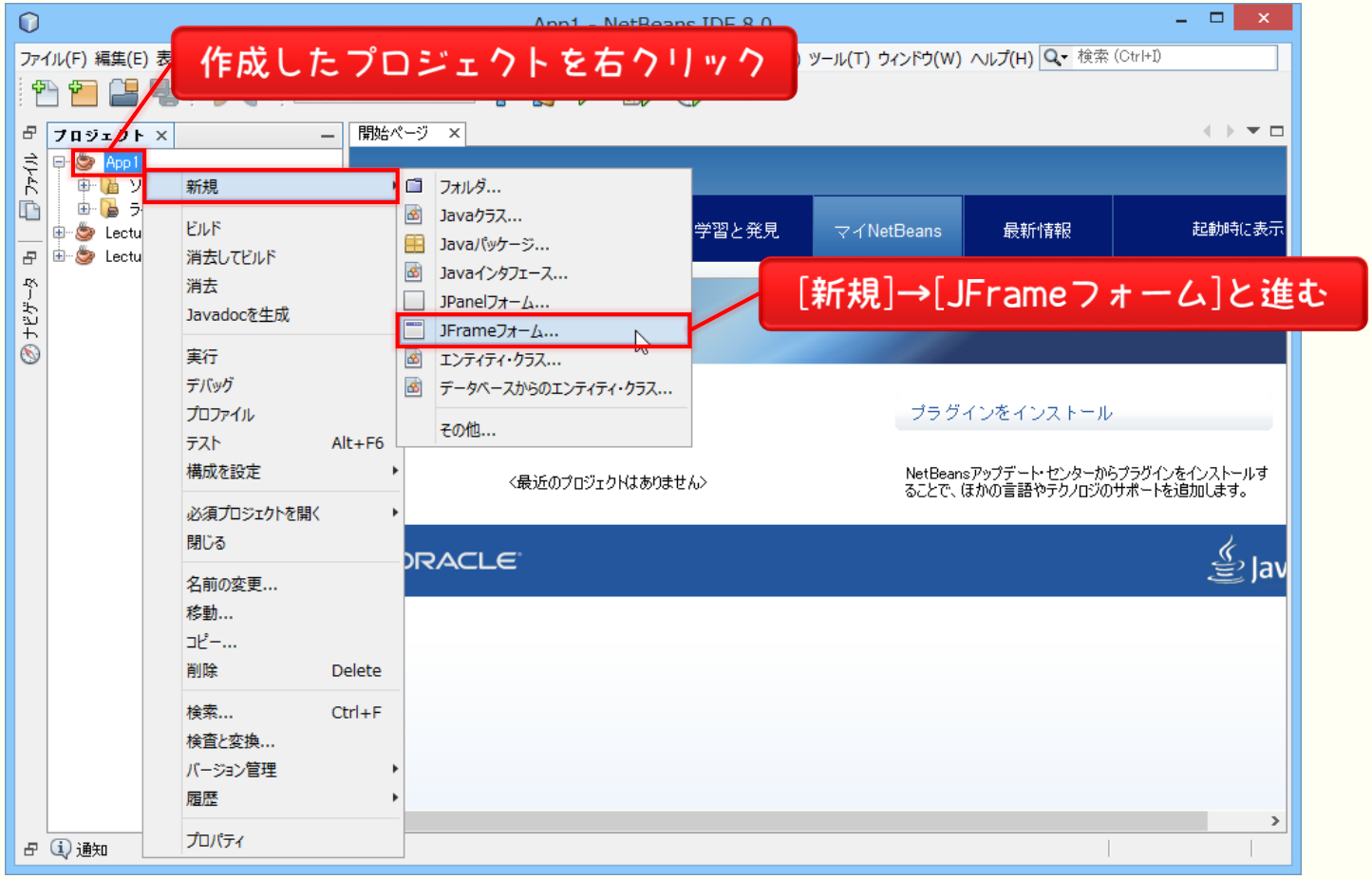
新規プロジェクト
「App1」を作ろう

プロジェクトの作成



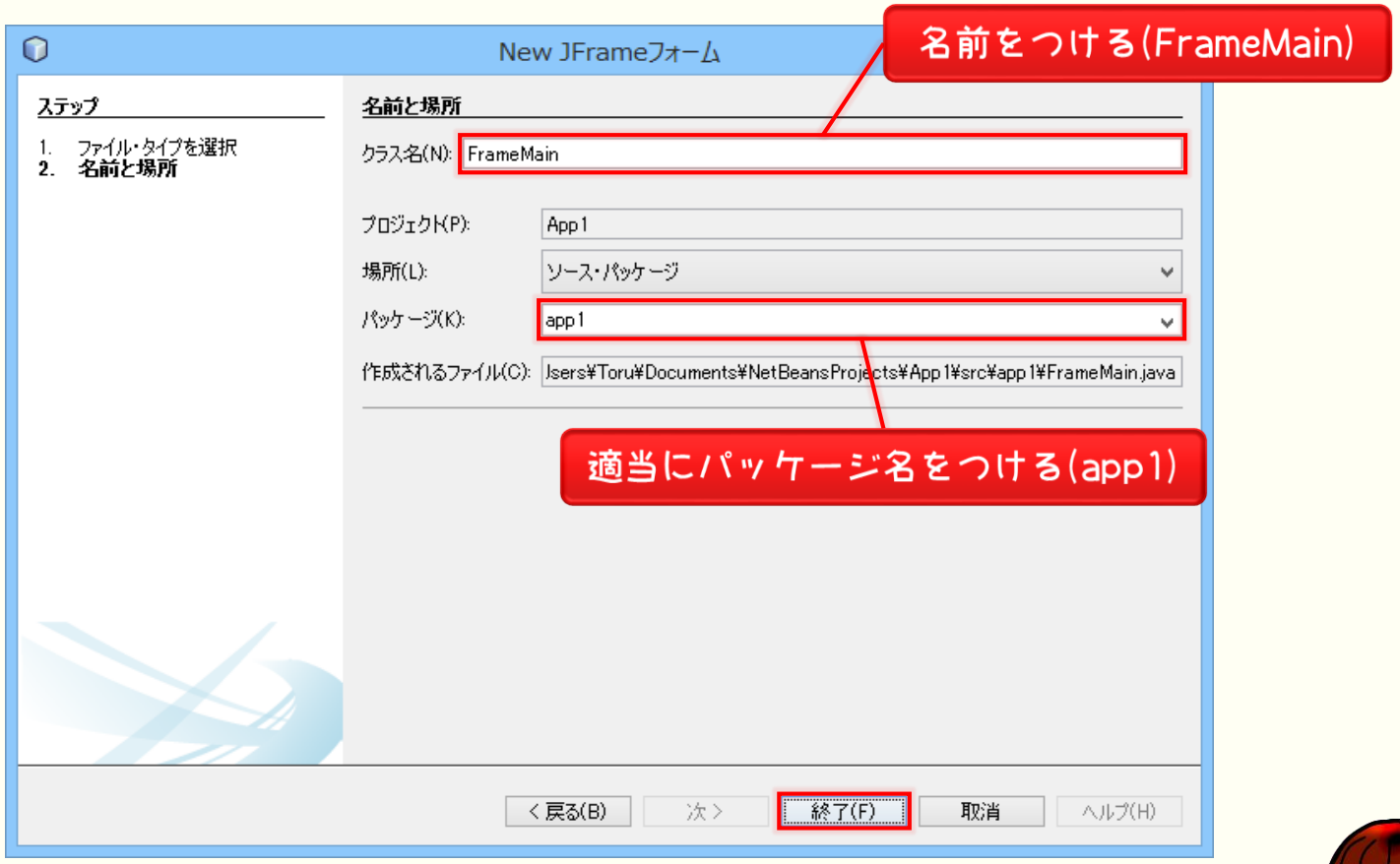
今回からはメインクラスを自分で作る

JFrameフォームの作成



メインクラスとなるJFrameフォームの作成

JFrameフォームの作成

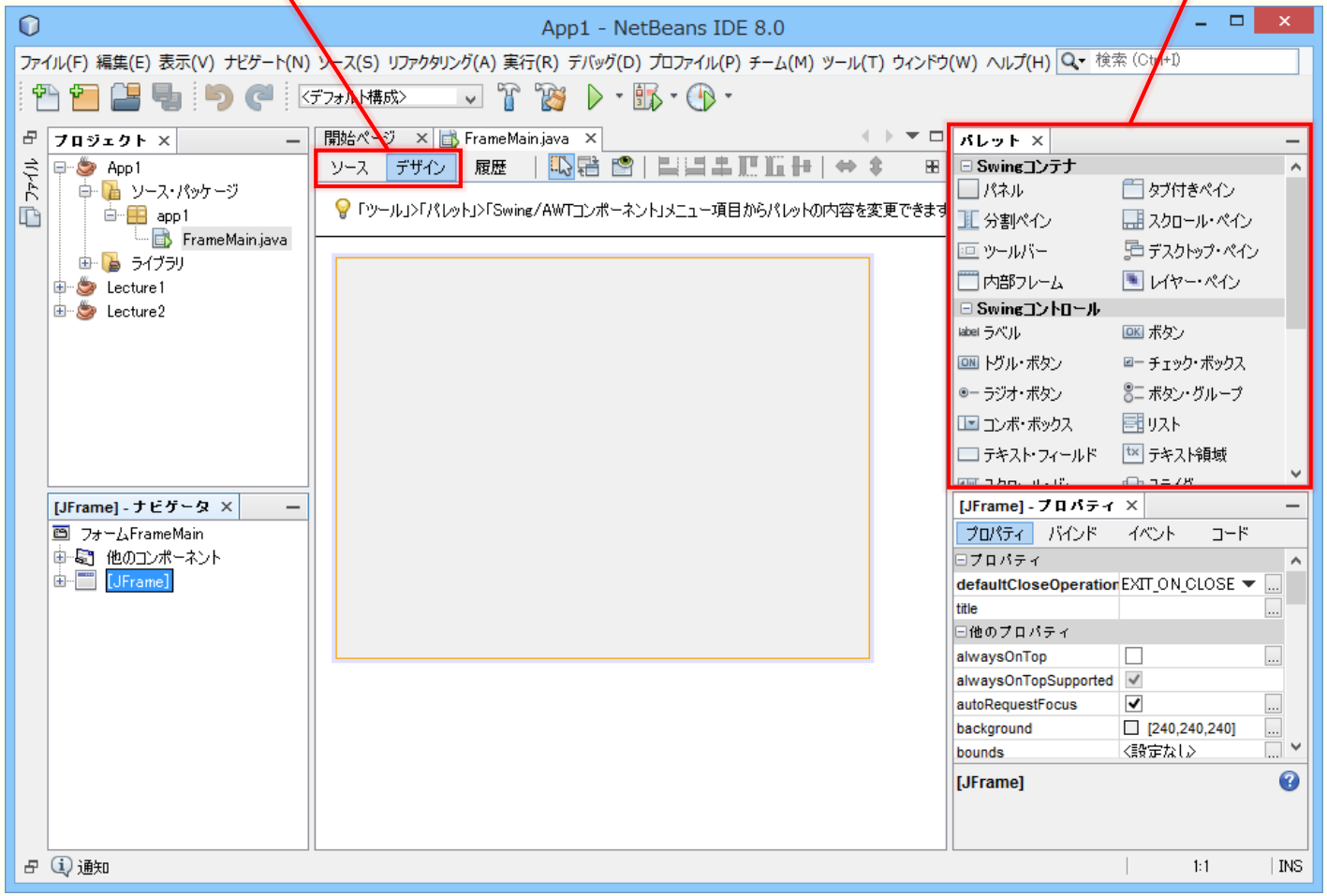


将来に備えてパッケージ名をつける習慣をつけておくと吉

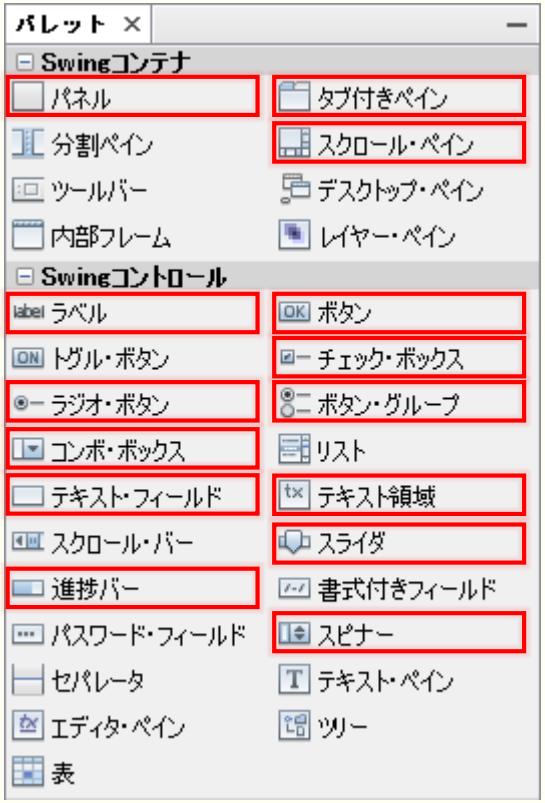
JFrameフォームの作成

ソースコードをGUIエディタを切り替えられる

GUI部品のパレット



よく使うGUI部品
(Swingコントロール)

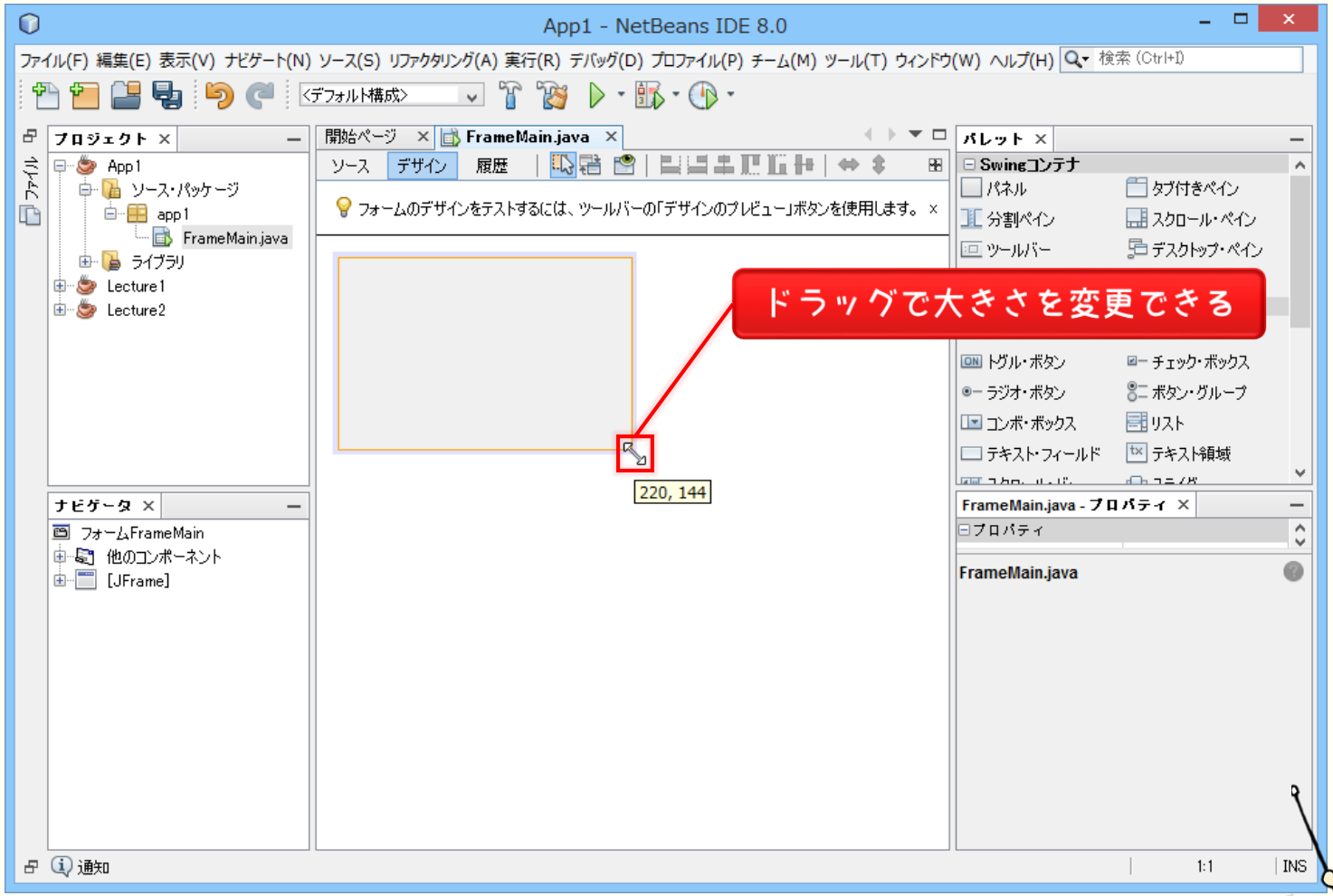


ArkOakホームページに使い方が載っています。
http://arkoak.com/java/

今回使う重要なのを幾つかピックアップして説明するよ

3.1 ボタンの使い方

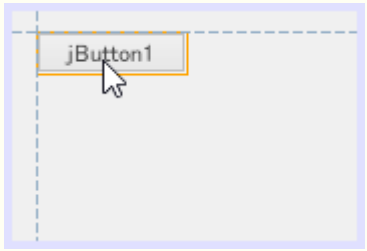
ボタンの使い方



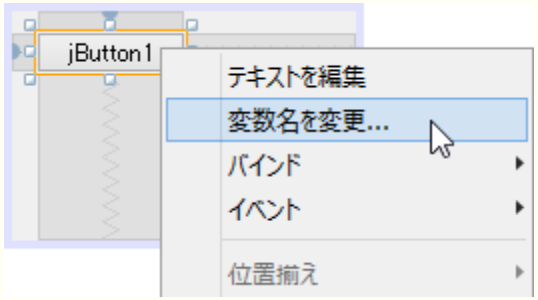
好きなサイズに変更しよう

ボタンの使い方

1. ボタンの配置

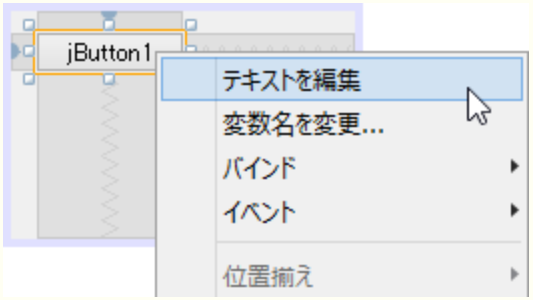


2. 右クリックして「変数名を変更」

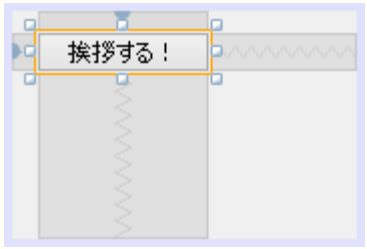


3. 名前をつける(btnHello)

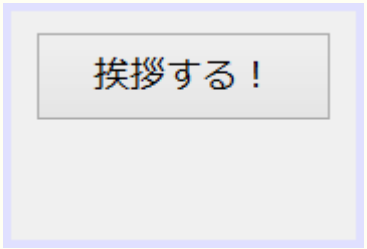
4. 右クリックして「テキストを編集」



5. テキストを入力する



6. ボタン準備完了！



名前の付け方のすすめ

GUI部品の変数名は、

GUI部品名 ＋ 機能説明

といった感じでつける！

「例」

- 開始ボタン → btnStart
- アニメ開始ボタン → btnStartAnimation
- 終了ボタン → buttonEnd
- 描画ラベル → lblDraw
- 設定コンボボックス → cmbSetting
- 出力用テキスト領域 → taOutput

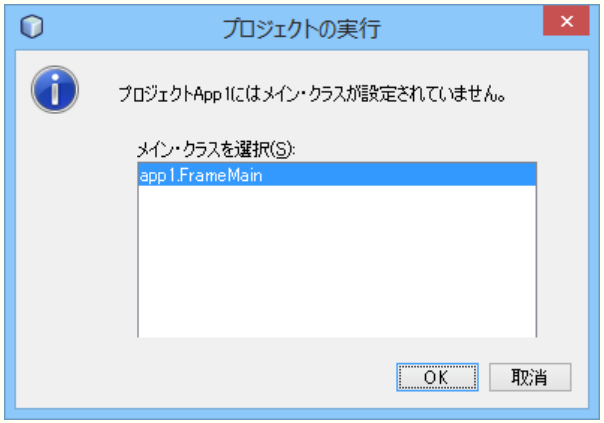
変数名の頭は小文字、単語の切れ目で大文字にする習慣

ボタンの使い方

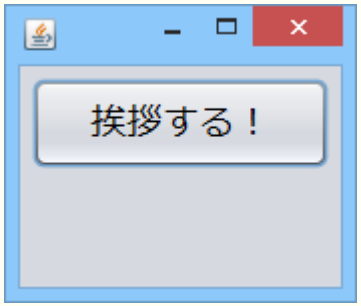
7. ボタンを押したときの処理を書く

```
private void btnHelloActionPerformed(java.awt.event.ActionEvent evt) {
        System.out.println("こんにちは！");
}
```

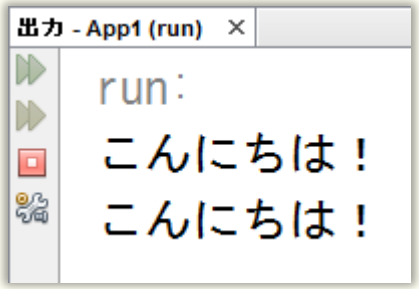
8. プロジェクトを実行する



9. 実行結果



10. ボタンを押すと・・・



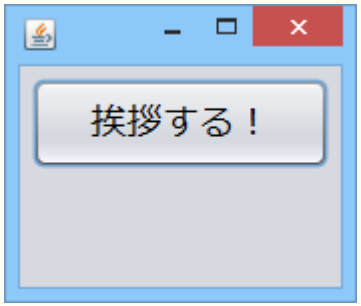
出力画面は開発者にしか見えない

ボタンの使い方

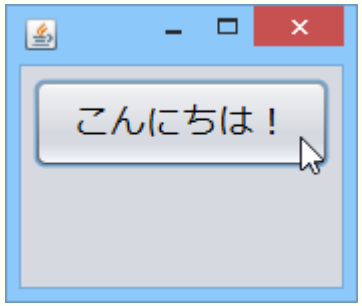
ボタンの処理を以下のように書き換えてみましょう

```
private void btnHelloActionPerformed(java.awt.event.ActionEvent evt) {
        btnHello.setText("こんにちは！");
}
```

ボタンを押すと・・・



こんにちは！

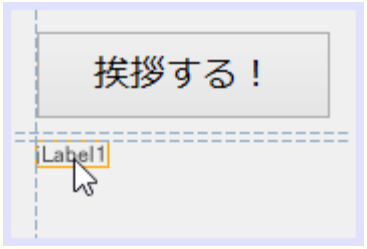


[GUI部品名].setText()で、その部品のテキストを変更できるよ

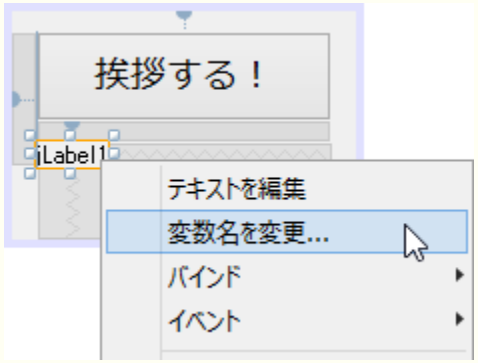
3.2 ラベルの使い方

ラベルの使い方

1. ラベルの配置

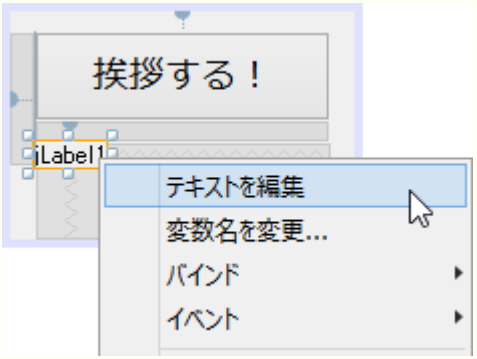


2. 右クリックして「変数名を変更」



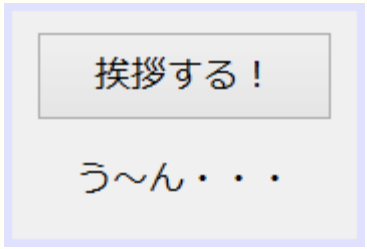
3. 名前をつける(lblHello)

4. 右クリックして「テキストを編集」



5. テキストを入力する

6. ボタン準備完了！

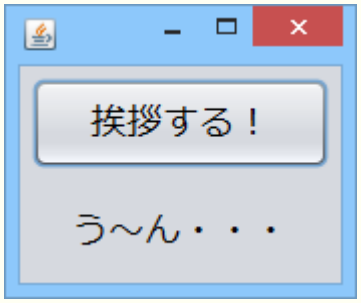


ラベルの使い方

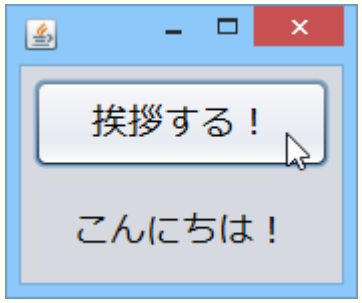
7. ボタンを押したときの処理を編集する

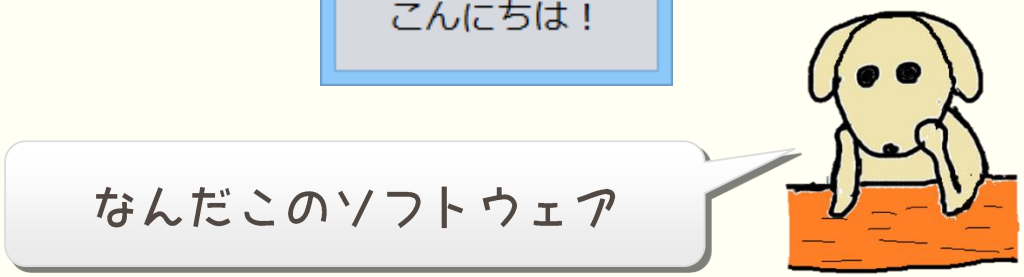
```
private void btnHelloActionPerformed(java.awt.event.ActionEvent evt) {
        lblHello.setText("こんにちは！");
}
```

9. 実行結果



10. ボタンを押すと・・・





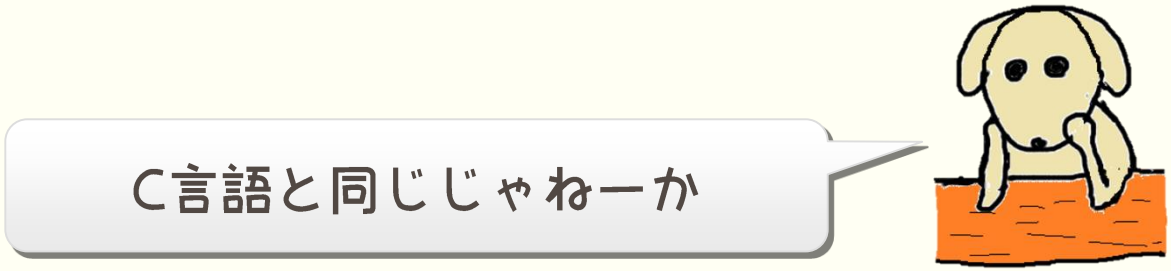
ファイル入出力

Javaのファイル操作

1. ファイルを開く

2. ファイルの読み書き

3. ファイルを閉じる



ファイル入出力クラス一覧

文字ストリーム

役割	入力	出力
抽象スーパークラス	Reader	Writer
簡易読み込み	FileReader	FileWriter
バイト単位で文字コード指定	InputeStreamReader	OutputStreamWriter
行単位での読み書き	BufferedReader	BufferedWriter
出力と入力のパイプ処理	PipedReader	PipedWriter
Stringからの読み書き	StringReader	StringWriter
フィルタリング	FilterReader	FilterWriter
印字出力用		PrintWriter

バイトストリーム

役割	入力	出力
抽象スーパークラス	InputStream	OutputStream
ファイルの読み書き	FileInputStream	FileOutputStream
マシンに依存しない読み書き	DataInputStream	DataOutputStream
オブジェクトの読み書き	ObjectInputStream	ObjectOutputStream
フィルタリング	FilteredInputStream	FilteredOutputStream
バッファを用いた読み書き	BufferedInputStream	BufferedOutputStream
入力と出力のパイプ処理	PipedInputStream	PipedOutputStream
圧縮ファイルの読み書き	ZipInputStream	ZipOutputStream
印字出力用		PrintStream

4. テキストファイルの入出力

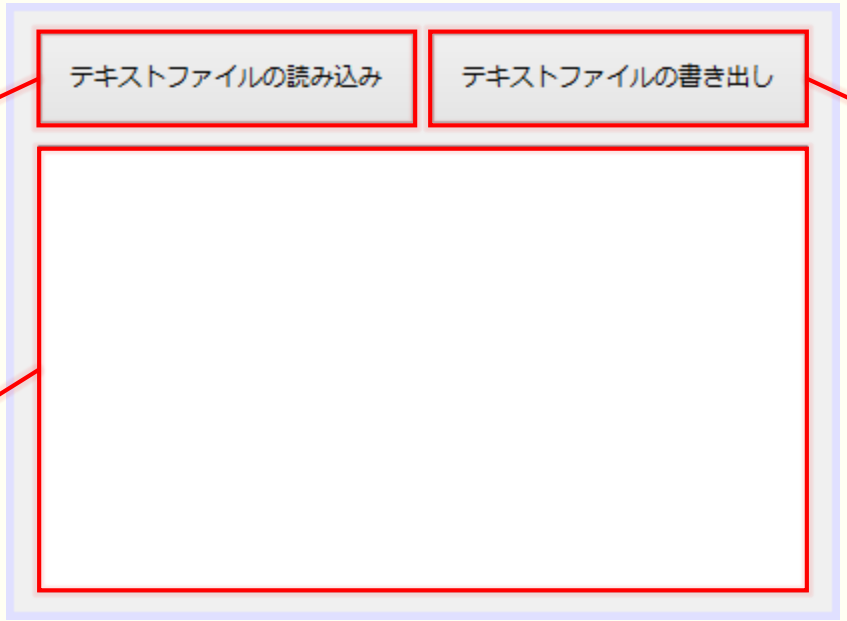
新規プロジェクト
「TextIO」を作ろう

テキストファイルの入出力

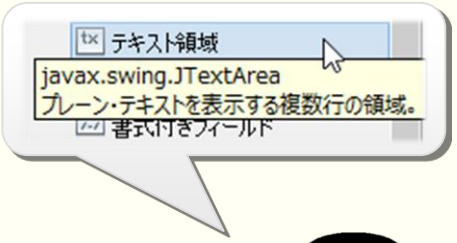
次のようにGUI部品を配置しましょう

GUI：JButton
変数名：btnRead

GUI：JTextArea
変数名：taOutput



GUI：JButton
変数名：btnWrite



パレットのGUI部品にカーソルを合わせると
その部品の本来の名前を見ることができる

既存プロジェクト
「TextIO」を使おう

ファイルの存在確認

「テキストファイルの読み込み」ボタンをダブルクリックして
以下のように処理を記述しましょう

ファイルオープン（インスタンス生成）

```
private void btnReadActionPerformed(java.awt.event.ActionEvent evt) {
        File file = new File("in.txt");
        if (file.exists()) {
                taOutput.append("ファイルは存在します。¥n");
        } else {
                taOutput.append("ファイルが見つかりません。¥n");
        }
}
```

ファイルが存在していたら実行される

テキスト領域に文字を追加するメソッド
（改行するためには ¥n が必要）

"C:¥¥Users¥¥Toru¥¥Documents¥¥NetBeansProjects¥¥TextIO¥¥in.txt"
といった具合に絶対パスを指定してもOK。¥は2個つけることに注意。

既存プロジェクト
「TextIO」を使おう

テキストファイルの読み込み

「テキストファイルの読み込み」ボタンの処理を
以下のように書き換えてみましょう

```
private void btnReadActionPerformed(java.awt.event.ActionEvent evt) {
        try {
                // ファイルオープン
                BufferedReader br = new BufferedReader(new FileReader("in.txt"));
                // ファイルの読み込み
                taOutput.setText("");
                for (String line; (line = br.readLine()) != null;) {
                        taOutput.append(line + "¥n");
                }
                // ファイルクローズ
                br.close();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

読み込むファイルのパスを指定する

テキスト領域の中身を空にする

行が読み込めなくなるまで
1行ずつテキスト領域に出力する

ファイルエラーが発生した場合
エラー内容を出力する

既存プロジェクト
「TextIO」を使おう

文字コードを指定した読み込み

```
private void btnReadActionPerformed(java.awt.event.ActionEvent evt) {
        try {
                // ファイルオープン
                FileInputStream fis = new FileInputStream("in.txt");
                InputStreamReader isr = new InputStreamReader(fis, "UTF-8");
                BufferedReader br = new BufferedReader(isr);
                // ファイルの読み込み
                taOutput.setText("");
                for (String line; (line = br.readLine()) != null;) {
                        taOutput.append(line + "¥n");
                }
                // ファイルクローズ
                br.close();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

バイトストリームとして読み込む

文字コードを指定できる
(UTF-8, Shift-JIS, EUC-JP)

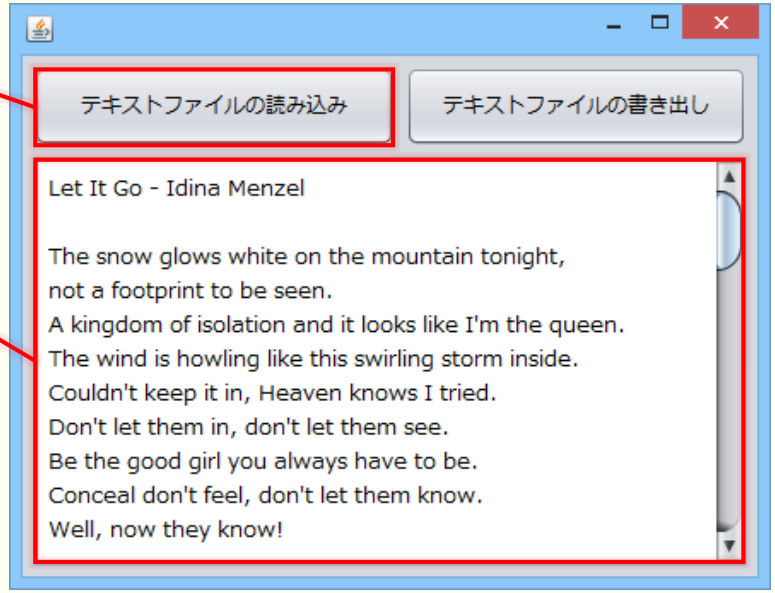
既存プロジェクト
「TextIO」を使おう

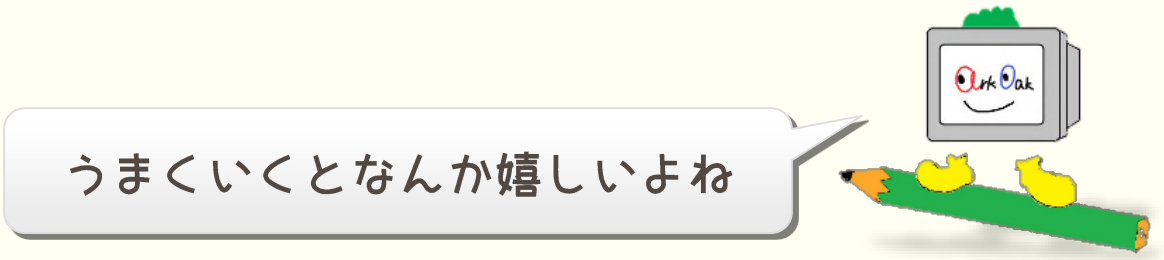
テキストファイルの読み込み

プロジェクトフォルダ内部に
「in.txt」というファイルを置いて実行しましょう

クリックすると・・・

「in.txt」の中身が
読み込まれる





既存プロジェクト
「TextIO」を使おう

テキストファイルの書き出し

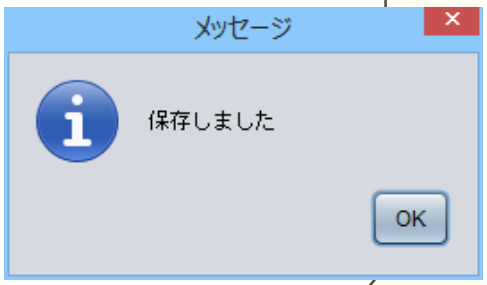
「テキストファイルの書き出し」ボタンをダブルクリックして
以下のように処理を記述しましょう

```
private void btnWriteActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // ファイルオープン
        PrintWriter pw = new PrintWriter("out.txt");
        // ファイルの書き込み
        pw.print(taOutput.getText());
        // ファイルクローズ
        pw.close();
        JOptionPane.showMessageDialog(this, "保存しました");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

書き出すファイル名を指定する

テキスト領域の中身を取得して書きだす
(printf()を用いた書式付出力も可能)

メッセージダイアログの表示



4.1 ファイル選択ダイアログ

既存プロジェクト
「TextIO」を使おう

ダイアログを用いた読み書き

次のようにGUI部品を配置しましょう

入出力ファイル変更の度にソースを変更するのは面倒･･･

ダイアログを用いた読み込み

「ダイアログで読み込み」ボタンをダブルクリックして
以下のように処理を記述しましょう

```
private void btnReadDialogActionPerformed(ja
        // ファイル選択ダイアログの生成
        JFileChooser fc = new JFileChooser(System.getProperty("user.dir"));
        if (fc.showOpenDialog(this) != JFileChooser.APPROVE_OPTION) return;
        try {
                // ファイルオープン
                BufferedReader br = new BufferedReader(new FileReader(fc.getSelectedFile()));
                // ファイルの読み込み
                taOutput.setText("");
                for (String line; (line = br.readLine()) != null;) {
                        taOutput.append(line + "¥n");
                }
                // ファイルクローズ
                br.close();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

ファイル読込ダイアログの表示

選択されたファイルのパスを取得

「System.getProperty(“user.dir”)」は実行中ファイルのフォルダパス
ここを変更すれば、ダイアログの初期フォルダを変更できる

ダイアログを用いた書き出し

「ダイアログで書き出し」ボタンをダブルクリックして
以下のように処理を記述しましょう

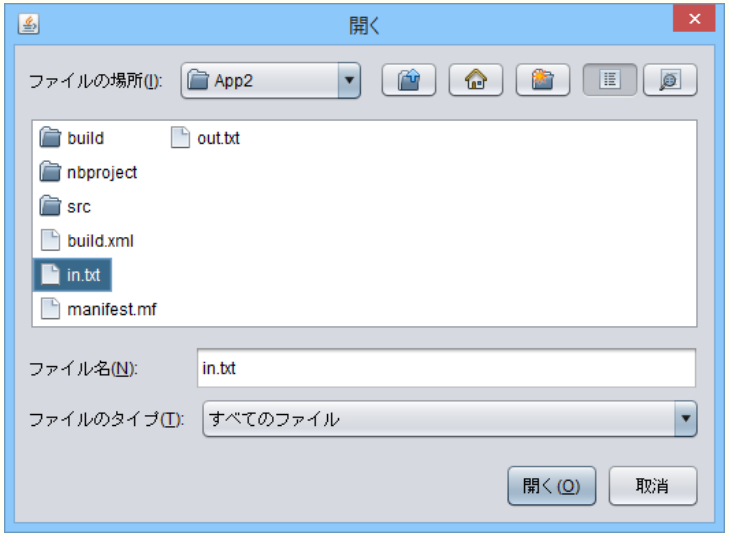
```
private void btnWriteDialogActionPerformed(ja
        // ファイル選択ダイアログの生成
        JFileChooser fc = new JFileChooser(System.getProperty("user.dir"));
        if (fc.showSaveDialog(this) != JFileChooser.APPROVE_OPTION) return;
        try {
                // ファイルオープン
                PrintWriter pw = new PrintWriter(fc.getSelectedFile());
                // ファイルの書き込み
                pw.print(taOutput.getText());
                // ファイルクローズ
                pw.close();
                JOptionPane.showMessageDialog(this, "保存しました");
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

ファイル保存ダイアログの表示

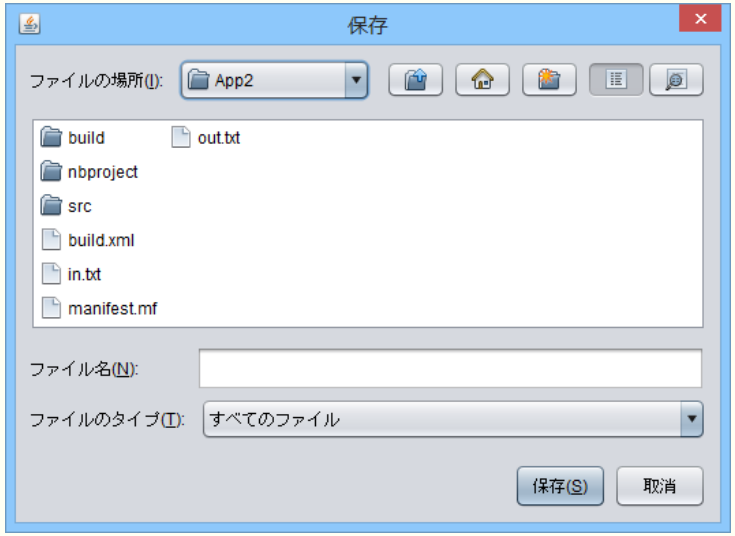
選択されたファイルのパスを取得

ダイアログを用いた読み書き

ファイル選択ダイアログ



ファイル保存ダイアログ



4.2 ファイルのフィルタリング

ファイルのフィルタリング

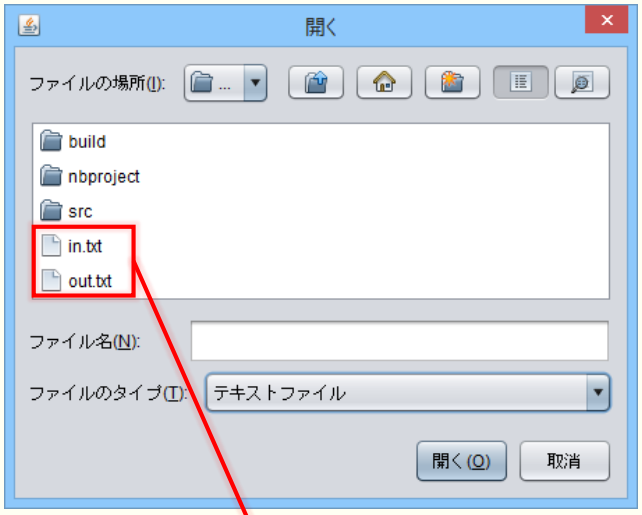
「ダイアログで読み込み」ボタンをダブルクリックして
以下のように編集してみましょう

```
private void btnReadDialogActionPerformed(java.awt.event.ActionEvent evt) {
        // ファイル選択ダイアログの生成
        JFileChooser fc = new JFileChooser(System.getProperty("user.dir"));
        FileNameExtensionFilter ff = new FileNameExtensionFilter("テキストファイル", "txt");
        fc.setFileFilter(ff);
        if (fc.showOpenDialog(this) != JFileChooser.APPROVE_OPTION) return;
        try {
                // ファイルオープン
                BufferedReader br = new BufferedReader(new FileReader(fc.getSelectedFile()));
                // ファイルの読み込み
                taOutput.setText("");
                for (String line; (line = br.readLine()) != null;) {
                        taOutput.append(line + "¥n");
                }
                // ファイルクローズ
                br.close();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

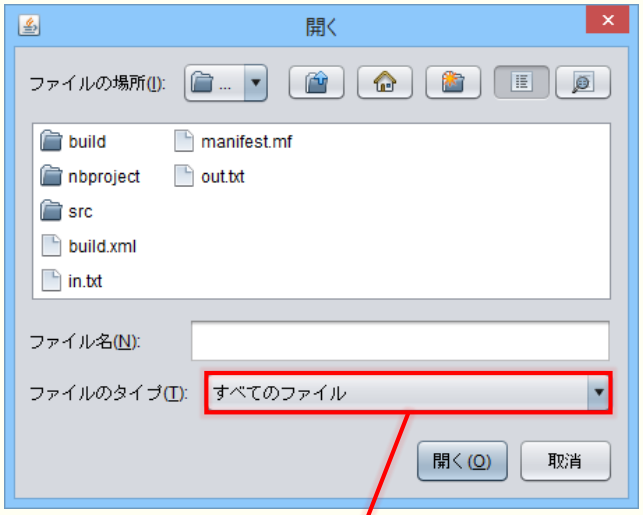
読み込むファイルのフィルタリング
テキストファイルのみ表示させる処理
（書き出し時も同様）

ファイルのフィルタリング

実行結果



テキストファイルのみ表示されている



ファイルのタイプを変更すれば
全てのファイルが表示される

だんだんそれっぽいソフトウェアになってきた

5 画像ファイルの入出力

新規プロジェクト
「ImageIO」を作ろう

画像の入出力

次のようにGUI部品を配置しましょう

GUI：JLabel
変数名：lblDraw
サイズ：512x512

GUI：JButton
変数名：btnRead

GUI：JButton
変数名：btnWrite

好きな画像を用意しよう

既存プロジェクト
「ImageIO」を使おう

画像ファイルの読み込み

「画像の読み込み」ボタンをダブルクリックして
以下のように処理を記述しましょう

```
BufferedImage bufImg;          // イメージパネル

private void btnReadImgActionPerformed(java.awt.event.ActionEvent evt) {
        try {
                // 画像ファイルの読み込み
                bufImg = ImageIO.read(new File("ct.bmp"));
                // 画像の表示
                lblDraw.setIcon(new ImageIcon(bufImg));
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

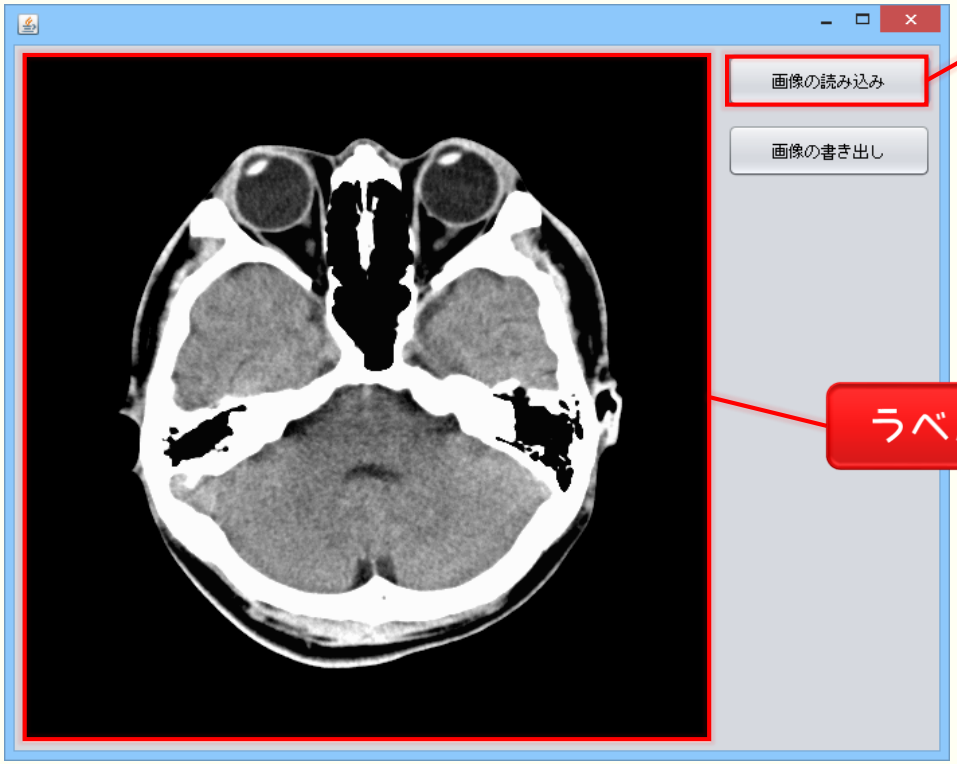
画像情報を記憶するクラス

画像ファイルのオープン

ラベルに画像を貼り付ける

画像ファイルの読み込み

プロジェクトフォルダ内部に
「ct.bmp」というファイルを置いて実行しましょう



著作権フリーの頭部断層画像

既存プロジェクト
「ImageIO」を使おう

画像ファイルの書き出し

「画像の書き出し」ボタンをダブルクリックして
以下のように処理を記述しましょう

```
private void btnWriteImgActionPerformed(java.awt.event.ActionEvent evt) {
        // 画像パネルが空の場合、終了
        if (bufImg == null)  return;
        try {
                // 画像ファイルの書き出し
                ImageIO.write(bufImg, "bmp", new File("out.bmp"));
                JOptionPane.showMessageDialog(this, "保存しました");
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

画像の有無を確認

画像の形式を指定した出力

bmp/jpg/png/gifなどの画像形式に対応しているよ

5.1 ドラッグ&ドロップによる入力

既存プロジェクト
「ImageIO」を使おう

画像ファイルの書き出し

次のように書いてみましょう

```
public class FrameMain extends javax.swing.JFrame implements DropTargetListener {

    public FrameMain() {
        initComponents();
        // ドラッグ & ドロップ領域の設定
        new DropTarget(lblDraw, this);
    }
```

インターフェースの実装

ドロップ領域の設定(lblDraw)

エラーが表示されるので [Alt]+[Enter]キーを押して
「すべての抽象メソッドを実装」を選択します

既存プロジェクト
「ImageIO」を使おう

画像ファイルの書き出し

```
@Override
public void dragEnter(DropTargetDragEvent dtde) {}

@Override
public void dragOver(DropTargetDragEvent dtde) {}

@Override
public void dropActionChanged(DropTargetDragEvent dtde) {}

@Override
public void dragExit(DropTargetEvent dte) {}

@Override
public void drop(DropTargetDropEvent dtde) {
    dtde.acceptDrop(DnDConstants.ACTION_COPY_OR_MOVE);
    Transferable trans = dtde.getTransferable();
    try {
        if (trans.isDataFlavorSupported(DataFlavor.javaFileListFlavor)) {
            File dragFile = ((List<File>) trans.getTransferData(DataFlavor.javaFileListFlavor)).get(0);
            bufImg = ImageIO.read(dragFile);
            lblDraw.setIcon(new ImageIcon(bufImg));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

5つのメソッドが自動追加される
（中身は消去する）

ドラッグされたファイルの情報が
「dragFile」に格納される（詳細はカット）

5.2 画像情報の表示

既存プロジェクト
「ImageIO」を使おう

画像情報の表示

次のようにGUI部品を配置しましょう

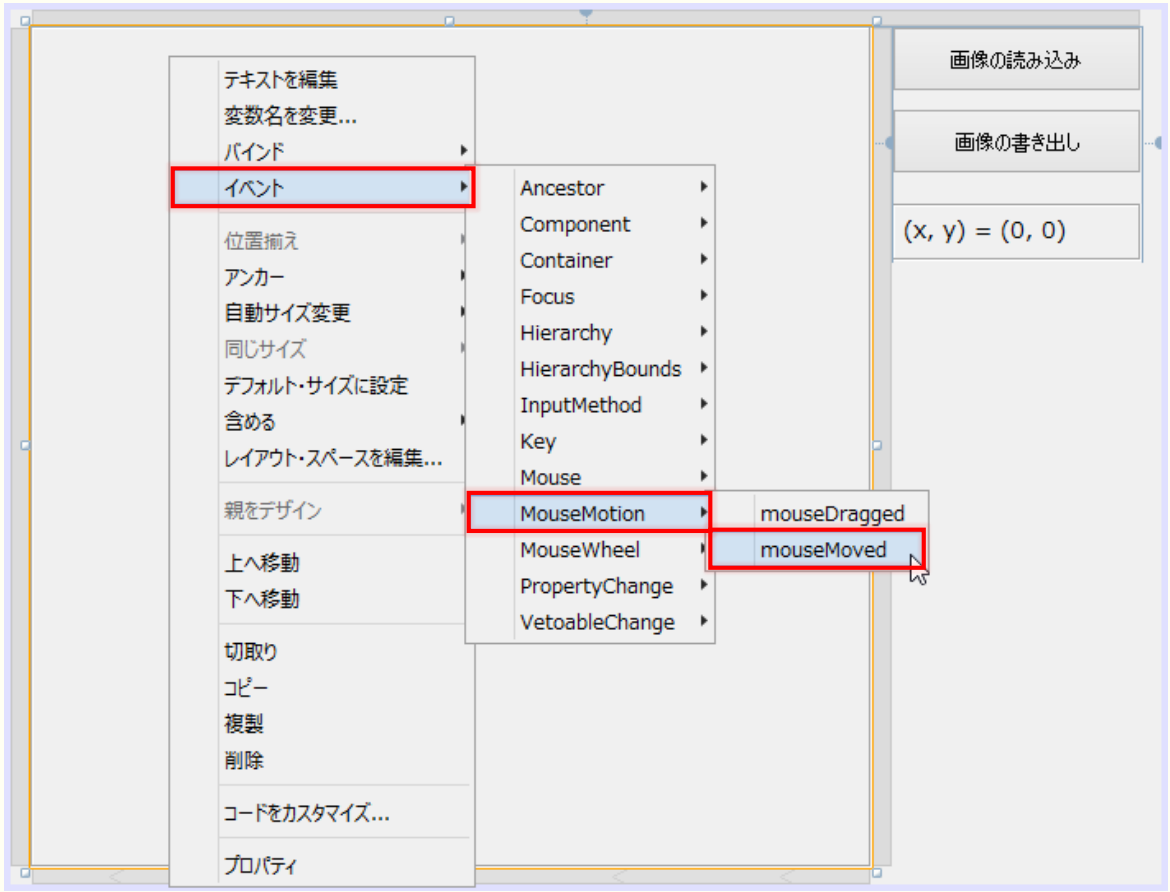
まずは、マウスの座標を取得してみよう

既存プロジェクト
「ImageIO」を使おう

画像情報の表示

ラベル上で右クリック

[イベント]→[MouseMotion]→[mouseMoved]と進む



既存プロジェクト
「ImageIO」を使おう

画像情報の表示

以下のように処理を記述しましょう

```
private void lblDrawMouseMoved(java.awt.event.MouseEvent evt) {
    // マウスの座標取得
    int mouseX = evt.getX();
    int mouseY = evt.getY();
    // マウス座標の表示
    lblMousePos.setText(" (x, y) = (" + mouseX + ", " + mouseY + ")");
}
```

ラベル上でマウスが動いた際に呼び出される

ラベルのテキストを更新してマウスの情報を表示する

ラベル上でマウスに動きがあると、
その度に何度も繰り返し呼び出されるメソッド

画像情報の表示

実行してみましょう

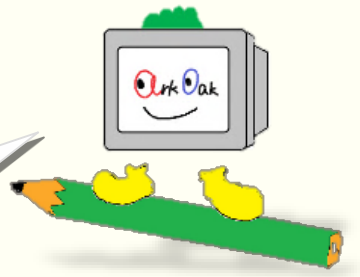
Javaでは左上が原点の座標系が基本

既存プロジェクト
「ImageIO」を使おう

RGB情報の表示

次のようにGUI部品を配置しましょう

配置ができたらソースコードに移動しよう



既存プロジェクト
「ImageIO」を使おう

RGB情報の表示

```
private void lblDrawMouseMoved(java.awt.event.MouseEvent evt) {
        // マウスの座標取得
        int mouseX = evt.getX();
        int mouseY = evt.getY();
        // マウス座標の表示
        lblMousePos.setText(" (x, y) = (" + mouseX + ", " + mouseY + ")");
        // 画像パネルが空の場合、終了
        if (bufImg == null)  return;
        // 色情報の取得
        Color c = new Color(bufImg.getRGB(mouseX, mouseY));
        // RGB情報の表示
        lblRed.setText("   R = " + c.getRed());
        lblGreen.setText("   G = " + c.getGreen());
        lblBlue.setText("   B = " + c.getBlue());
}
```

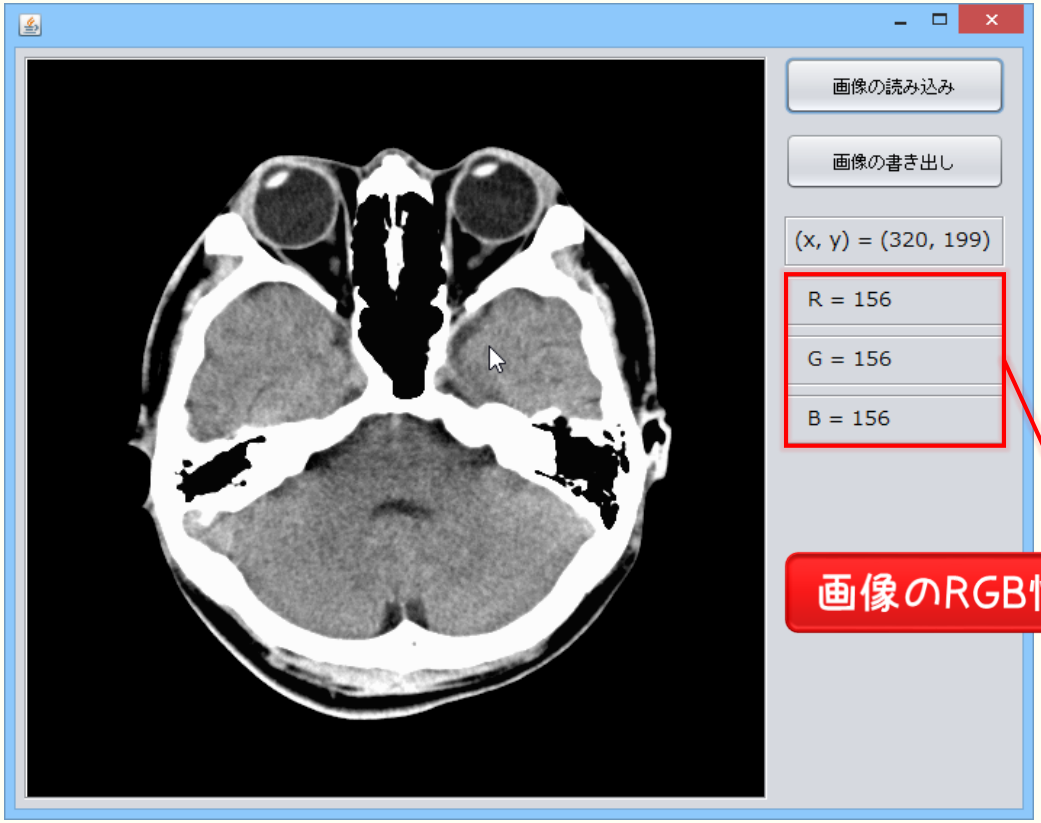
画像が読み込まれていない場合は
マウス座標だけ表示して終了

マウス座標に対応する
画像位置のRGB情報を取得

RGB情報を持ったColor変数から
各成分を抽出して表示

RGB情報の表示

実行してみましょう



グレースケール画像ならRGB値は同じになるね
色のついた画像も読み込んでみよう

次回予告

6. 描画処理と画像の編集

7. バイナリファイルの入出力

8. TIFFファイルの入出力

9. 画像処理アルゴリズム

10. 画像再構成アルゴリズム

お疲れ様でした

アンケートにご協力ください

https://docs.google.com/forms/d/1OK4Eb9wGrR9-4_ax1PCfJ84zx657XEC8UwrttXxSJkw/viewform

製作： 加納 徹