

Java講習会

文法編 第1回

ArkOak 加納 徹



Java講習会 文法編 第1回の流れ

1. Javaとは何か

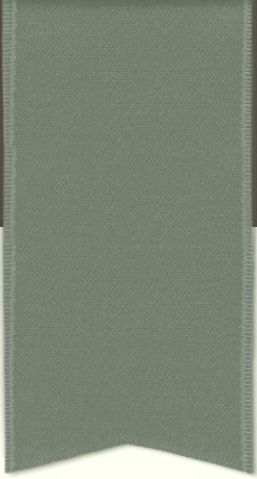
2. 開発環境の構築

3. Javaの決まり事

4. 文字の出力と変数

5. if文とswitch文

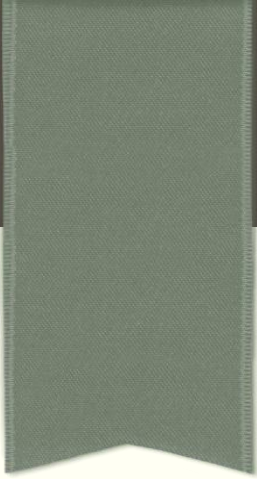
6. for文とwhile文



1. Javaとは何か

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上でも動く）
- オブジェクト指向



2. 開発環境の構築

開発環境の構築

■ 統合開発環境(Eclipse)のダウンロード

<http://mergedoc.sourceforge.jp/>

上記サイトから、最新版のEclipseをダウンロード&解凍。

JDKがインストール済みならば、Standard EditionでOK。

MergeDoc Project

1363 users

4,903,773 downloads

Pleiades プラグイン日本語化プラグイン

JStyle 改行タブ表示プラグイン

MergeDoc / Javadoc 日本語化

フォーラム

チケット

プロジェクト Wiki

ブログ

Pleiades - Eclipse プラグイン日本語化プラグイン

Pleiades All in One ダウンロード

Pleiades All in One は Windows 向けに開発対象プログラミング言語別 Pleiades および便利なプラグインのセットです。ダウンロードした zip ファイル eclipse.exe を起動すれば、すぐに日本語化された Eclipse を利用できます。Pleiades プラグイン本体をダウンロードしてください。

Eclipse 4.3 Kepler
Pleiades All in One

Eclipse 4.2 Juno
Pleiades All in One

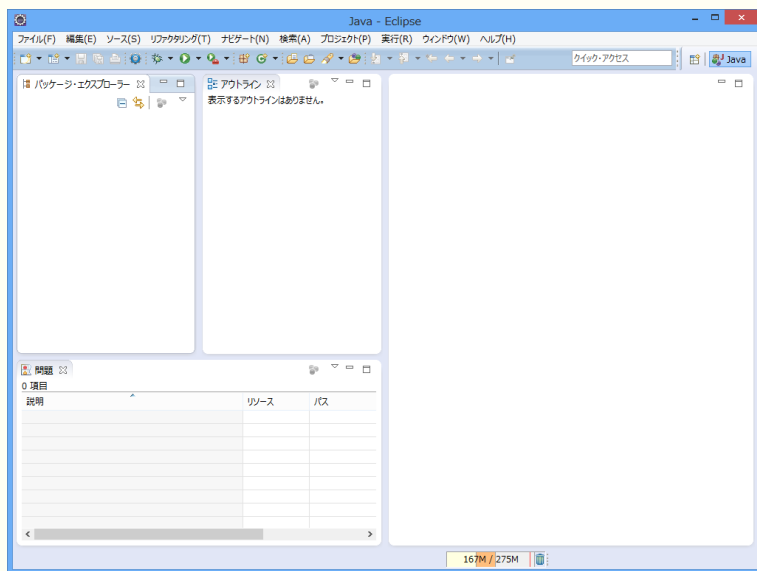
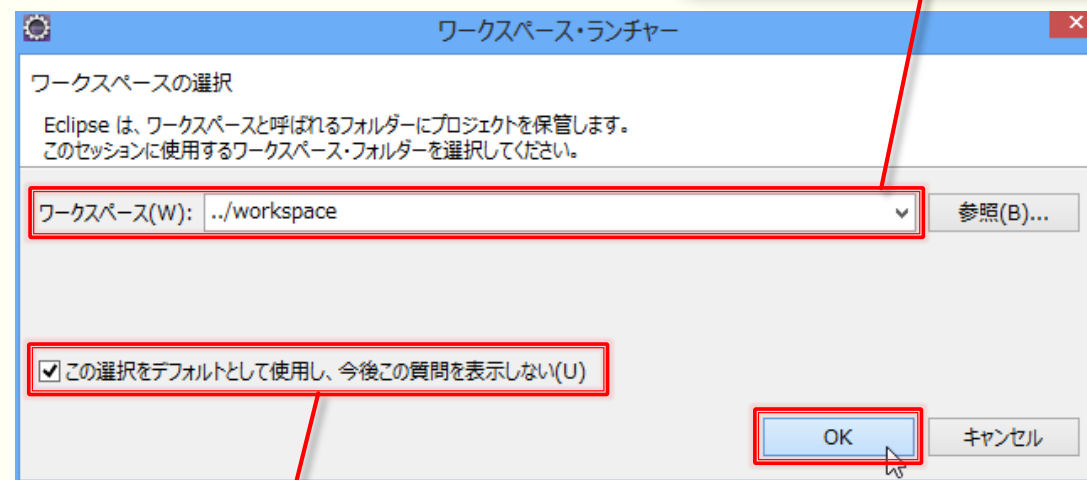
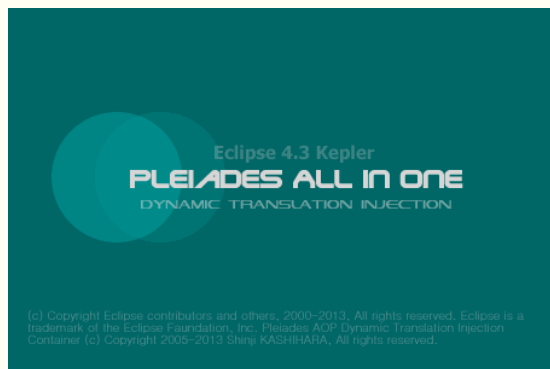
Eclipse 3.6 Helios
Pleiades All in One

Eclipse 3.5 Galileo
Pleiades All in One

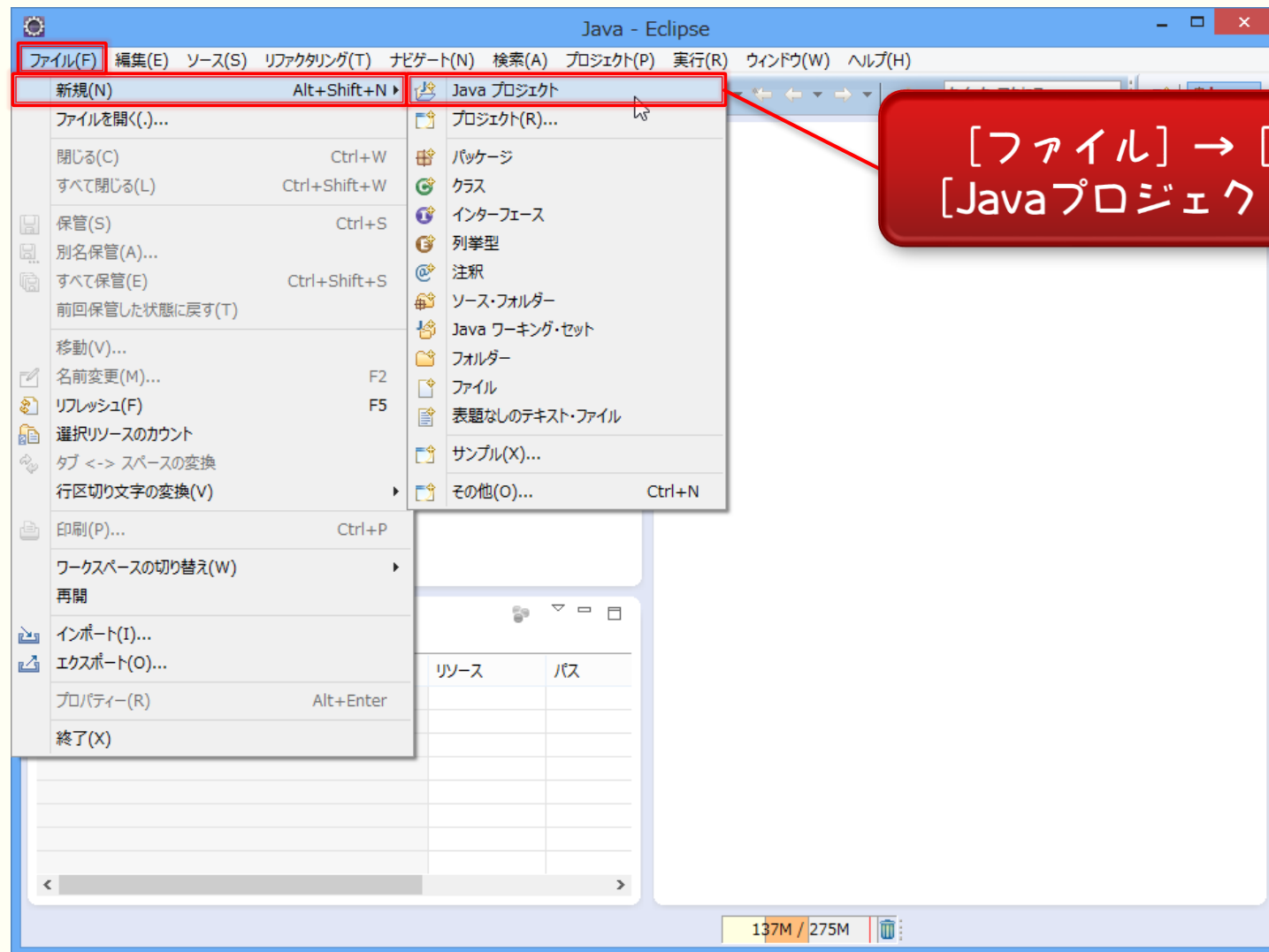
[illegible]

Eclipseの起動

- Eclipseフォルダ内の「Eclipse.exe」を起動。



新規プロジェクトの作成



新規プロジェクトの作成

The screenshot shows the 'New Java Project' dialog box. The 'Project name (P):' field is highlighted with a red box and contains the text 'Lecture01'. Below it, the 'Default location (L):' field shows 'D:\%pleiades%\workspace\Lecture01'. The 'JRE' section has '実行環境 JRE の使用 (V):' selected with 'JavaSE-1.7' in the dropdown. The 'Project layout' section has 'ソースおよびクラス・ファイルのフォルダーを個別に作成 (C)' selected. The 'Working set' section has 'ワーキング・セットにプロジェクトを追加 (T)' unchecked. At the bottom, the '完了 (F)' button is highlighted with a red box.

新規 Java プロジェクト

Java プロジェクトの作成
Java プロジェクトをワークスペースまたは外部ロケーションに作成します。

プロジェクト名(P): **Lecture01**

☒ デフォルト・ロケーションを使用(D)
ロケーション(L): D:\%pleiades%\workspace\Lecture01 参照(R)...

JRE
☒ 実行環境 JRE の使用(V): JavaSE-1.7
☐ プロジェクト固有の JRE を使用(S): jre7
☐ デフォルト JRE の使用(A) (現在は 'jre7') [JRE を構成...](#)

プロジェクト・レイアウト
☐ プロジェクト・フォルダーをソースおよびクラス・ファイルのルートとして使用(U)
☒ ソースおよびクラス・ファイルのフォルダーを個別に作成(C) [デフォルトを構成...](#)

ワーキング・セット
☐ ワーキング・セットにプロジェクトを追加(T)
ワーキング・セット(O): 選択(E)...

現在のワークスペースのデフォルト・コンパイラ準拠レベルは 1.6 です。新規プロジェクトはプロジェクトで指定するコンパイラ準拠レベル 1.7 が使用されます

< 戻る(B) 次へ(N) > **完了(F)** キャンセル

プロジェクト名をつける
(今回はLecture01)

The screenshot shows a dialog box asking if the user wants to open the associated perspective. The '常にご利用 (R)' checkbox is checked and highlighted with a red box. The 'はい (Y)' button is also highlighted with a red box.

関連付けられたパースペクティブを開きますか?

この種のプロジェクトは Java パースペクティブに関連付けられます。
このパースペクティブは、Java 開発をサポートするために設計されています。パッケージ・エクスプローラー、型階層、および Java 固有のナビゲーション・アクションを提供します。
このパースペクティブを開きますか?

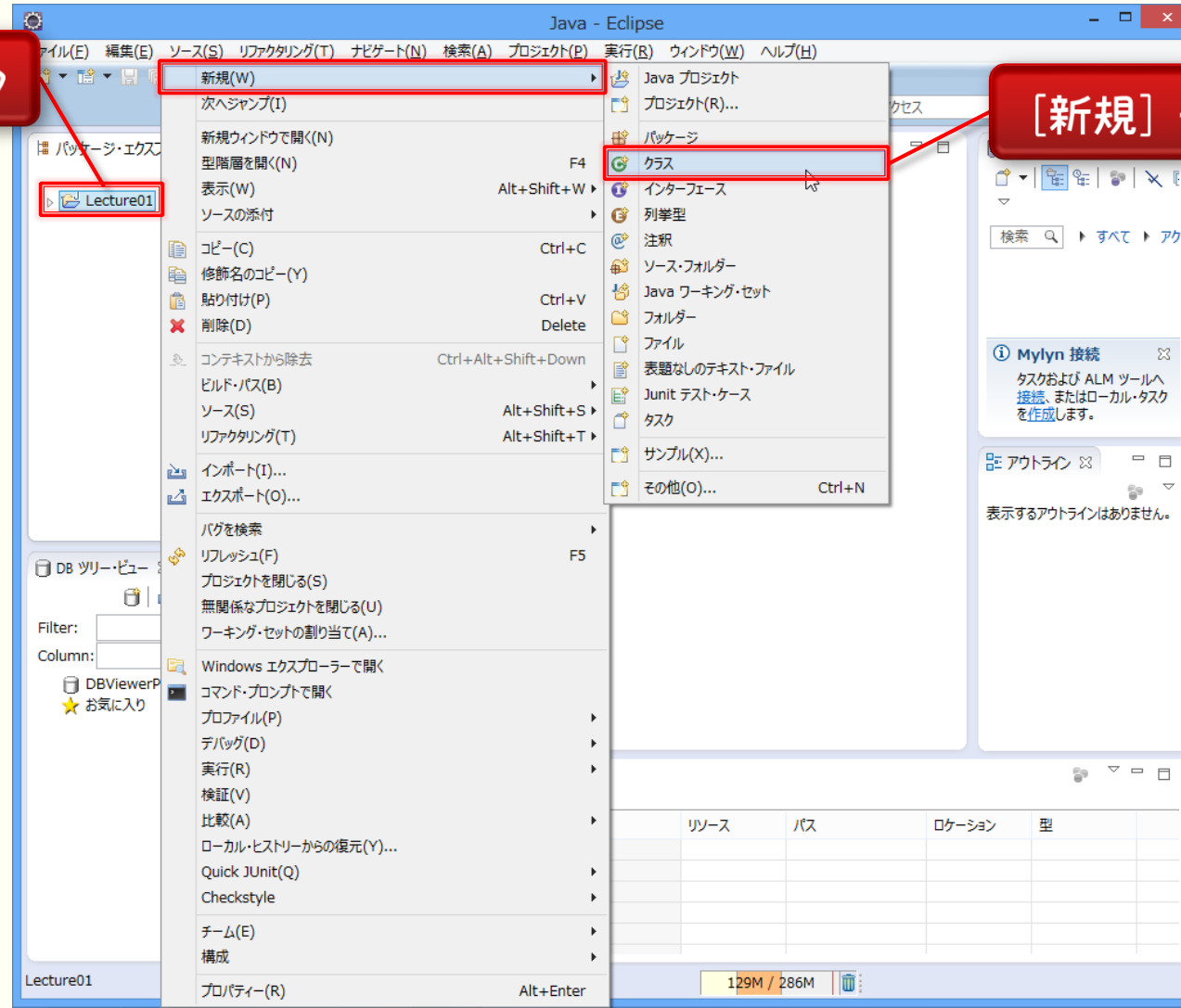
☒ 常にご利用 (R)

はい (Y) いいえ (N)

チェック

新規Javaクラスの作成

[Lecture01]を右クリック



[新規] → [クラス] と進む

新規Javaクラスの作成

Javaクラスの名前をつける
(今回はPractice01)

新規 Java クラス

Java クラス

デフォルト・パッケージの使用は推奨されません。

ソース・フォルダー(D): Lecture01/src 参照(O)...

パッケージ(K): (デフォルト) 参照(W)...

☐ エンクロージング型(Y): 参照(W)...

名前(M): Practice01

修飾子: ☒ public(P) ☐ default(U) ☐ private(V) ☐ protected(T)
☐ abstract(T) ☐ final(L) ☐ static(C)

スーパークラス(S): java.lang.Object 参照(E)...

インターフェース(I): 追加(A)...

どのメソッド・スタブを作成しますか?

☒ public static void main(String[] args)(V)

☐ スーパークラスからのコンストラクター(C)

☒ 継承された抽象メソッド(H)

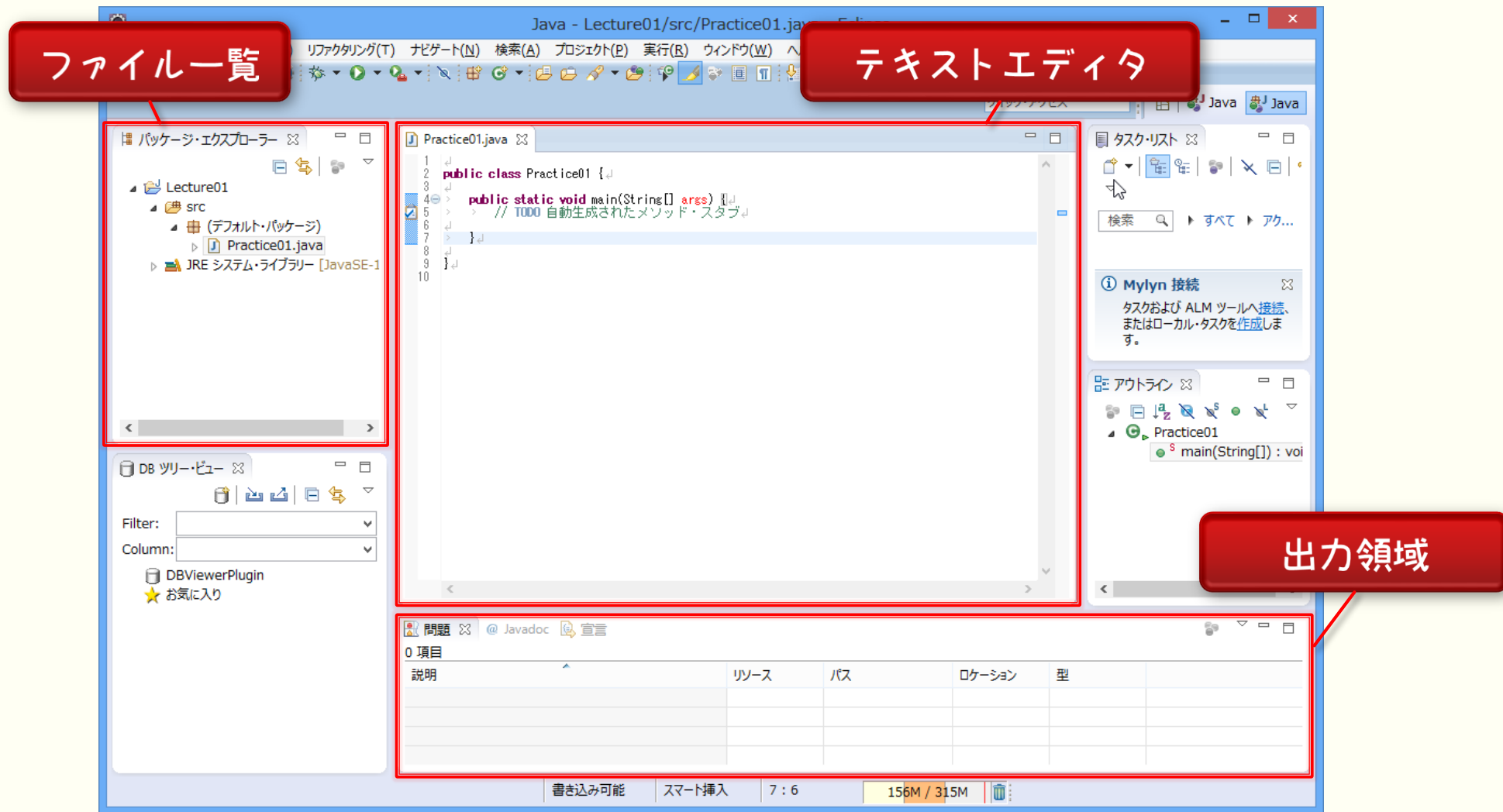
コメントを追加しますか? (テンプレートの構成およびデフォルト値については[ここ](#)を参照)

☐ コメントの生成(G)

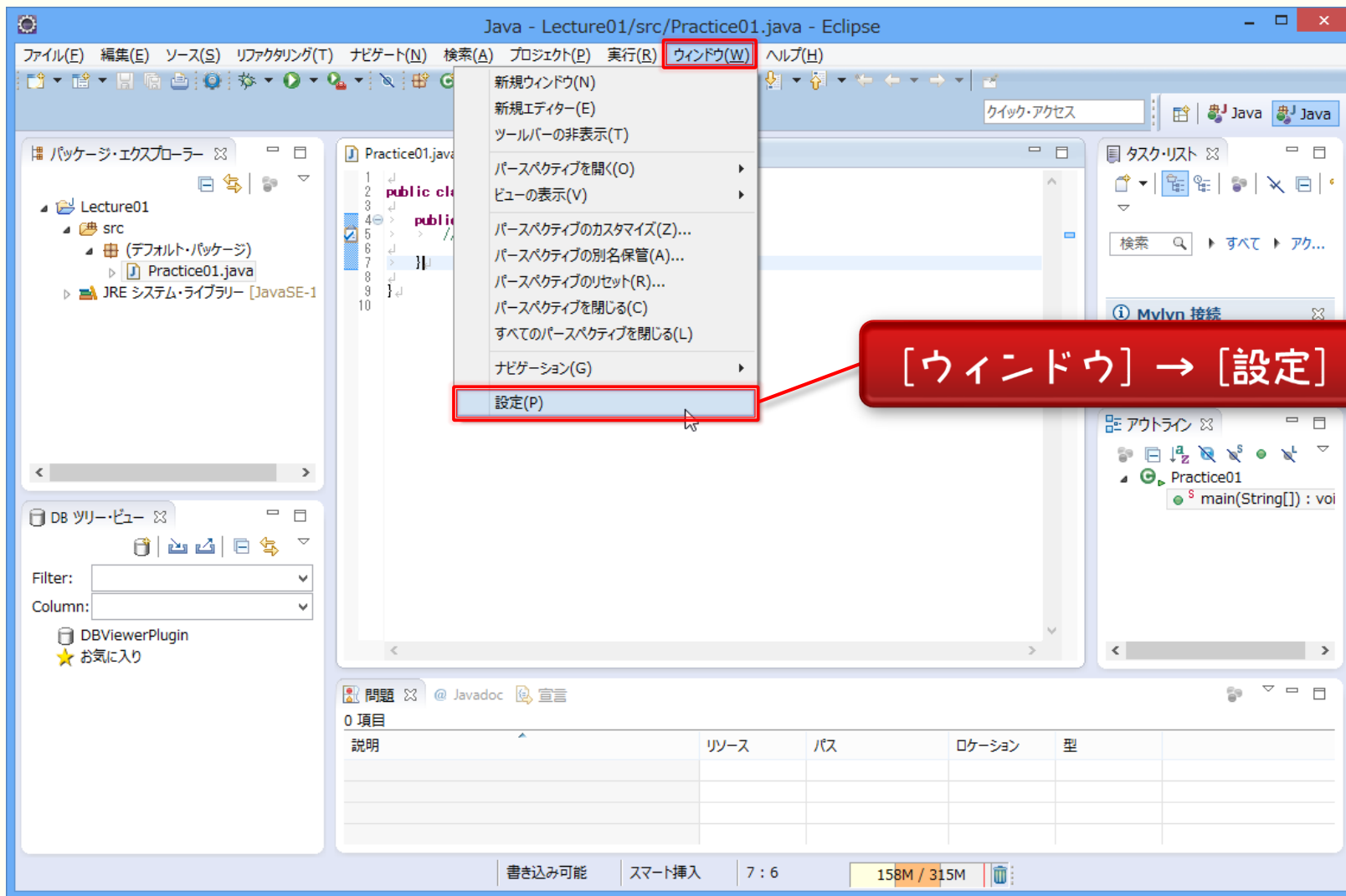
完了(F) キャンセル

チェック

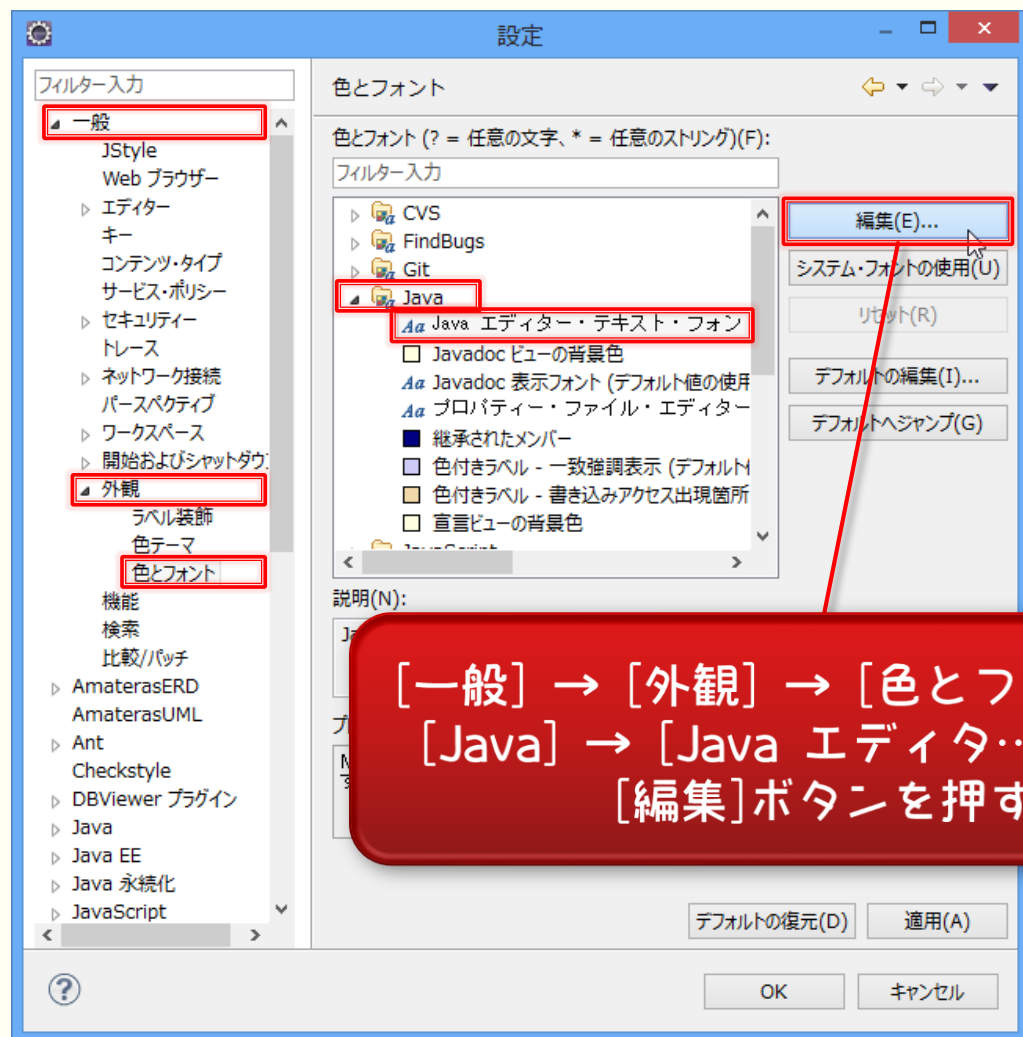
新規Javaクラスの作成



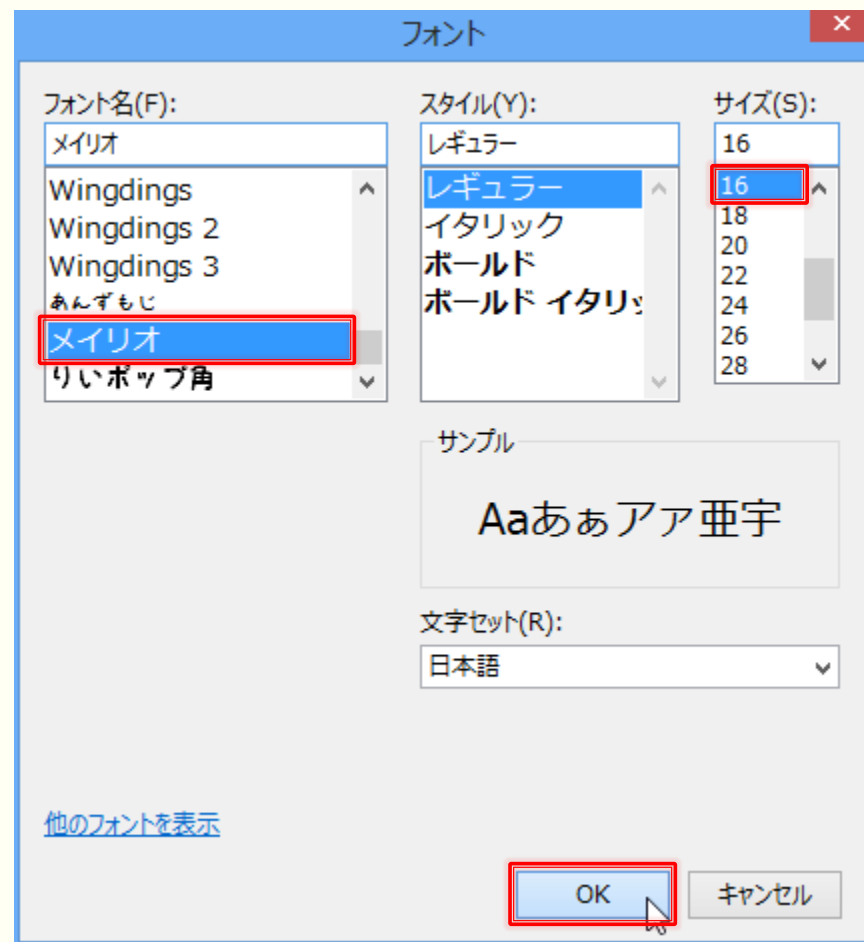
フォントサイズの変更



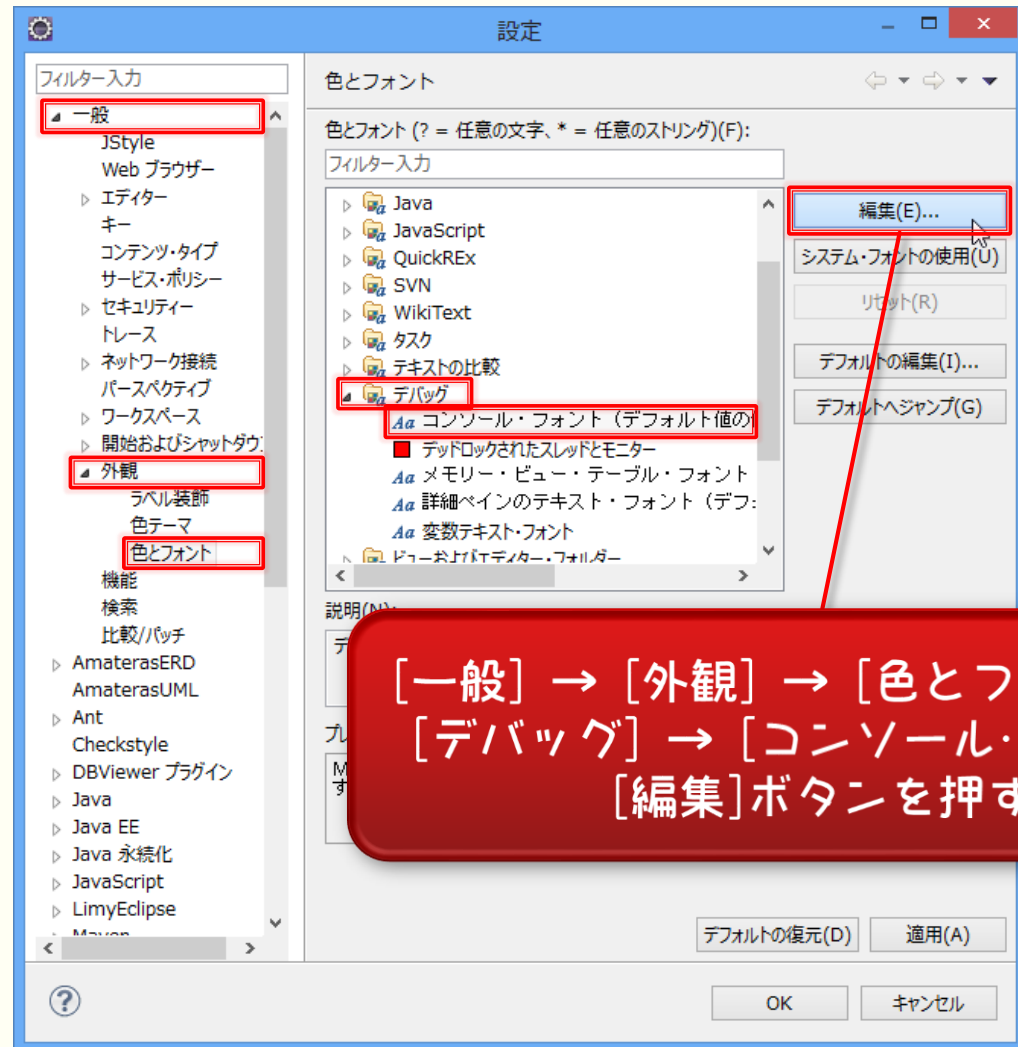
フォントサイズの変更



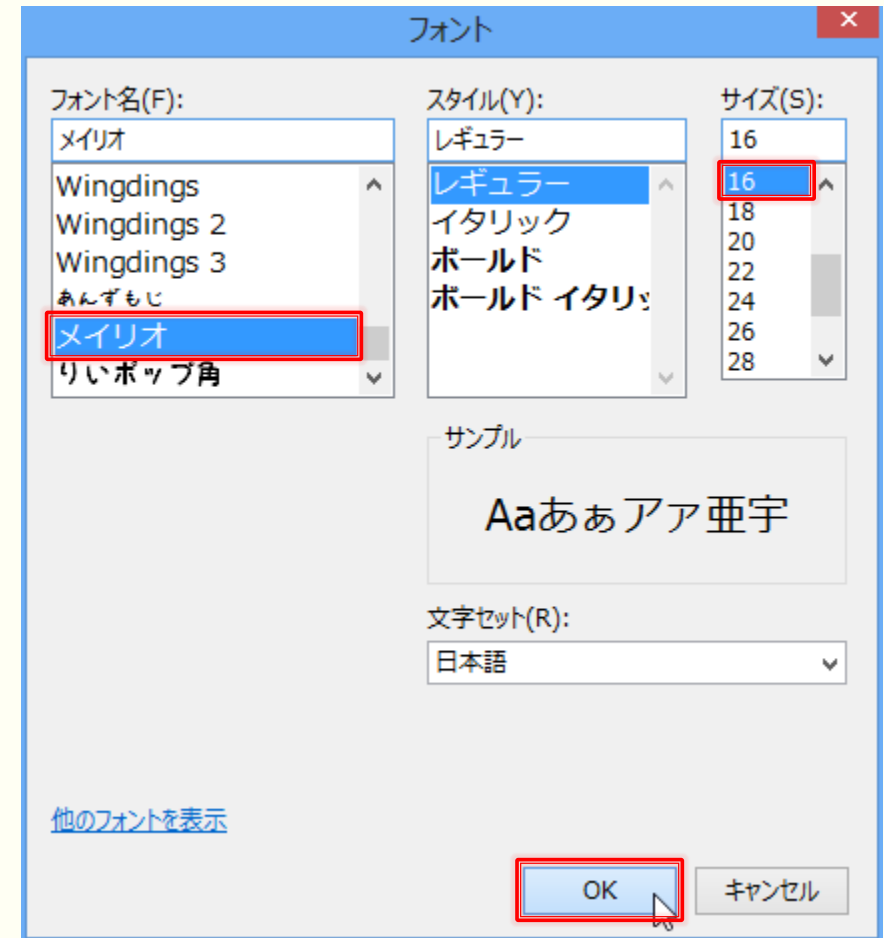
[一般] → [外観] → [色とフォント] →
[Java] → [Java エディタ...] と進み
[編集]ボタンを押す



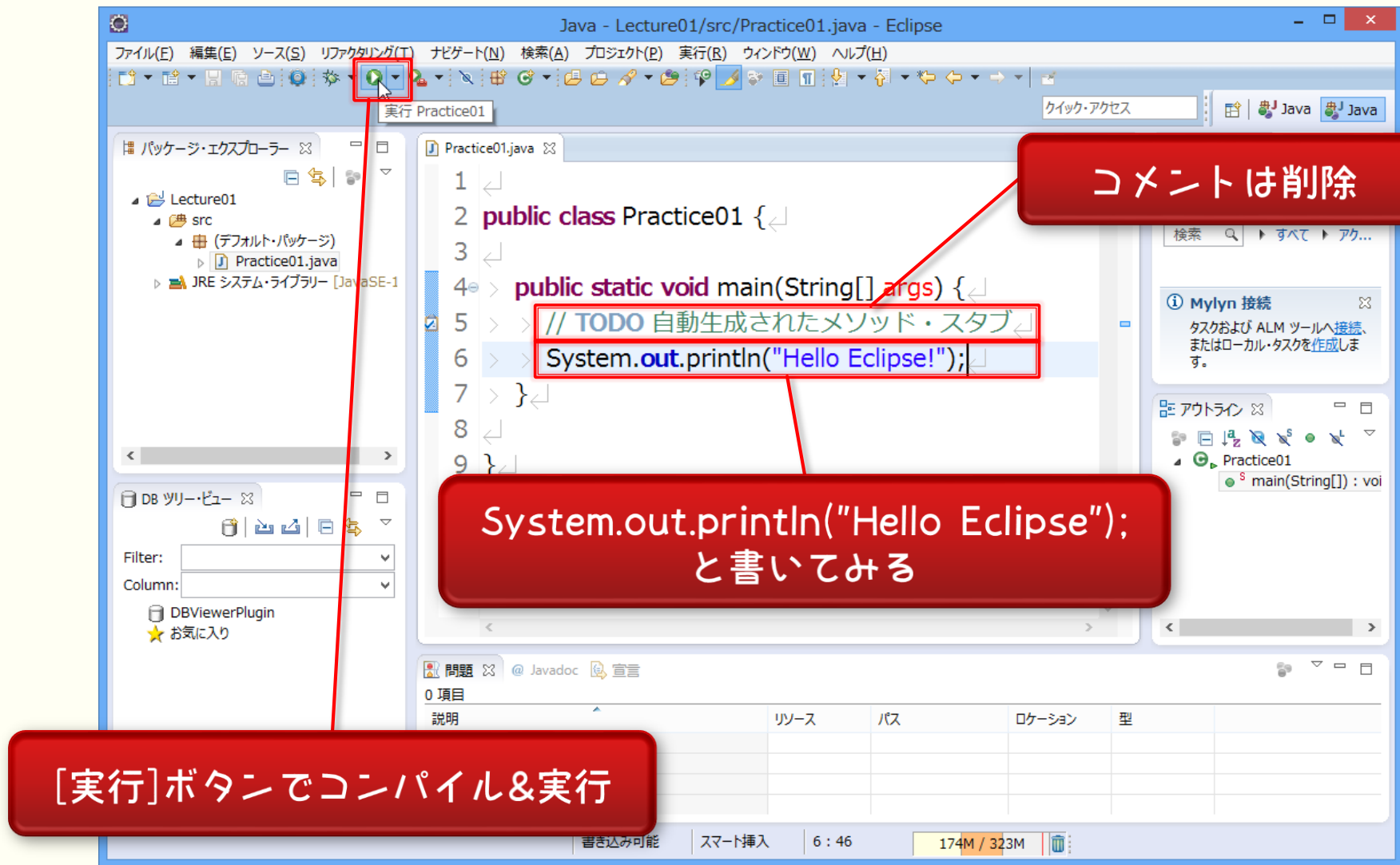
フォントサイズの変更（出力領域）



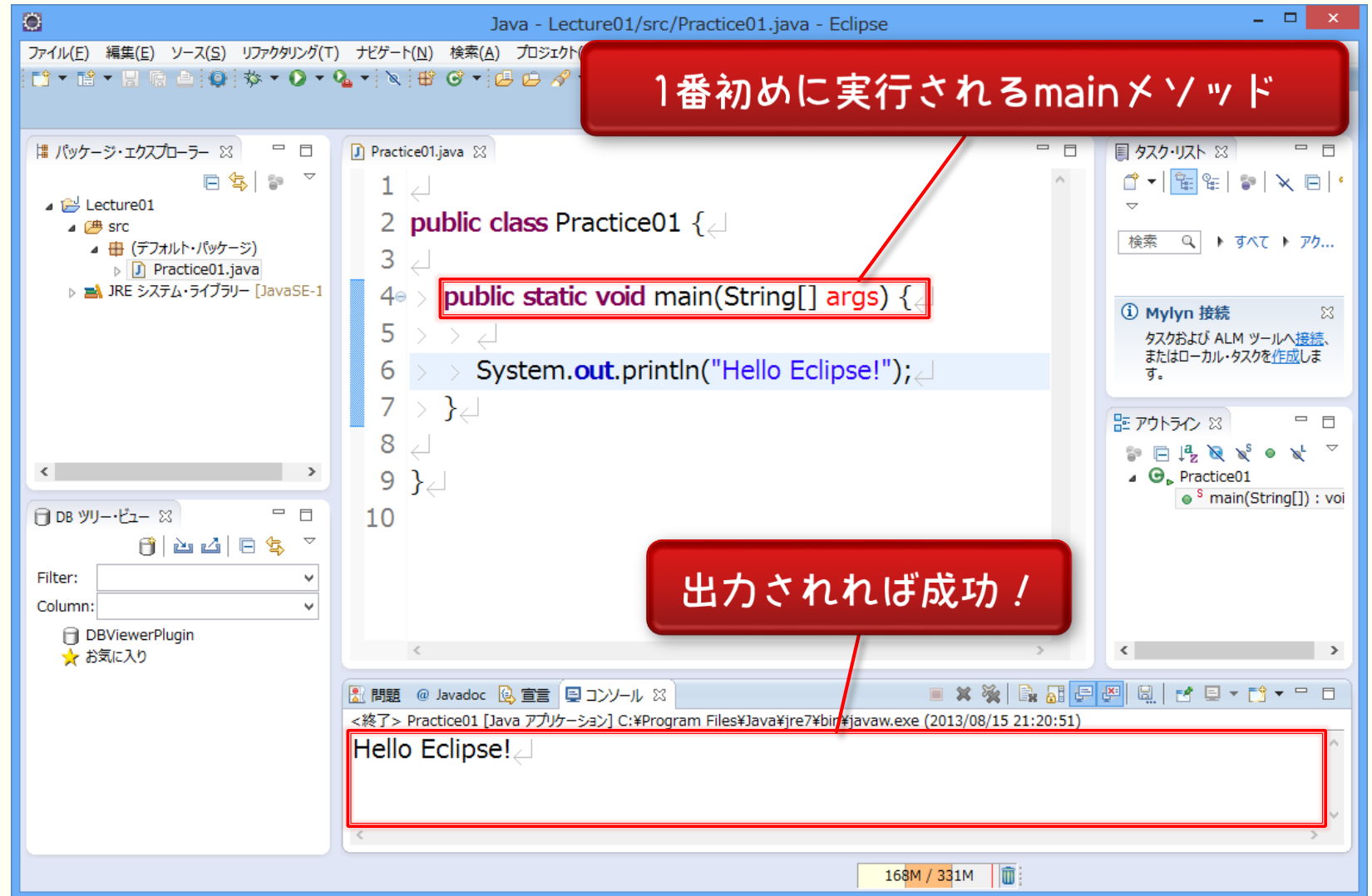
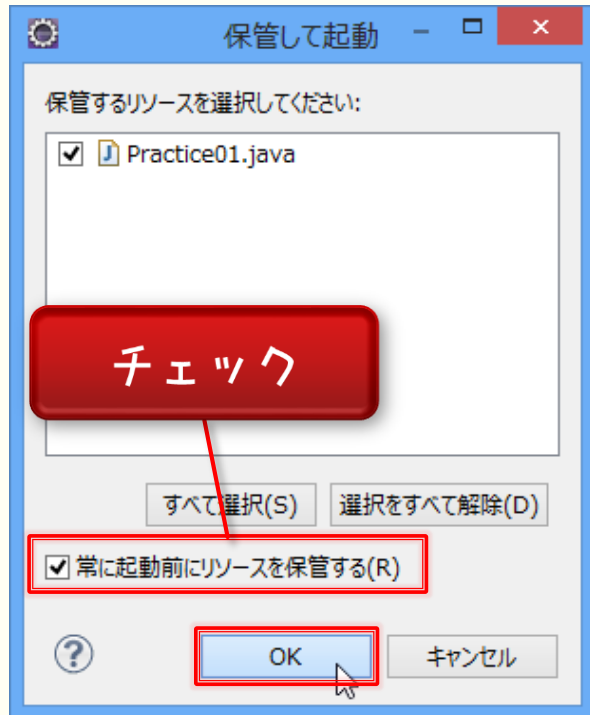
[一般] → [外観] → [色とフォント] →
[デバッグ] → [コンソール...] と進み
[編集]ボタンを押す

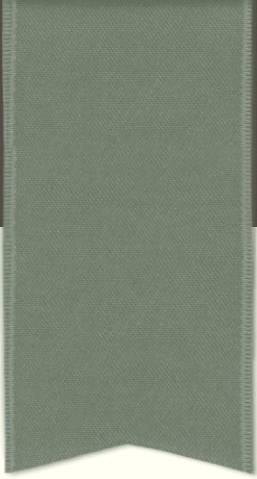


Hello Eclipse !



Hello Eclipse !





3. Javaの決まり事

決まり事 その1

全ての変数・メソッド(関数)は
クラスに属さなければならない。

クラスとは

メソッド(関数)もひとまとめにした構造体のようなもの。

構造体

変数 a
変数 b
配列 c[]

クラス

変数 a
変数 b
配列 c[]
メソッド d

最も単純なJavaコード

Practice01クラス

Practice01.java

```
public class Practice01 {
```

```
    public static void main(String[] args) {
```

```
        System.out.println("Hello Eclipse!");
```

```
    }
```

```
}
```

mainメソッドもクラス内部に書く

オブジェクト指向プログラミング

全ての変数・メソッドをクラスで扱うことで、次の機能を実現。

カプセル化

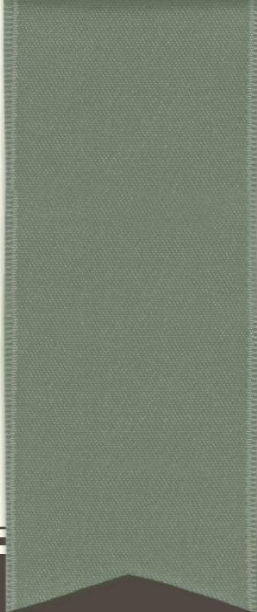
データや振る舞いを一体化して「オブジェクト」として定義する仕組み。
データの独立性が高まり、ソフトウェアの保守性・再利用性が向上する。

継承 (Extends)

既存クラスの機能・構造を受け継ぎ、新たなクラスを派生させる仕組み。(スーパークラスとサブクラス)
一連の機能を引き継ぎ、新たな機能を追加できるため、コードの再利用性が向上する。

多態性 (Polymorphism)

変数やメソッドが複数の型に属することを許す性質。
同じ名前のメソッドを呼び出す時、変数の型に応じて適切な処理を施すことができる。



決まり事 その2

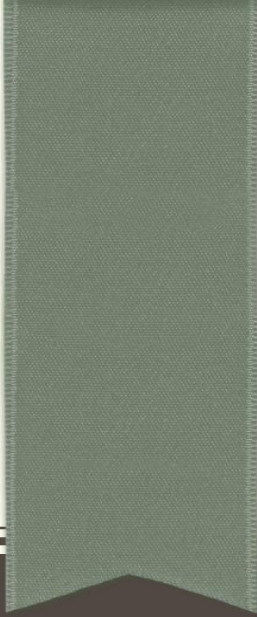
クラスファイル「〇〇.java」には
必ず〇〇クラスが1つ存在しなければならない。

最も単純なJavaコード

Practice01.java

同じ名前にする

```
public class Practice01 {  
    public static void main(String[] args) {  
        System.out.println("Hello Eclipse!");  
    }  
}
```



決まり事 その3

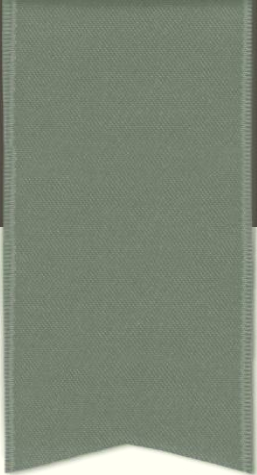
mainメソッドは必ず1つ
存在しなければならない。

最も単純なJavaコード

Practice01.java

```
public class Practice01 {  
    public static void main(String[] args) {  
        System.out.println("Hello Eclipse!");  
    }  
}
```

1番初めに実行されるmainメソッド
C言語のmain関数に相当



3. 文字の出力と変数

文字の出力

Practice01.java

文字の出力

文字の出力と改行

```
public class Practice01 {  
    public static void main(String[] args) {  
        System.out.print("Hello ");  
        System.out.println("Eclipse!");  
    }  
}
```

Javaの変数

基本的にはC言語と同じ。新たにString型、boolean型が仲間に。

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	少数以下約7桁までの値
double	浮動小数点数	8 バイト	少数以下約14桁までの値
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

String型

Practice01.java

```
public class Practice01 {  
    public static void main(String[] args) {  
        String s1 = "Hello ";  
        String s2 = "Eclipse ";  
        System.out.println(s1 + s2 + "Kepler!");  
    }  
}
```

String型の変数

“+” で文字を結合できる

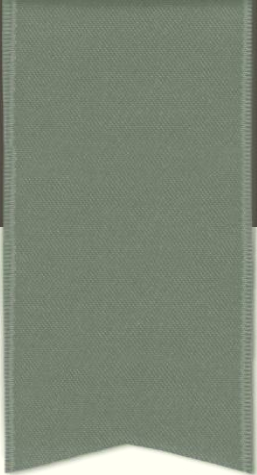
Javaの算術演算

Practice02.java

```
public class Practice02 {  
    public static void main(String[] args) {  
        int a = 64, b = 16;  
        System.out.println("a = " + a);  
        System.out.println("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));  
    }  
}
```

“+” で変数を結合できる

括弧で囲うと結合ではなく
和を計算する



4. if文とswitch文

if文

Practice03.java

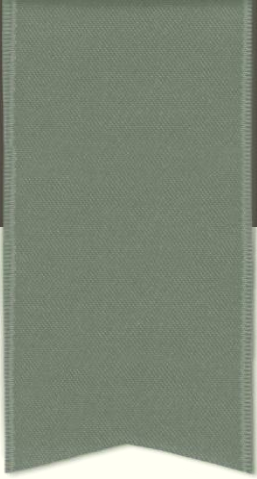
```
public class Practice03 {  
    public static void main(String[] args) {  
        int a = 64, b = 16;  
  
        if (a > b) {  
            System.out.println("aはbよりも大きい");  
        } else if (a < b) {  
            System.out.println("aはbよりも小さい");  
        } else {  
            System.out.println("aとbは等しい");  
        }  
    }  
}
```

switch文

Practice04.java

[0, 1) の実数値を
ランダムで返す関数

```
public class Practice04 {  
    public static void main(String[] args) {  
        int n = (int) (Math.random() * 3);  
        switch (n) {  
            case 0:  
                System.out.println("大吉");  
                break;  
            case 1:  
                System.out.println("末吉");  
                break;  
            case 2:  
                System.out.println("大凶");  
                break;  
        }  
    }  
}
```



5. for文とwhile文

for文

Practice05.java

```
public class Practice05 {  
    public static void main(String[] args) {  
        int sum = 0;  
        for (int i = 1; i <= 100; i++){  
            sum += i;  
        }  
        System.out.println("1から100までの和は、" + sum);  
    }  
}
```

for文の中でだけ有効な変数 i

while文

Practice06.java

```
public class Practice06 {  
    public static void main(String[] args) {  
        double distance = 380000e3; // 地球と月の距離[m]  
        double paper = 100e-6; // 紙の厚み[m]  
        int count = 0; // 紙を折った回数  
  
        // 月に到達するまで、紙を折っていく  
        while (paper < distance) {  
            paper = paper * 2;  
            count++;  
        }  
  
        System.out.println(count + "回紙を折ると月に届きます");  
    }  
}
```

10の3乗を表す

10のマイナス6乗を表す

練習問題

問 1

コインを1,000回投げて、何回表が出たかを調べなさい

問 2

問 1 の試行を100,000回行い、表が出た回数の平均値を求めなさい

問 3

コインを1,000回投げたとき、表が550回以上出る確率を考えなさい

問 1 解答例

Practice07.java

1/2の確率で、if文の条件を満たす

```
public class Practice07 {  
    public static void main(String[] args) {  
        int omote = 0;  
        for (int i = 0; i < 1000; i++) {  
            if (Math.random() < 0.5) {  
                omote++;  
            }  
        }  
        System.out.println("表が出た回数は、" + omote + "回");  
    }  
}
```

問2 解答例

Practice07.java

```
public class Practice03 {  
    public static void main(String[] args) {  
        int omote = 0;  
        for (int j = 0; j < 100000; j++) {  
            for (int i = 0; i < 1000; i++) {  
                if (Math.random() < 0.5) {  
                    omote++;  
                }  
            }  
        }  
        double average = omote / 100000.0;  
        System.out.println("表が出た回数の平均値は、" + average + "回");  
    }  
}
```

100,000回繰り返す

試行回数で除して平均を出す

問3 解答例

Practice07.java

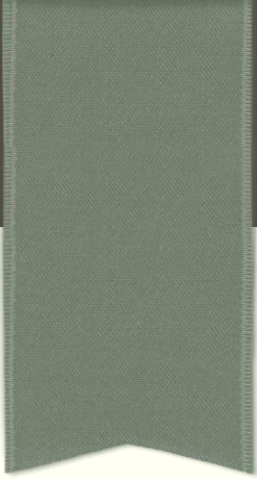
```
public class Practice07 {  
    public static void main(String[] args) {  
        int omote = 0, count = 0;  
        for (int j = 0; j < 100000; j++) {  
            omote = 0;  
            for (int i = 0; i < 1000; i++) {  
                if (Math.random() < 0.5) {  
                    omote++;  
                }  
            }  
            if (omote >= 550) count++;  
        }  
        double rate = 100.0 * count / 100000.0;  
        System.out.println("1000回中550回以上表が出る確率は、" + rate + "%");  
    }  
}
```

試行毎に表の回数をリセット

550回以上表が出た回数をカウント

試行回数で除して比率を出し、
100分率に換算している

厳密解：0.0863%



おわり