

Java基礎文法

2017/10/2(月) プログラミングII 第一～二回
福井大学 工学研究科 情報・メディア工学専攻
長谷川達人

本日の目次

- **本講義の概要**
- Javaについて
- 使用する開発環境
- Java基礎文法

確認

講義名称 プログラミングII

担当教員 石井先生, 長谷川

確認事項 1

今年新設科目なので, 過年度生は振替科目を確認すること.
旧) プロII → 現) プロI

確認事項 2

計算機室の定員の都合で105名を超える場合は抽選を行う.
必修である情報システムコースの学生を優先する.
本日受講が確定した人たちは履修登録を実施すること.

本講義の概要

| 前半：Java 担当：長谷川 | | 後半：Scala 担当：石井先生 | |
|----------------|-------------------|------------------|------------------|
| 第1回 | Java基礎文法/開発環境の使い方 | 第9回 | Scala言語の基本 |
| 第2回 | Java基礎文法/C言語との差異 | 第10回 | 関数型プログラミングの基本 |
| 第3回 | オブジェクト指向 | 第11回 | 関数とクロージャ |
| 第4回 | クラス/インスタンス/メソッド1 | 第12回 | 関数型プログラミングの簡単な例 |
| 第5回 | クラス/インスタンス/メソッド2 | 第13回 | 静的型付けと動的型付け |
| 第6回 | 継承/カプセル化1 | 第14回 | ケースクラス/パターンマッチング |
| 第7回 | 継承/カプセル化2 | 第15回 | 多相性やパターンマッチングの例 |
| 第8回 | 中間試験/Java言語のまとめ | 第16回 | 期末試験 |

※来年以降は全てJavaになる予定

本講義の概要

成績評価と参考図書

成績評価：全体

中間試験と期末試験の結果を総合し成績評価する。
60%以上で合格とする。

参考図書

このような注釈を講義中で表示するが、注釈は公開PDFに
含まれないものもある。必要に応じてメモするとよい。

中山 清喬 他 (2014)
『スッキリわかるJava入門 第2版』 インプレス
※必携ではない（JavaではWeb上でPDF資料を公開する）

本講義の概要

その他



次回以降は3号館計算機室で授業を実施する。
後半（第9回～）は石井先生担当なので別途指示する。



毎週の課題提出にて出席とする。
後半（第9回～）は石井先生担当なので別途指示する。



毎週の資料は学内のみWebで公開する（必要な人は**事前印刷**）。
後半（第9回～）は石井先生担当なので別途指示する。

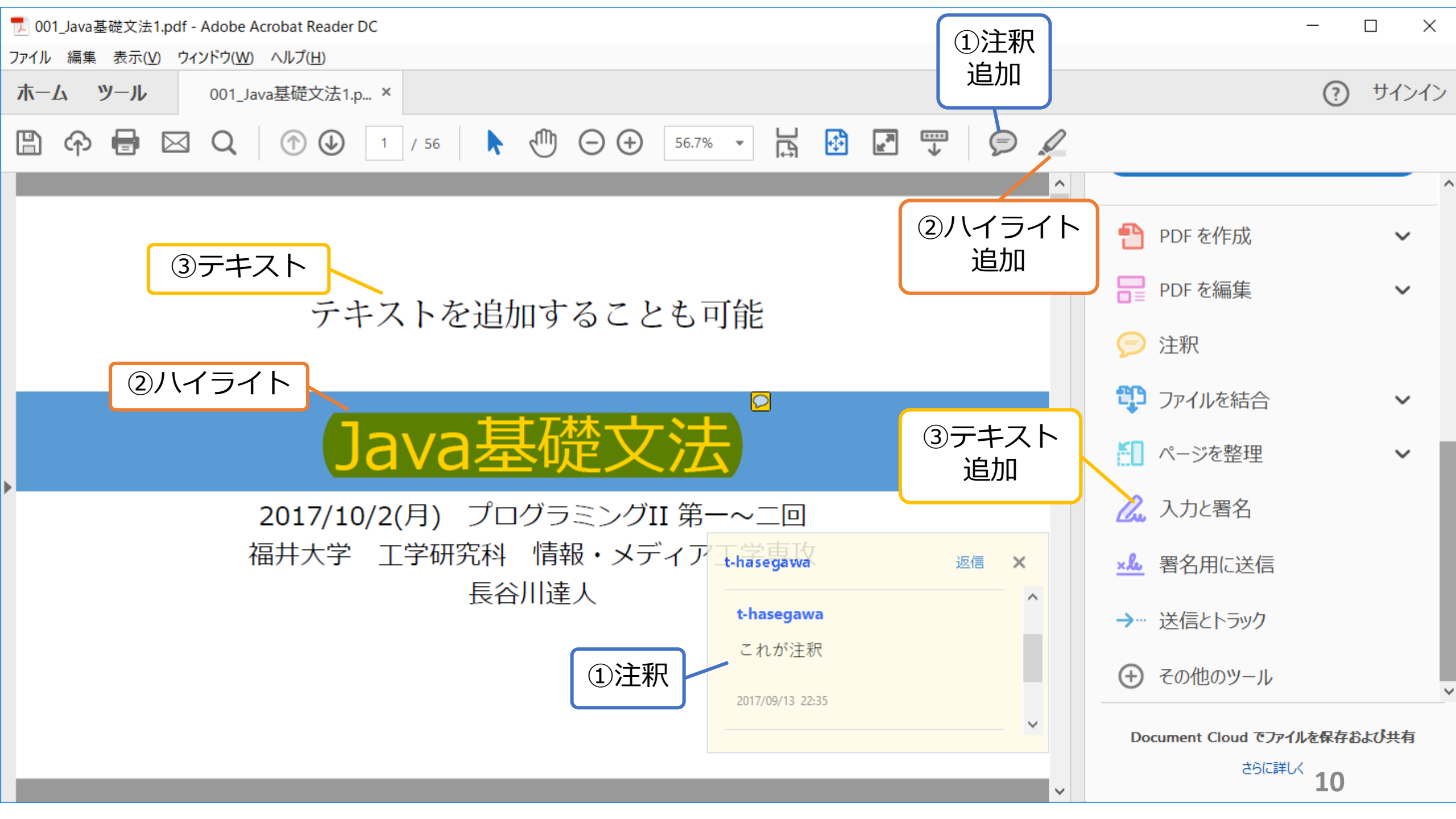
<http://hsgw-nas.fuis.u-fukui.ac.jp/lecture.html>

本講義の概要

メモについて

事前印刷が面倒な人は、Adobe Acrobat Reader DCの
①注釈、②ハイライト、③入力と署名
機能を使うと、そのままPDFに追記が可能となる。

紙で管理しなくてよくなるので便利である。



①注釈
追加

②ハイライト
追加

③テキスト

②ハイライト

③テキスト
追加

①注釈

テキストを追加することも可能

Java基礎文法

2017/10/2(月) プログラミングII 第一～二回
福井大学 工学研究科 情報・メディア工学専攻
長谷川達人

t-hasegawa

返信

t-hasegawa

これが注釈

2017/09/13 22:35

- PDF を作成
- PDF を編集
- 注釈
- ファイルを結合
- ページを整理
- 入力と署名
- 署名用に送信
- 送信とトラック
- その他のツール

Document Cloud でファイルを保存および共有

本講義の概要

理解には段階がある



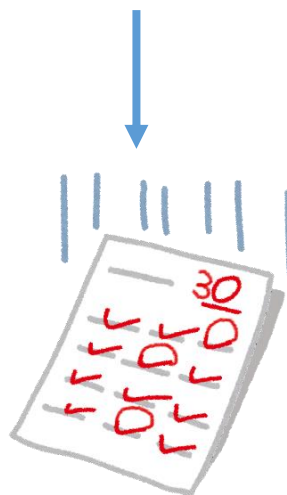
何もしていない



ここまでではできている人が多い



ちゃんと理解した



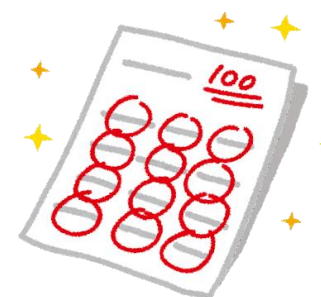
このギャップに戸惑うことが多い



ちゃんと理解した



何度も練習



本日の目標

概要

Javaの背景と基礎文法の一部を紹介する． C言語とほぼ同じ文法にのみ焦点を当てるため，今回は量が多い．

目標

Javaを学ぶ重要性を理解する．
C言語で習った文法をJavaでプログラミングできる．



なるほど

本日の提出課題

課題 1

本日の授業を聞いて、
初めて知ったと思う内容を2点簡潔に述べよ。
「簡潔に述べよ」≡「1, 2文程度で述べよ」と考えるとよい。

課題 2

本日の授業を聞いて、
質問事項または**気になった点**を2点簡潔に述べよ。
質問を考えるときは、様々な点に対して5W1Hを考えるとよい。

できればわからないこと等のコメントはその場でしてくれるとありがたい。

本日の目次

- 本講義の概要
- **Javaについて**
- 使用する開発環境
- Java基礎文法

Javaについて

- 1995年にSun Micro Systems（現Oracle）社が開発した言語

- 以下のような特徴がある.

特に**オブジェクト指向**を理解できるように
今後の講義を進めていく

- **オブジェクト指向**

- オブジェクト（操作対象）単位でプログラミングを行う技法

- メモリ管理の自動化（**ガベージコレクション**）

- 使わなくなった変数等のメモリを自動で解放してくれる仕組み

- **マルチスレッド**

- 複数処理を並列実装することが容易

- **Java仮想マシン**

- どこでもJava仮想マシンが動く環境ならばJavaが動く

Javaについて

雇用者側から見た需要ランキング

| Language Rank | Types | Jobs Ranking |
|---------------|--------|--------------|
| 1. Java | 🌐 📱 🖥️ | 100.0 |
| 2. C | 📱 🖥️ 🧠 | 99.4 |
| 3. Python | 🌐 🖥️ | 99.3 |
| 4. C++ | 📱 🖥️ 🧠 | 92.2 |
| 5. JavaScript | 🌐 📱 | 89.9 |
| 6. C# | 🌐 📱 🖥️ | 86.4 |
| 7. PHP | 🌐 | 80.5 |
| 8. HTML | 🌐 | 79.7 |
| 9. Ruby | 🌐 🖥️ | 76.6 |
| 10. Swift | 📱 🖥️ | 76.4 |



引用 : Interactive: The Top Programming Languages 2017
© Copyright 2017 IEEE Spectrum

Javaについて

Javaで開発できるもの



Webシステム

Twitter
Evernote



PCアプリ

OpenOffice
MINECRAFT



Androidアプリ

ほぼ全て

要するに、大体何でも開発できる。
即ち、需要も高い。

本日の目次

- 本講義の概要
- Javaについて
- **使用する開発環境**
- Java基礎文法

使用する開発環境

コマンドでもコンパイルできるが、**Eclipse**という**IDE**を使う。

IDE

統合開発環境のことで、GUIベースでエディタやコンパイラ、デバッガなどを簡単に使用できるようにしたものである。

Eclipse

オープンソースの統合開発環境（非常に有能）で、Java開発はEclipseを使うことが多い。

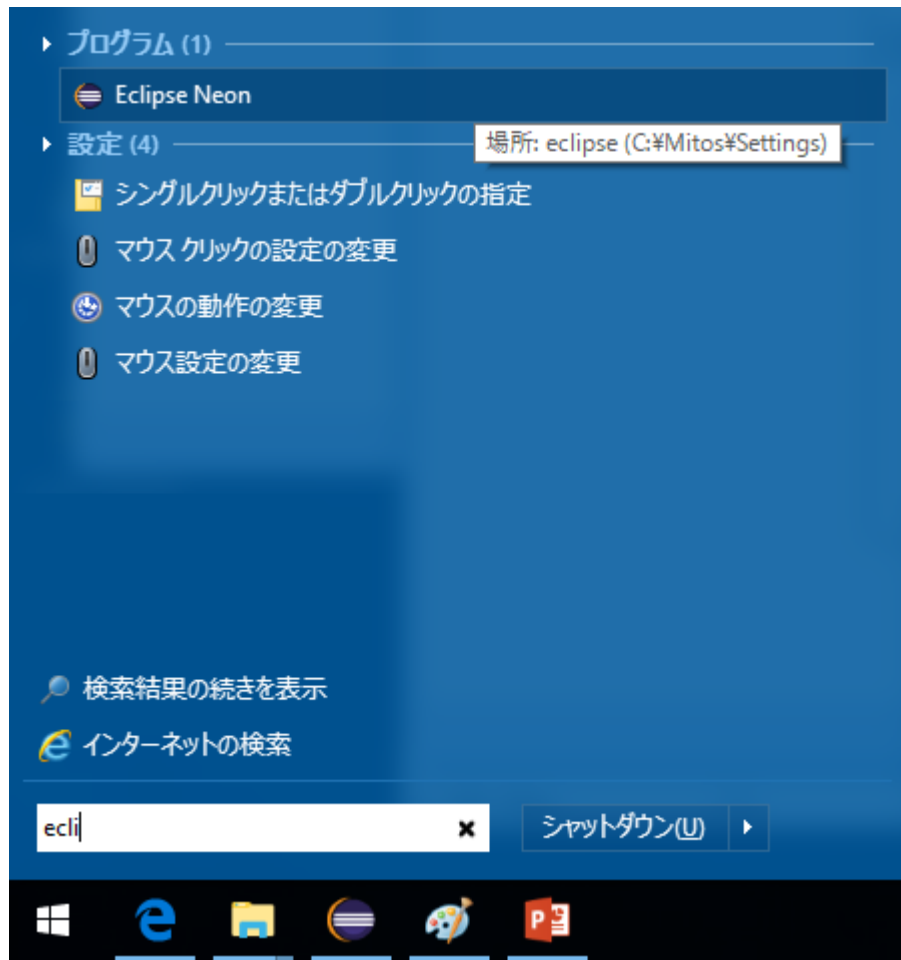
<http://mergedoc.osdn.jp/>

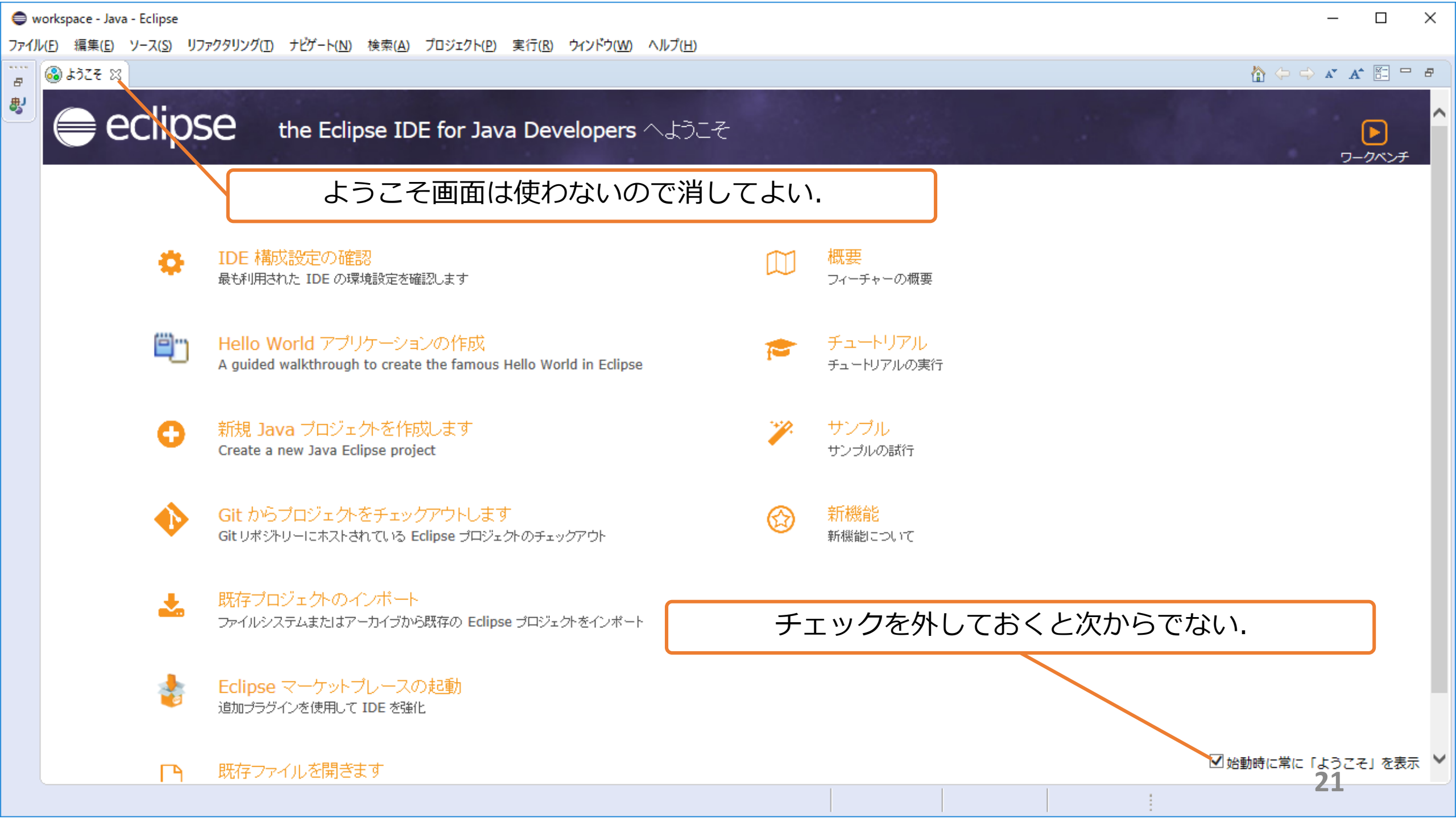
※自宅で学習したい人は上記から日本語版フリーDL可能

使用する開発環境

まずは、**Eclipse**を起動する.

> Windowsキーを押して「ecli」等を入力すると出てくるので起動する.






workspace - Java - Eclipse


ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)


ようこそ


the Eclipse IDE for Java Developers へようこそ


ワークベンチ


ようこそ画面は使わないので消してよい。


IDE 構成設定の確認
最も利用された IDE の環境設定を確認します


概要
フィーチャーの概要


Hello World アプリケーションの作成
A guided walkthrough to create the famous Hello World in Eclipse


チュートリアル
チュートリアルの実行


新規 Java プロジェクトを作成します
Create a new Java Eclipse project


サンプル
サンプルの試行

Git からプロジェクトをチェックアウトします
Git リポジトリにホストされている Eclipse プロジェクトのチェックアウト

新機能
新機能について

既存プロジェクトのインポート
ファイルシステムまたはアーカイブから既存の Eclipse プロジェクトをインポート

Eclipse マーケットプレースの起動
追加プラグインを使用して IDE を強化

既存ファイルを開きます

チェックを外しておくとか次からでない。

☒ 始動時に常に「ようこそ」を表示

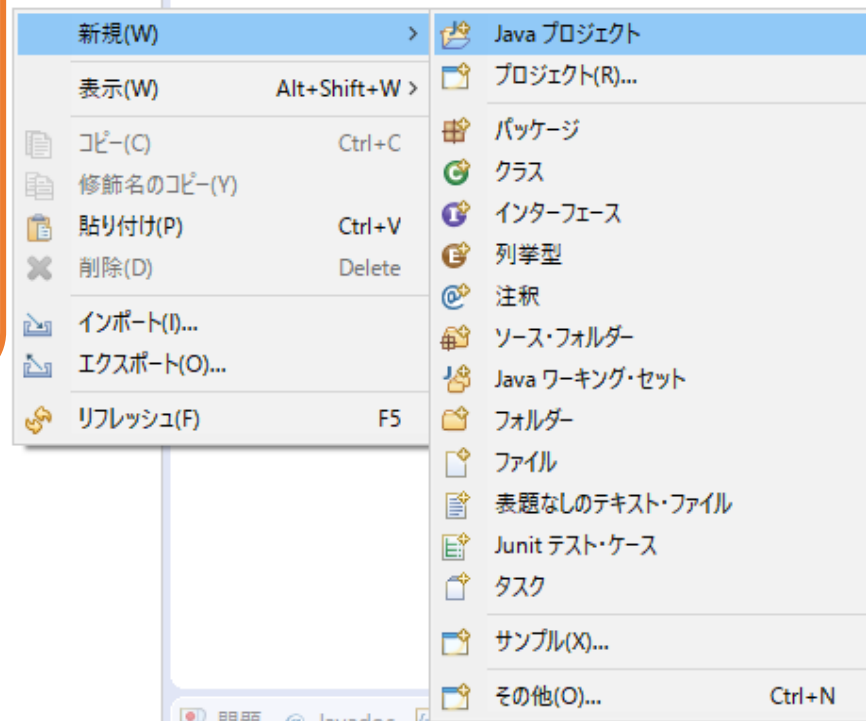
21



パッケージ・エクスプローラー

このゾーンで
1. 右クリック
2. 新規 (W)
3. Javaプロジェクト
を選択する。

早速プロジェクトを作る。



クイック・アクセス

タスク・リスト



検索

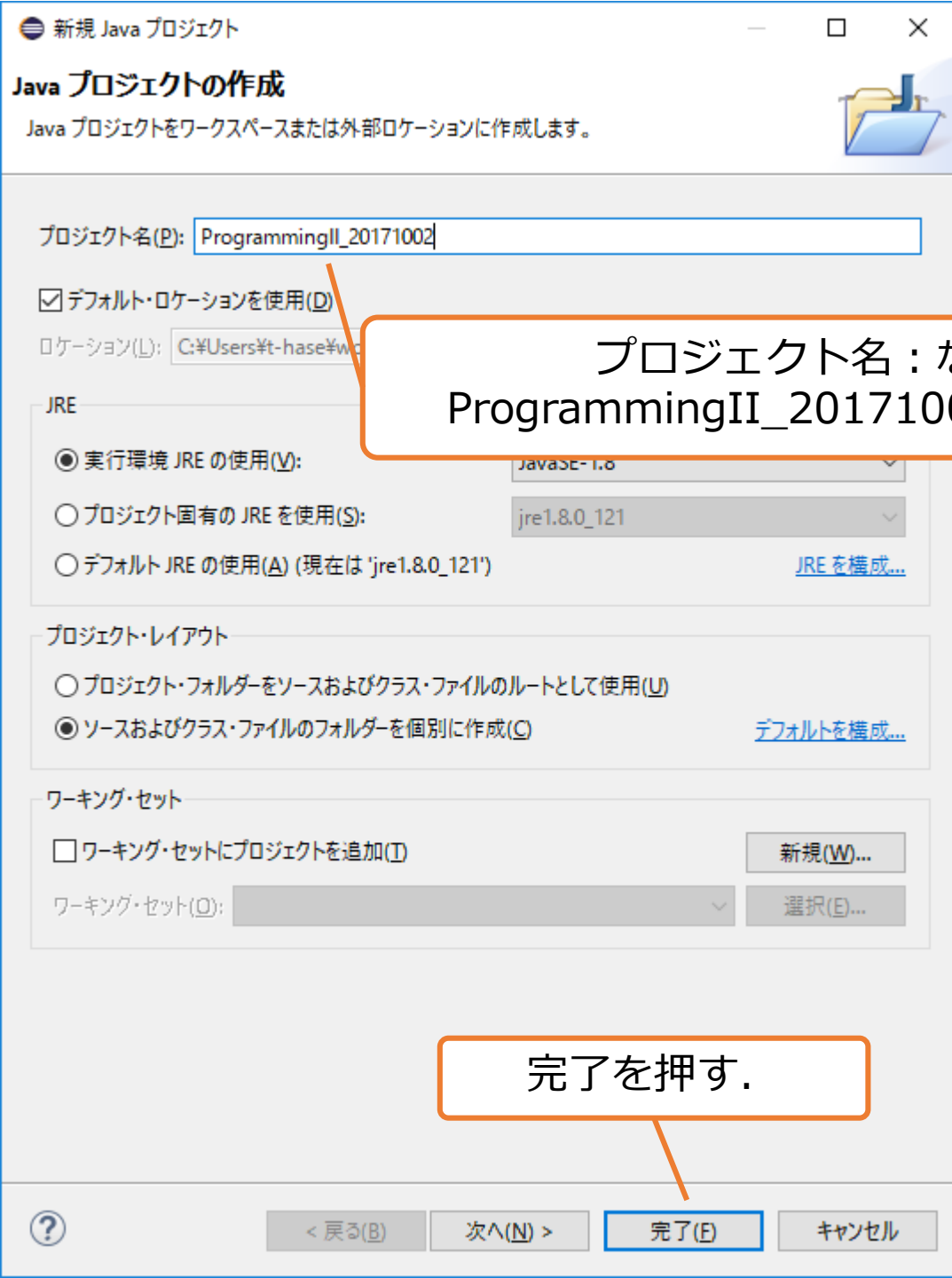
すべて アクティブに...

アウトライン

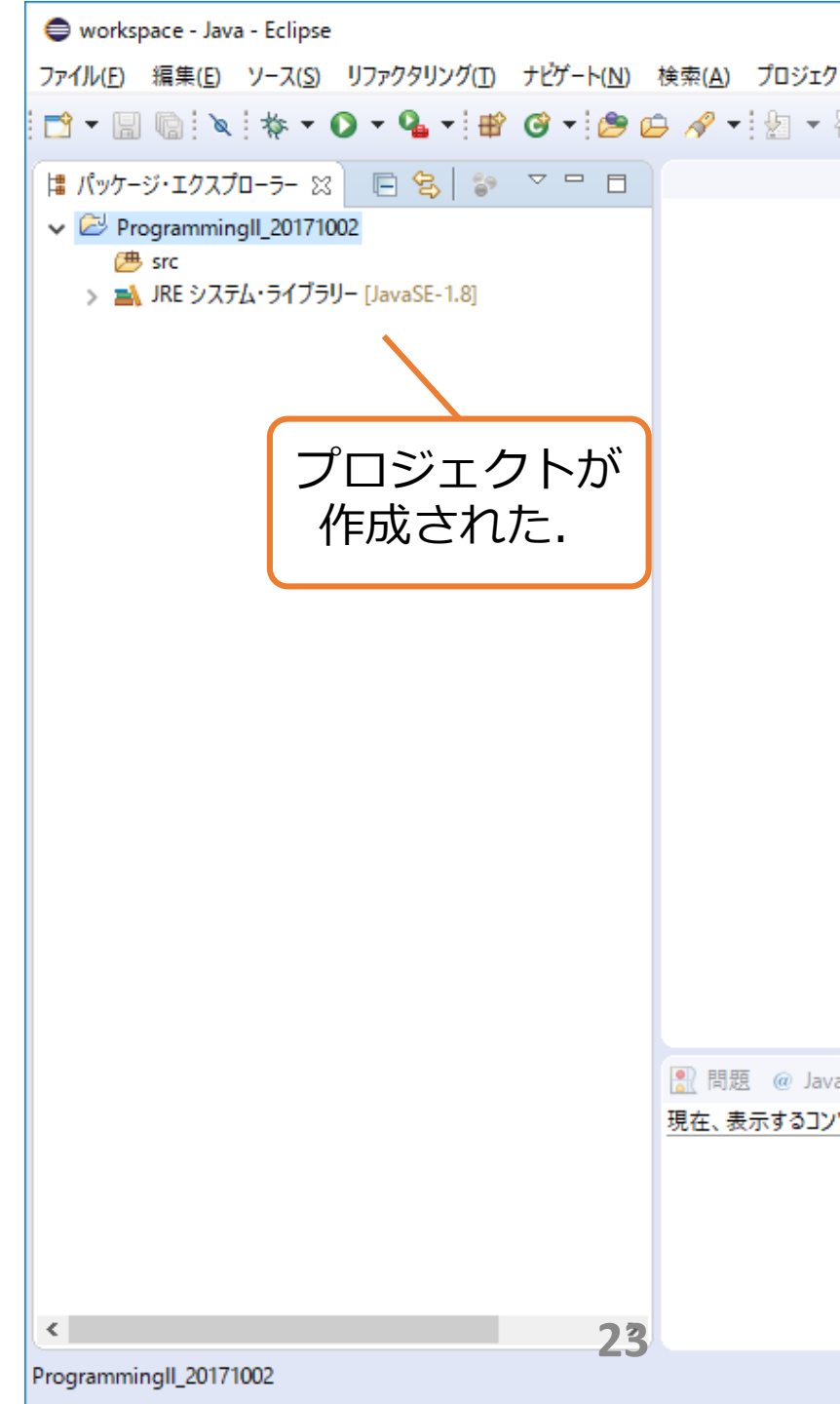
表示するアウトラインはありません。

問題 @ Javadoc

現在、表示するコンソールがありません。



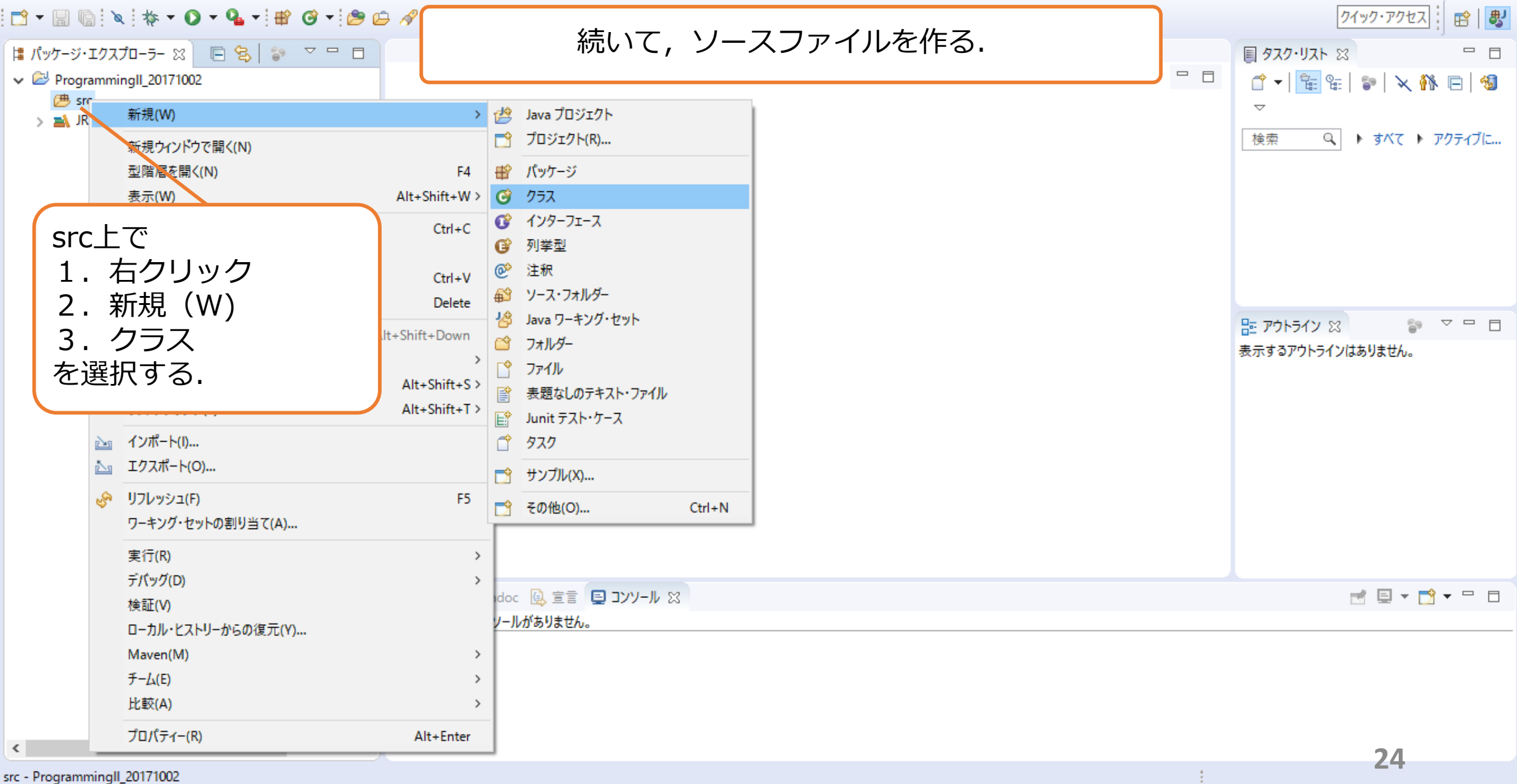
プロジェクト名：なんでもよいが
ProgrammingII_20171002としておくとよい。



プロジェクトが
作成された。

続いて、ソースファイルを作る。

src上で
1. 右クリック
2. 新規 (W)
3. クラス
を選択する。



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): ProgrammingII_20171002/src 参照(O)...

パッケージ(K):

☐ エンクロージング型(Y):

名前(M): Renshu1_1

修飾子: ☒ public ☐ パッケージ(C) ☐ private ☐ protected
☐ 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)... 除去(R)

どのメソッド・スタブを作成しますか?

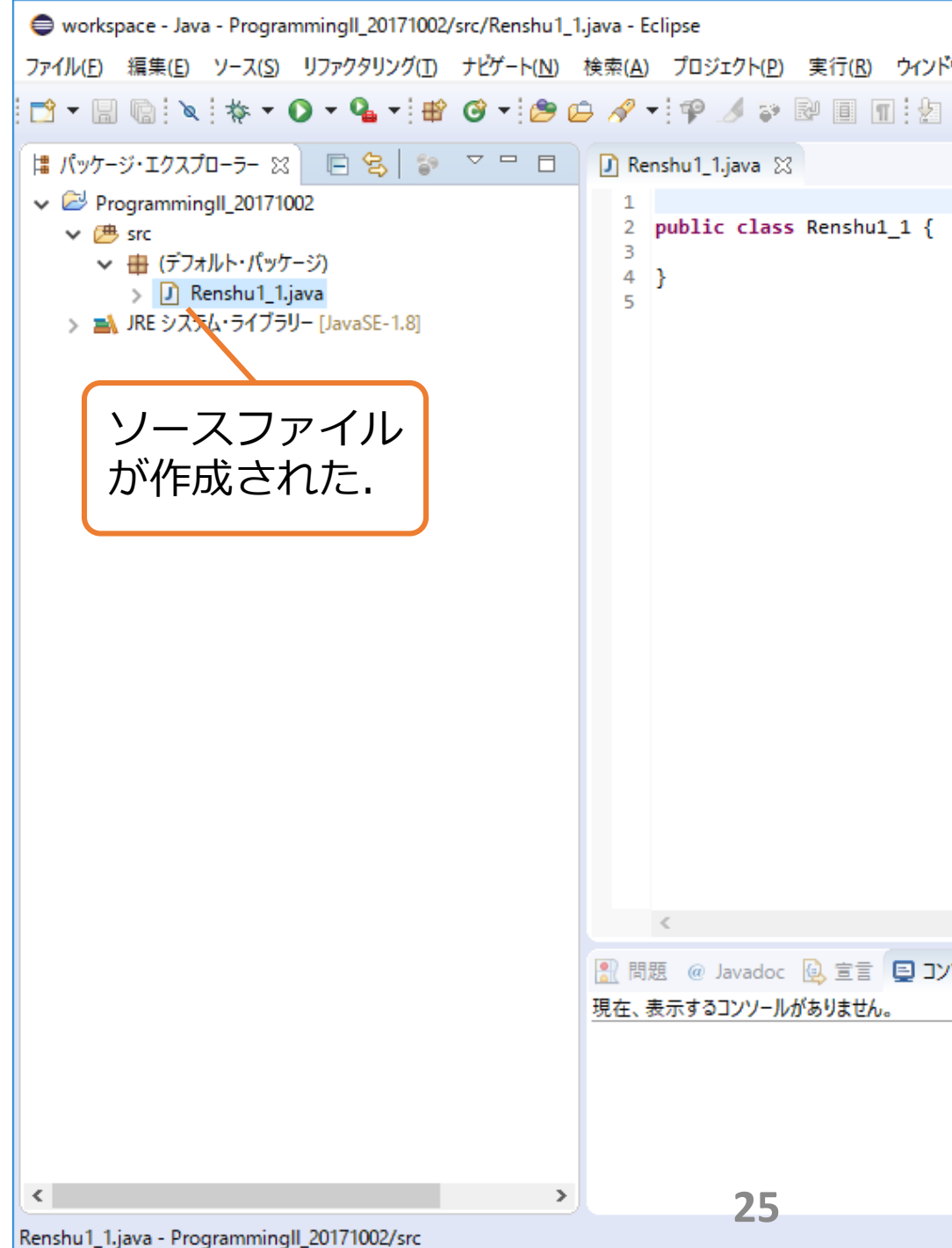
☐ public static void main(String[] args)(V)
☐ スーパークラスからのコンストラクター(U)
☒ 継承された抽象メソッド(H)

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)

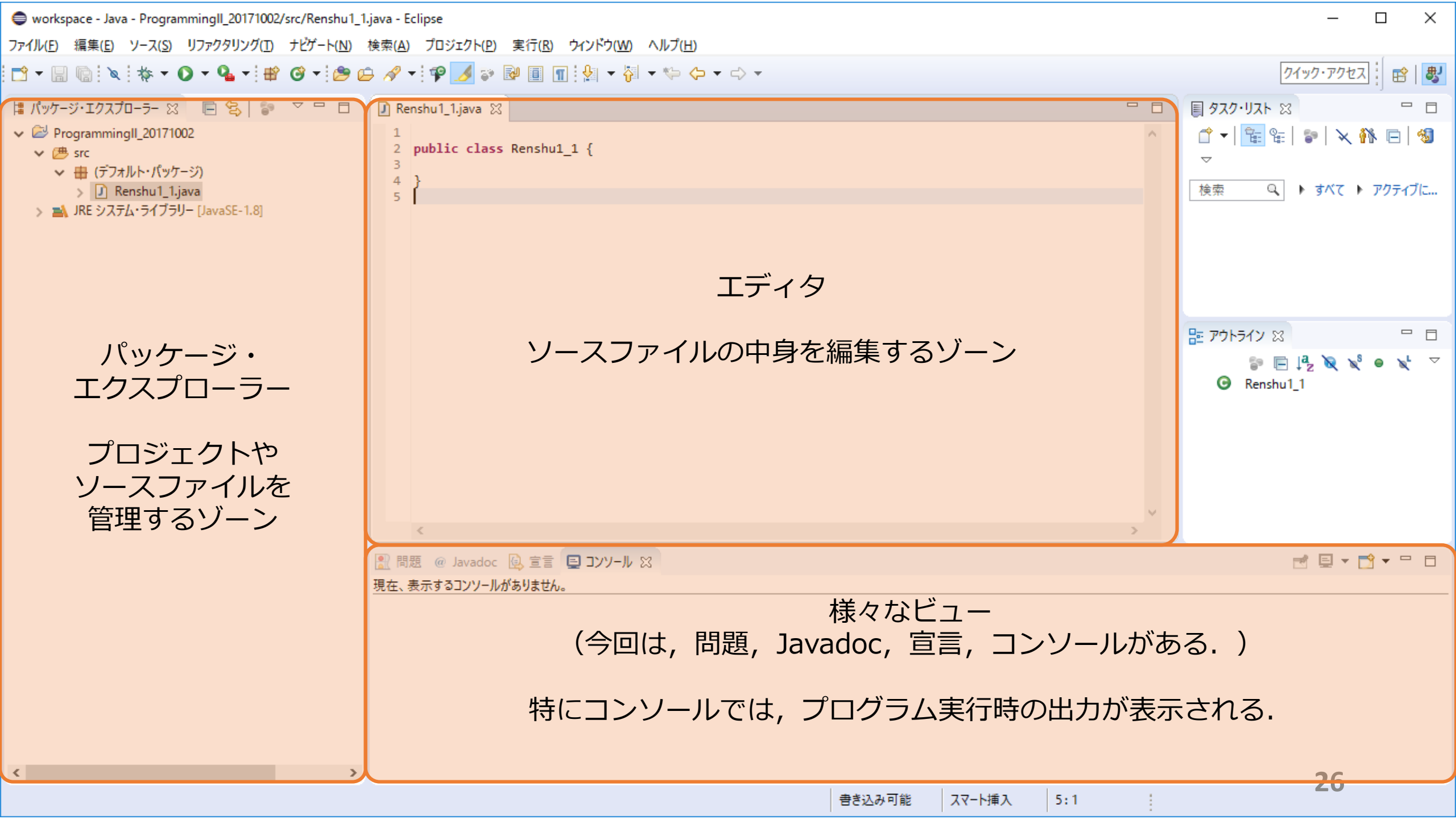
完了(E) キャンセル

ファイル名：なんでもよいが
Renshu1_1としておくとよい。

完了を押す。



ソースファイル
が作成された。



パッケージ・エクスプローラー

プロジェクトやソースファイルを管理するゾーン

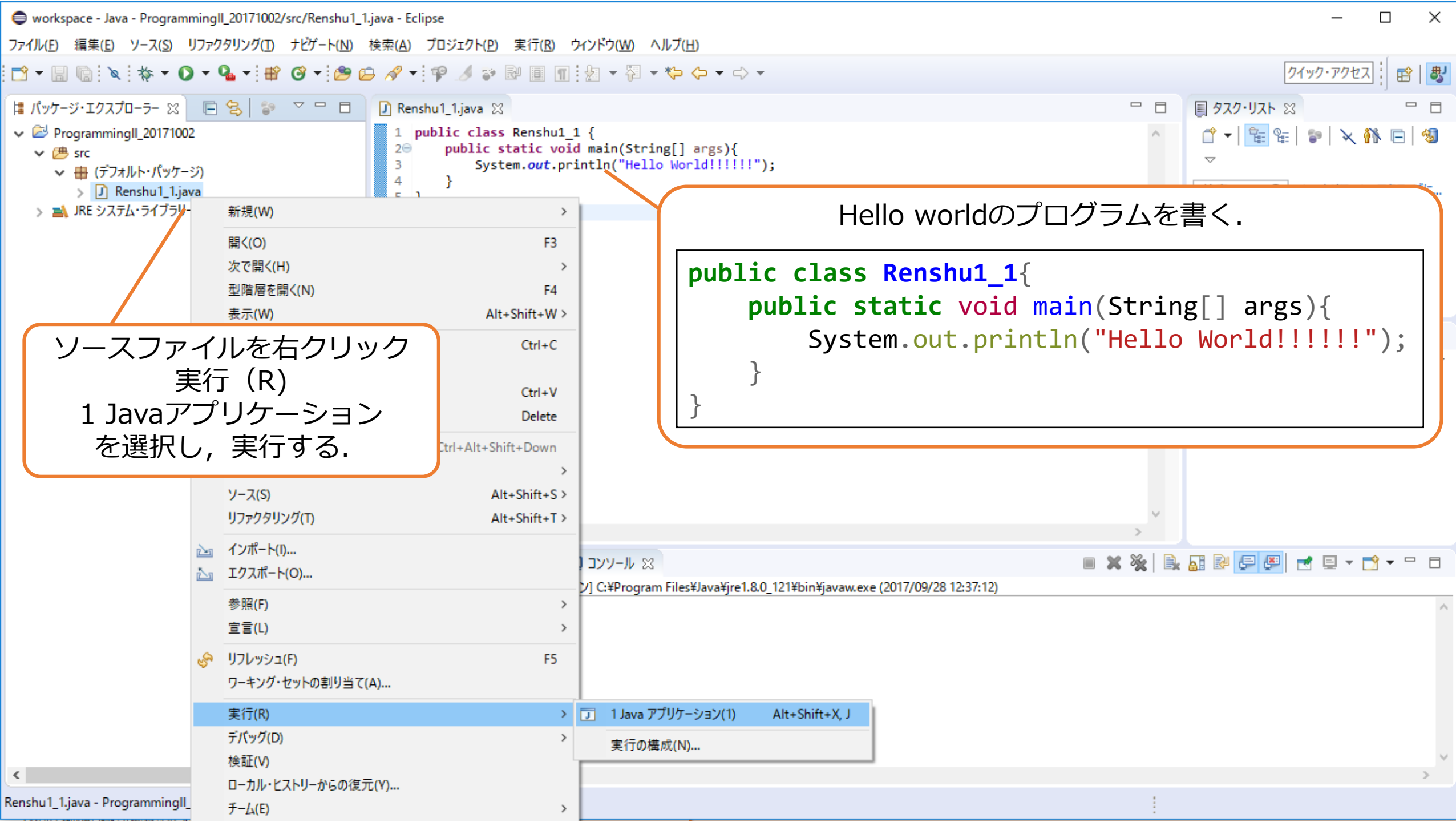
エディタ

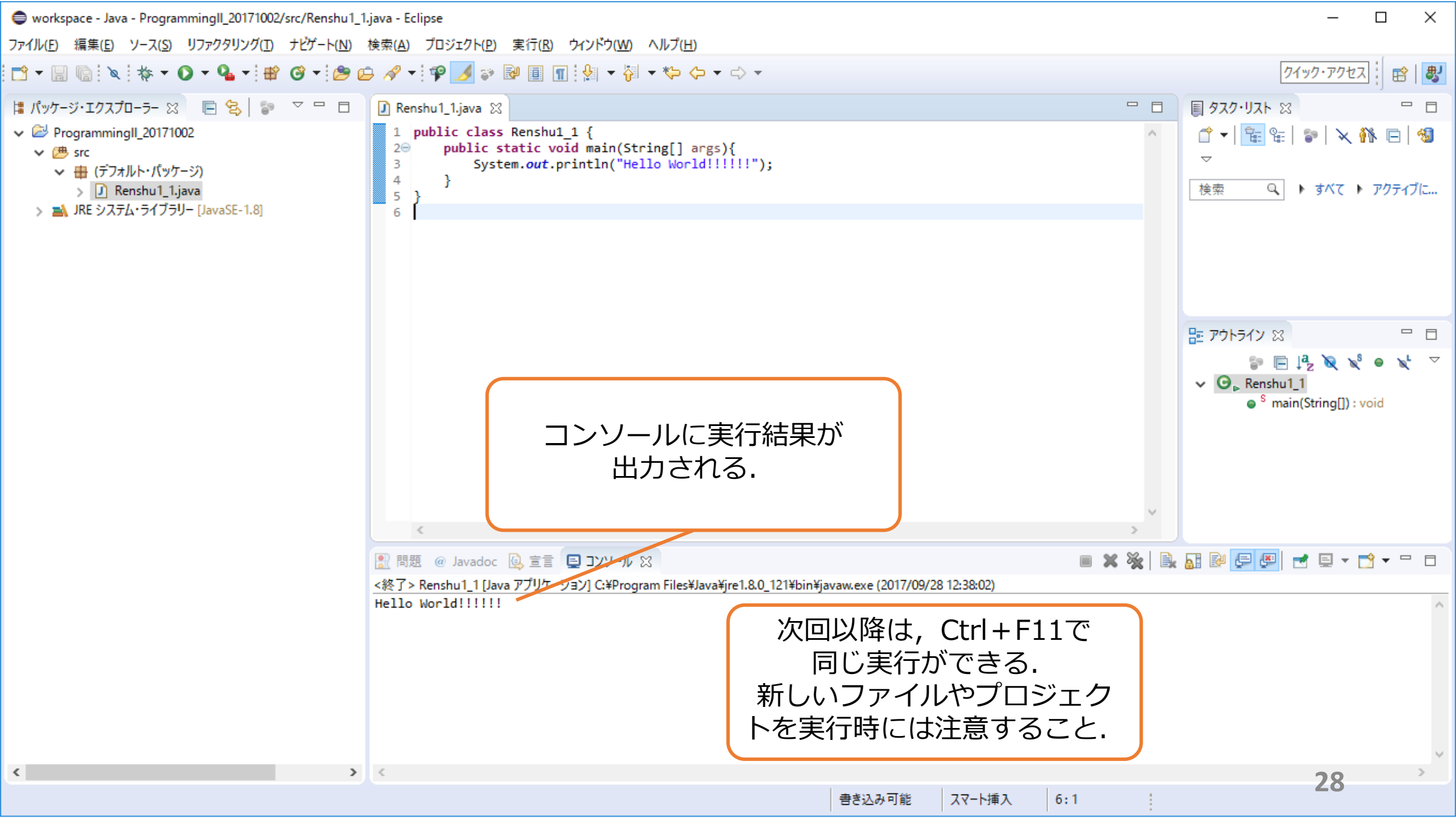
ソースファイルの中身を編集するゾーン

様々なビュー

(今回は、問題、Javadoc、宣言、コンソールがある。)

特にコンソールでは、プログラム実行時の出力が表示される。





コンソールに実行結果が
出力される。

次回以降は、Ctrl+F11で
同じ実行ができる。
新しいファイルやプロジェクト
を実行時には注意すること。

今後の運用方法

- **プロジェクトファイル**は各回ごとに作成する.
 - 次はProgrammingII_20171016
 - なお, 10月9日は体育の日 (祝日)
- **ソースファイル**は同じプロジェクト内に複数作成していく.



ここに, Renshu1_2等を追加していく.

本日の目次

- 本講義の概要
- Javaについて
- 使用する開発環境
- **Java基礎文法**

第一，二回はC言語とほとんど同じ文法を取る部分に
焦点を当てて基礎文法を紹介する（≡復習）。

本日の目次

Javaの基礎文法

- 最もシンプルなJavaプログラム
- 変数宣言と型
- 演算子, 代入演算子, 型変換
- 条件分岐 (if)
- 関係演算子, 文字列比較, 論理演算子
- 繰り返し (for, while)
- continue, break
- 条件分岐 (switch)
- 繰り返し (do-while)
- 配列, 配列for, 多次元配列
- コマンドライン引数



Java基礎文法

最もシンプルなJavaプログラム

ファイル「Sample1.java」の中身を以下とする.

Sample1.java

```
class Sample1{  
    public static void main(String[] args){  
        System.out.println("Hello world!");  
    }  
}
```

出力

Hello world!

Java基礎文法

最もシンプルなJavaプログラム

Javaプログラム記述時には次の点に気を付ける.

Sample1.java

```
class Sample1{  
    public static void main(String[] args){  
        System.out.println("Hello world!");  
    }  
}
```

出力

Hello world!

Java基礎文法

変数宣言と型

JavaにもC言語同様に様々な変数型がある。
宣言，初期化方法はC言語と基本的には同じである。

```
int i = 0;  
long l = 0L;  
float f = 0.0f;  
double y = 0.0d;  
boolean b = false;  
char c = 'A';  
String s = "文字列";
```


Java基礎文法

演算子

JavaにもC言語同様に様々な演算子がある。
宣言，初期化方法はC言語と基本的には同じである。

```
int i = 0;
i = 10 + 4; // + 加算 10+4=14
i = 10 - 4; // - 減算 10-4=6
i = 10 * 4; // * 積算 10*4=40
i = 10 / 4; // / 除算 10/4=2 (int型なので切り捨て)
i = 10 % 4; // % 余り 10%4=2 (商2, 余り2なので)
```

```
String s = "str";
s = s + "ing";
System.out.println(s);
```

文字列型変数と何かを「+」で
結ぶと文字列として結合される。

出力は文字列型変数を引数に指定するだけで良い。この場合
s = "str"+"ing"なので「string」と出力される。

Java基礎文法

代入演算子

JavaにもC言語同様に様々な代入演算子がある。
インクリメント, デクリメント演算子もある。

```
int i = 10;  
i += 4;  // + 加算 10+4=14  
i -= 4;  // - 減算 14-4=10  
i *= 4;  // * 積算 10*4=40  
i /= 4;  // / 除算 40/4=10  
i %= 4;  // % 余り 10%4=2  
String s = "str";  
s += "ing";  // + 結合 str+ing=string
```

```
i++;  // ++ インクリメント 1加算される  
i--;  // -- デクリメント 1減算される
```

Java基礎文法

型変換

なお, 大小関係は,
byte < short < int < long < float < double < String

Javaの型変換のタイミングは3種類ある.

- 代入時の自動型変換
 - 小さい型→大きい型に代入したとき, 自動的に大きい型になる.

```
short s = 10;  
int i = s;  
long l = i;
```

小さい型→大きい型
なので自動的に型変換

```
long l = 10L;  
int i = l;  
short s = i;
```

大きい型→小さい型
なのでエラー

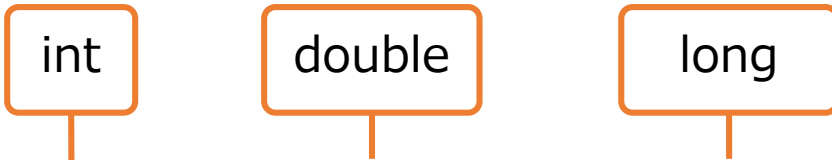
- 演算時の自動型変換 : 次頁以降で説明
- 強制的な型変換 : 次頁以降で説明

Java基礎文法

演算時の自動型変換

演算時には、**各演算で**大きい方の型に**統一されてから**計算されている。

`double d = 10 + 1.5d + 10L;`



`10.0d + 1.5d = 11.5d`



`11.5d + 10.0d = 21.5d`



Java基礎文法

強制的型変換（キャスト）

キャスト演算子を明示的に記述することで、大きい型→小さい型でも、強制的に型変換を行うことができる。

```
long l = 10000000000L;  
int i = (int)l;  
short s = (short)i;  
System.out.println(l + "," + i + "," + s);
```

(int) や (short) を
キャスト演算子と呼ぶ。



10000000000,10000000000,-13824

ただし、情報落ちすることがある。
この場合、long→intは大丈夫だが、int→shortで情報が落ちた。

Java基礎文法

強制的型変換（キャスト）

文字列→数値のキャストも可能である.

```
String str = "12345";  
int i = Integer.parseInt(str);  
i += 100;  
System.out.println("" + i);
```

String→intのキャスト

int→Stringの
演算時の自動型変換

12445

strが数字の場合は成立するが、
文字が含まれている場合エラーが発生する.

Java基礎文法

練習問題 1 : 演算時の型変換

暗黙的な型変換の利用時には注意が必要である。
以下のケースの出力を予測せよ。

```
class Sample1{  
    public static void main(String[] args){  
        double d;  
        d = 10 / 4;  
        System.out.println("d:" + d);  
        d = 10d / 4;  
        System.out.println("d:" + d);  
    }  
}
```

Java基礎文法

練習問題 1 : 演算時の型変換

次のプログラムの実行結果(1)～(3)を予測せよ.

```
class Sample1{
    public static void main(String[] args){
        int i = 10, j = 5;
        System.out.println("i+j=" + i + j);    //(1)
        System.out.println("i+j=" + (i + j));  //(2)
        System.out.println(i + j + "=i+j");    //(3)
    }
}
```


Java基礎文法

条件分岐 (if)

Javaの条件分岐はC言語同様に記述する.

今日の日付を取得している.
細かい説明はクラスの話以降で行う.

```
Calendar cal = Calendar.getInstance();  
int day = cal.get(Calendar.DAY_OF_MONTH);  
if(day % 2 == 0){  
    System.out.println("今日は" + day + "日で偶数日です. ");  
}else{  
    System.out.println("今日は" + day + "日で奇数日です. ");  
}
```



今日は2日で偶数日です.

Java基礎文法

関係演算子

if文等で条件式を書くときに使う演算子を関係演算子と呼ぶ。
関係演算子の宣言，初期化方法はC言語と基本的には同じである。

```
int i=100;
boolean flag1 = (i == 100);
boolean flag2 = (i != 100);
boolean flag3 = (i > 100);
boolean flag4 = (i <= 100);
System.out.println(flag1 + "," + flag2 + "," +
                    flag3 + "," + flag4);
```



```
true,false,false,true
```

Java基礎文法

関係演算子

したがってJavaでは、if文等で指定する条件式はboolean型の結果をもとに条件の認否を決定している。

Java

```
int day = 10;  
if(day % 2 == 0){}
```

判定

```
if(true){}
```

trueなので実行する

C言語

```
int day = 10;  
if(day % 2 == 0){}
```

判定

```
if(1){}
```

0でないので実行する

Java基礎文法

文字列の比較

なお，String型に対する比較は注意が必要である．

```
String str1 = "str";  
  
boolean flag1 = (str1 == "str");  
boolean flag2 = (str1.equals("str"));  
System.out.println(flag1 + "," + flag2);
```



true,true

「.(ドット)」 + 「関数名」
という表記はJavaで頻出するが、
詳細は第三回クラスの説明
以降で実施する．

正しく動いているが，正しく動かないことがあり得るため、
文字列の一致比較は==を用いないようにする．

Java基礎文法

論理演算子

ANDは「&&」，ORは「||」，NOTは「!」を用いてboolean型変数を「0:false, 1:true」と見たときに論理演算ができる。

```
boolean flagT = true;  
boolean flagF = false;  
  
boolean flagA = flagT && flagF;  
boolean flagO = flagT || flagF;  
boolean flagN = !flagT;  
System.out.println(flagA + "," + flagO + "," + flagN);
```



false,true,false

Java基礎文法

練習問題 2 : mainメソッド

先ほど覚えたmainメソッドを思い出して、下記に述べよ.

Java基礎文法

繰り返し (for)

Javaの繰り返し (for) はC言語同様に記述する.

```
int sum = 0;
for(int i=0;i<10;i++){
    sum += i;
}
System.out.println("0から9の合計は"+sum);
```



0から9の合計は45

Java基礎文法

繰り返し 2 (while)

Javaの繰り返し (while) はC言語同様に記述する.

```
boolean flag = true;
int i = 0;
while(flag){
    System.out.println("i=" + i);
    if(i>10000)
        flag = false;
    i++;
}
```

flag が true の間ループするという意味
「flag == true」 と同義である.

if内の処理が1行の時に限り、鍵括弧を付けないこのような記述が可能である.
ただし、可読性低下のため**非推奨**である.

i=0
i=1
...(略)

Java基礎文法

continueとbreak

処理をスキップするcontinueとループを終了するbreakがある.

```
int i=0;
while(true){
    i++;
    if(i<10)    continue;
    System.out.println("実行中:" + i);
    if(i>100)   break;
}
```



```
実行中:10
実行中:11
...(略:101まで)
```

Java基礎文法

練習問題 3 : ifとforとcontinue

for文を用いて $\sum_{i=1}^{100} i$ を計算し表示するプログラムを実装せよ。
ただし、 i が5の倍数の時、continue文を用いて加算をスキップせよ。

Java基礎文法

条件分岐 2 (switch)

Javaの条件分岐 (switch) はC言語同様に記述する.

```
String str = "str";
switch(str){
case "a":
    System.out.println("Uhm..");
    break;
case "str":
    System.out.println("Great");
    break;
default:
    System.out.println("Uhm..");
    break;
}
```

```
String str = "str";
if(str.equals("a")){
    System.out.println("Uhm..");
}else if(str.equals("str")){
    System.out.println("Great");
}else{
    System.out.println("Uhm..");
}
```

switchもifも同じ条件分岐が書ける.
ただし, switchは**1変数**を**定数**と比較する
場合のみ利用可能である.

Java基礎文法

条件分岐 2 (if-switchの比較)

If

使える条件に制限がない

シンプルで可読性が高い

二分岐では高速

多分岐では低速

同じ条件がelse ifにあっても
エラーにならない

Switch

1変数対定数の一致比較のみ

break;を余計に書く必要がある

二分岐でも定速

多分岐でも定速

同じ条件がcaseにあると
エラーになる

従って、基本はifで書くが、一変数多分岐の際はSwitchを使うと効率が良い

Java基礎文法

繰り返し3 (do while)

Javaの繰り返し (do while) はC言語同様に記述する.

```
boolean flag = true;
int i = 0;
do{
    System.out.println("i=" + i);
    if(i>10000)
        flag = false;
    i++;
} while(flag);
```



```
i=0
i=1
...(略:10001まで)
```

Java基礎文法

配列

配列の概念はC言語とほとんど同じである。
しかし宣言方法はC言語と少々異なる。

配列の宣言はこの書き方を暗記すること。
変数型名[] 配列名 = **new** **変数型名**[要素数]

要素数を変数で（動的に）定義可能である。
C言語ではmallocやcallocを使う必要があった。

```
int i=10;  
int[] array = new int[i];  
System.out.println("array[0]:" + array[0]);  
System.out.println("length:" + array.length);
```

要素の参照方法はC言語と同じである。
自動的に0で初期化してくれている。

配列名.lengthで配列の長さが取得可能である。
.length()ではないことに注意されたい。

array[0]:0
length:10

Java基礎文法

配列 (forと拡張for)

配列の参照にはfor文を使うことが多く、C言語同様に書ける (1) .
Javaでは拡張for文 (for-each文) という使い方ができる (2) .

```
String[] array = {"", "", "", "", ""};  
for(int i=0;i<array.length;i++){ //(1)  
    array[i] = ""+i;           //文字列としてiを格納する  
}  
for(String str : array){ //(2)  
    System.out.print(str + ",");  
}
```

str = "a"; 等, 代入処理をしても,
array側は更新されないことに注意されたい.

0,1,2,3,4,

print("")は出力を改行しない.
println("")は最後に改行する.

Java基礎文法

配列（多次元配列）

多次元配列は次の書き方で宣言が可能である.

```
String[][] array = new String[2][3]; //2行3列

System.out.println("array[0][1]:" + array[0][1]);
System.out.println("array.length:" + array.length);
System.out.println("array[0].length:" + array[0].length);
```



```
array[0][1]:null
array.length:2
array[0].length:3
```


Java基礎文法

コマンドライン引数

メインメソッドの引数(`String[] args`)は暗記するとしてきたが、実はコマンドラインから実行した際に送られてくる引数である。

```
public class Sample1{  
    public static void main(String[] args){  
        System.out.println("args.length:" + args.length);  
        int i=0;  
        for(String str : args){  
            System.out.println("args[" + i + "]: " + str);  
            i++;  
        }  
    }  
}
```

このこと

`args[]`の要素数と要素全てを表示する処理

Java基礎文法

コマンドライン引数

コマンドプロンプトから、Javaを実行する時に引数を指定すると次のような動作をする.

```
java Sample1
```

引数なしで実行

0

```
java Sample1 aa ii uu
```

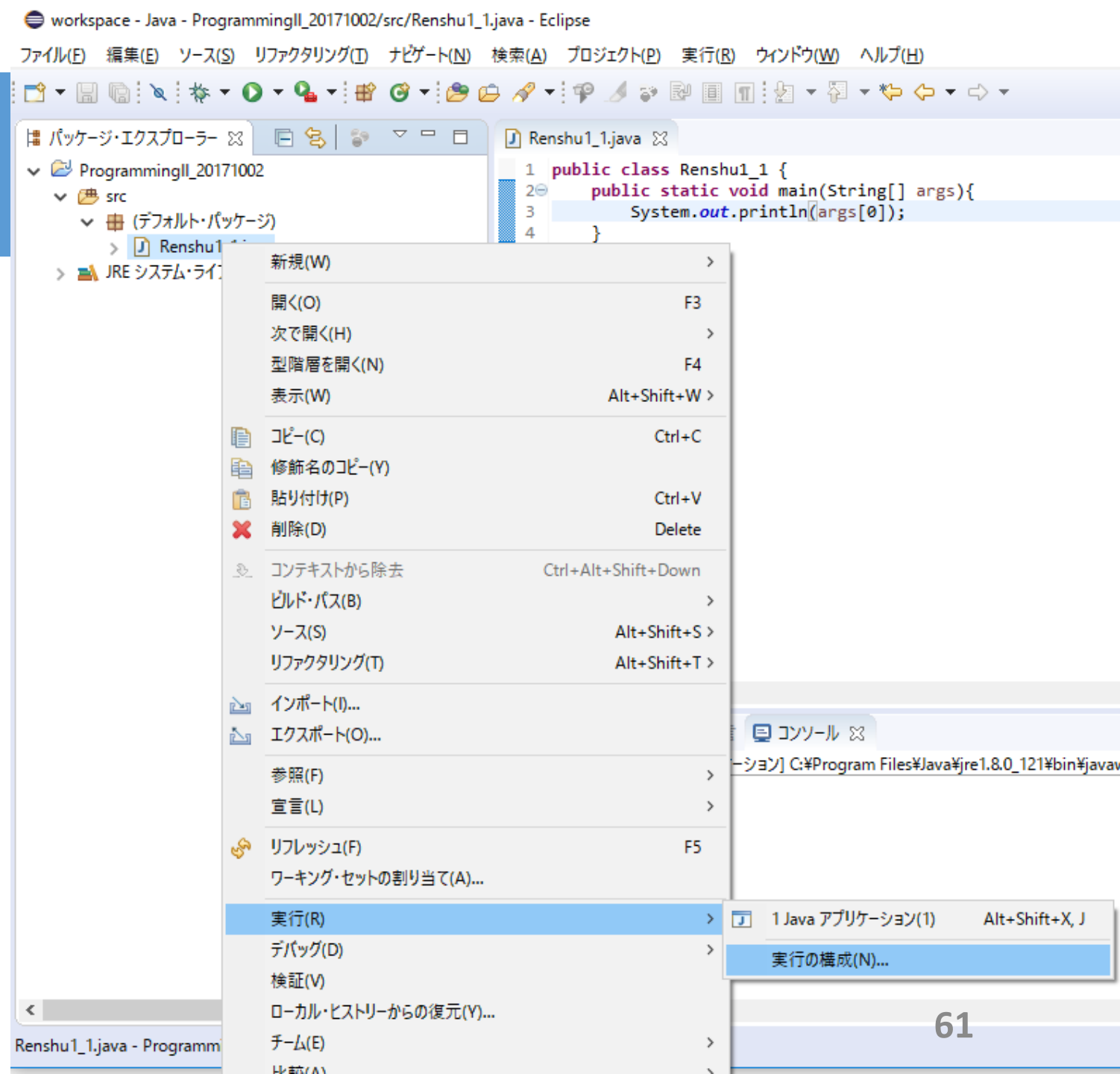
3引数(aa, ii, uu)で実行

```
3  
aa  
ii  
uu
```

Java基礎文法

コマンドライン引数

Eclipseでコマンドライン引数を用いて実行するには、
ソースファイル右クリック
実行 (R)
実行の構成 (N)
を選択し、



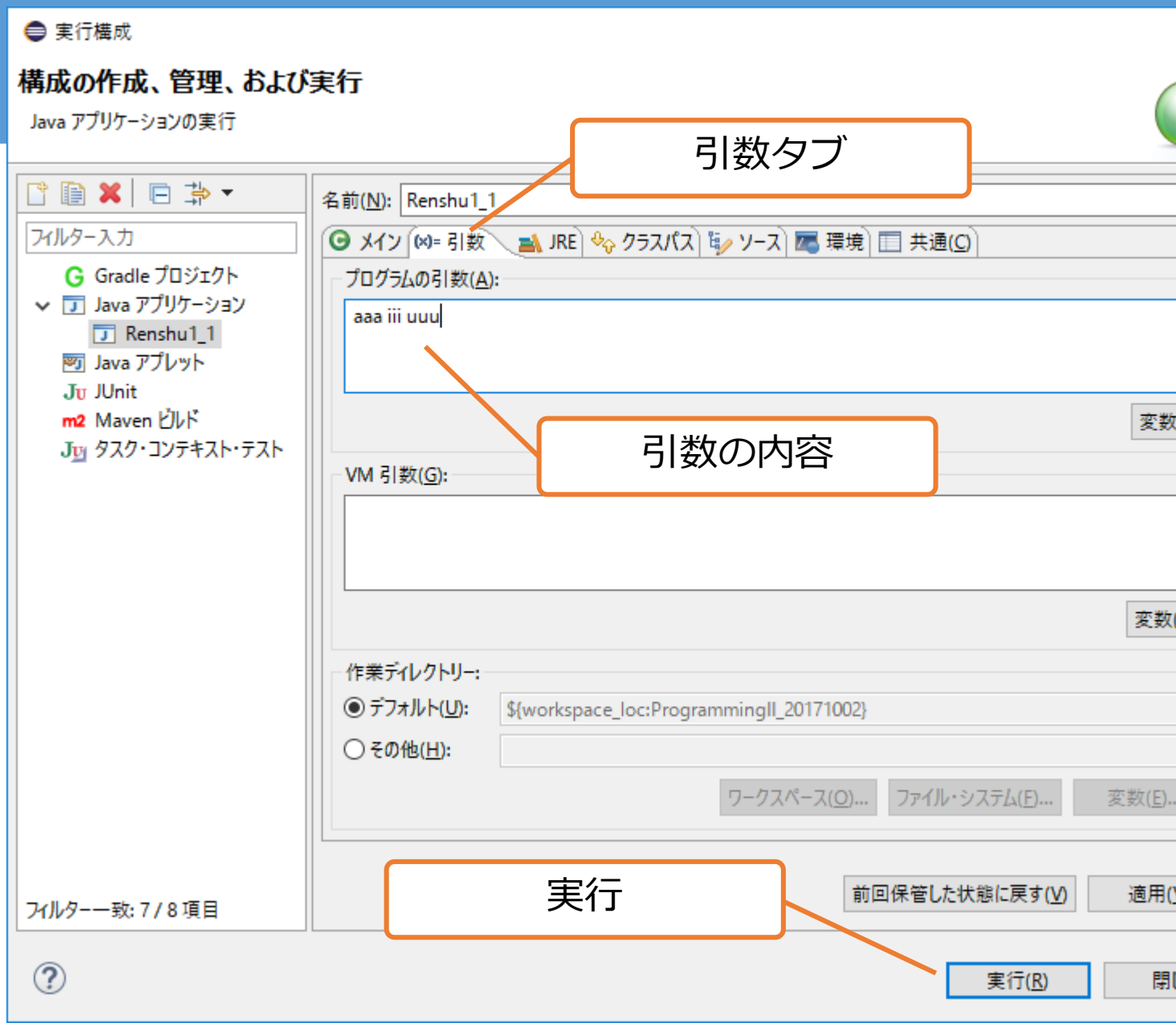
Java基礎文法

コマンドライン引数

引数タブを押下すると、
プログラムの引数 (A)を
設定することができる。

ここで実行を押下すると、
>java Renshu1_1 aaa iii uuu
と同じ動作となる。

args[0]に"aaa"が、
args[1]に"iii"が、
args[2]に"uuu"が
格納された状態でプログラム
が実行される。



本日のまとめ

- Javaの背景と，基本文法を紹介した．
- 基本文法はほとんどC言語と書き方が同じである．
- 一部Java独自の技法があるため，しっかりと理解されたい．
- 次週以降は計算機室で実施する．

わかりました



次週予告

※次週以降は計算機室

前半

本日の残りを説明した後，Javaプログラムの作成方法と開発環境の使い方を紹介する．

※**10月16日の資料**は本日のものを使いますので**印刷不要**である．

後半

Javaの基本文法（本日の内容）に関するプログラミング課題を実施する．

本日の提出課題

課題 1

本日の授業を聞いて、
初めて知ったと思う内容を2点簡潔に述べよ。
「簡潔に述べよ」⇨「1, 2文程度で述べよ」と考えるとよい。

課題 2

本日の授業を聞いて、
質問事項または**気になった点**を2点簡潔に述べよ。

課題 3

資料を見ずに、mainメソッドを書け（間違えてもよい）。