

プログラミング基礎 I

第 1 回

変数・代入・演算

九州産業大学理工学部

pk@is.kyusan-u.ac.jp

第1回講義概要

- 変数
- 代入
- 演算
 - 数値演算、文字列演算

変数

- データを格納する入れ物
- 変化するデータを扱うのに使う
- 自分で変数を作ることできる

変数を作る（変数宣言）

- 変数宣言の構文

型 変数の名前;

- 型** 格納するデータの種類

- 変数の名前** 自分でつける

- 例 整数型の変数score

```
int score;
```

整数型の変数x, y

```
int x, y; （複数まとめて宣言）
```



score



x



y

変数名の命名ルール

- **全て半角英数**とする。全角文字を含んではならない。
 - `int 金額; // エラーにはならないが、変数名に全角を使ってはならない。`
- **最初の文字は小文字英字**とする
 - `int score;` はOK、 `int Score;` や `int SCORE;` は×
- 複数の単語を繋げた変数名とする場合は、**最初の単語を小文字、それ以降の最初の文字を大文字**とする。
変数名に記号は使用しない（ただし定数を除く）
 - `int mathScore; String sendMessage;`
 - このような記法を、**小文字キャメルケース(lowerCamelCase)** と言う

プログラミングをする際のルールについて

- 複数人で開発を行う場合、
全員で守るコーディングのルールを決めることが多い
 - 全員バラバラだと、コードが読みにくくなり、開発効率を下げることになる
- コーディング規約
 - コーディングの際にチーム内で書き方を統一するためのガイドライン
 - Javaの場合には、OracleやGoogleのコーディング規約がある他、上記の規約をカスタムした各会社独自の規約が存在することが多い

Javaコーディング規約

- Code Conventions for the Java TM Programming Language
 - <https://www.oracle.com/java/technologies/javase/codeconventions-contents.html>
- Google Java Style Guide
 - <https://google.github.io/styleguide/javaguide.html>
- この授業でも、一部についてコーディング規則を設ける
 - 変数名、メソッド名、クラス名の命名規則、制御構文の記法(if, for, while)
 - 基本的にはGoogle Java Style Guideに準じるが、この授業のルールが絶対的なものとの誤解はしないように注意
 - 実際の開発現場では、そのプロジェクト内でのルールが決められている

型の種類 教科書P30

分類	名前	値
整数型	byte	8ビット整数 -128～127
	short	16ビット整数 -32768～32767
	int	32ビット整数 -2147483648～2147483647
	long	64ビット整数 -9223372036854775808～ 9223372036854775807
小数型	float	32ビット小数
	double	64ビット小数
文字型	char	1文字
文字列型	String	文字列
真偽値型	boolean	true(真)/false(偽)

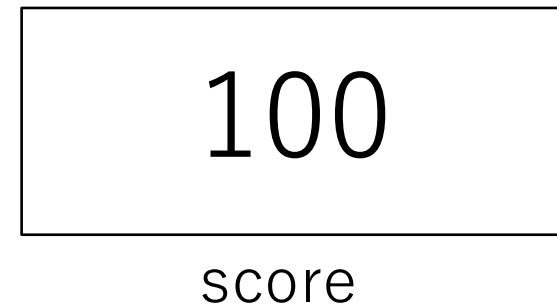
値を格納する（代入）

- 代入の構文

変数の名前 = データ;

- データは具体的な数値であったり式であったり
- 例 scoreという変数に100を代入

score = 100;



変数の初期化

- 変数の宣言と初期値（変数に最初に格納される値）の格納
- 初期化の構文

型 変数の名前 = データ;

- 例：整数型の変数scoreを宣言し、100で初期化

```
int score = 100;
```

算術演算子 教科書P34

演算子	意味	例
+	加算（足し算）	$5+3 \rightarrow 8$
-	減算（引き算）	$10-3 \rightarrow 7$
*	乗算（掛け算）	$3*2 \rightarrow 6$
/	除算（割り算） （※整数演算では商は整数）	$3.2/2 \rightarrow 1.6$ $9/2 \rightarrow 4$
%	剰余（割り算の余り）	$9\%2 \rightarrow 1$

変数の値の変更(1/2) 教科書P38

- 例：scoreの値を10増やす

```
score = score + 10;
```

- 「=」は左辺の変数に右辺の値を代入する → **代入演算子**
- scoreに10を足したものを、左辺のscoreに代入する

```
score += 10;
```

- **複合代入演算子**（算術演算子とイコールを組み合わせたもの）を利用
- +=（加算代入） -=（減算代入） *=（乗算代入） /=（除算代入） %=（剰余代入）

変数の値の変更(2/2) 教科書P40

- 例：scoreの値を1増やす：インクリメント演算子

score++;

- 例：scoreの値を1減らす：デクリメント演算子

score--;

教科書 P38 リスト3.31 P40 リスト3.32

変数

- データを格納する入れ物
- 変数を使うためには、まず変数を宣言する
- 変数宣言の構文 型 名前;
- 変数宣言の例
 - `int x;    //int型の変数 x を宣言`
 - `int num;    //int型の変数num`
 - `String msg;    //String型の変数 msg`
 - `double vol;    //double型の変数 vol`
 - `boolean check;    //boolean型の変数 check`

代入

- 変数に値（データ）を格納すること
- 代入の構文は、変数名 = データ;
- 代入の例
 - `x = 0;` `//xに 0 を代入`
 - `num = 10;` `//numに 10 を代入`
 - `msg = "OK";` `//msgに OK を代入`
 `//（文字列はダブルクォート（" "）で囲む）`
 - `vol = 0.5;` `//volに 0.5 を代入`
 - `check = false;` `//check に false を代入`
 - `num = x + 5;` `//numに x+5 を代入(変数に変数を代入)`

演算

- 変数には値（データ）だけでなく、演算結果も代入できる

- 例

- `x = num + 20;` `//x に num + 20 を代入`
- `x = x + 20;` `//x に x + 20 を代入`
- `x += 10;` `//x に x + 10 を代入（複合代入演算子を使用）`
- `vol = vol * 0.1;` `//vol に 0.1 をかける（vol *= 0.1; も同じ結果）`
- `vol *= vol;` `//vol に vol をかける（vol = vol * vol;）`

文字列や変数を画面に表示する

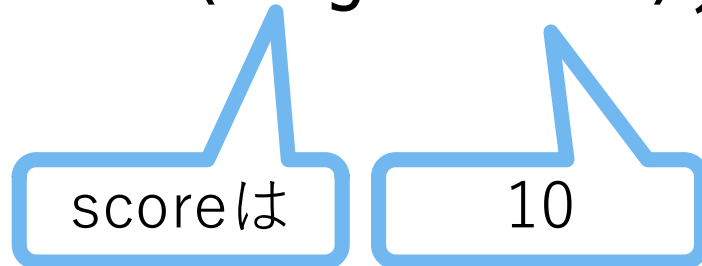
- Javaでは、
`System.out.print("文字列");` //表示後改行無し
`System.out.println("文字列");` //表示後改行する
の命令でコンソールに文字列を表示出来る
- Eclipseでは、
`sysout` と入力して、 `Ctrl + スペースキー`を押すと、
`System.out.println` に自動変換してくれる（少し楽）

文字列、数値の連結

- **+演算子**で、文字列と文字列、文字列と数値を連結

```
String msg = "scoreは"; //msgを宣言、文字列を代入  
System.out.println(msg); //scoreは と表示される
```

```
int val = 10; //valを宣言し、10を代入  
System.out.println(msg + val); // scoreは10
```



文字列の連結（文字列型変数への代入）

- 文字列型の変数に文字列を連結する

```
String msg = "scoreは"; //msgを宣言、文字列を代入
```

```
int val = 10;
```

```
msg += val; // 複合代入演算子で変数に文字列を連結
```

```
System.out.println(msg); // scoreは10
```

```
msg += "点";
```

```
System.out.println(msg); // scoreは10点
```

文字列連結の例

- 例

- `msg = x + "point";` //msgに x の値と “point” を結合した文字列を代入
- // 2つの変数を半角スペースで繋げて代入
`String lastName = "理工";`
`String firstName = "太郎";`
`String fullName = lastName + " " + firstName;`
- `String resultString = "";` //変数resultStringに空文字を代入
`resultString += x1 + ",";` //x1,
`resultString += x2 + ",";` //x1,x2,
`resultString += x3 + ",";` //x1,x2,x3,

複数の変数の値を
繋げていく

文字列演算：文字列を対象とした演算

- 文字列の長さを取得する：String.length メソッド
 - ```
String msg = "k25rs999@st.kyusan-u.ac.jp";
int msgLength = msg.length();
System.out.println("msgの長さ:" + msgLength); //結果: msgの長さ26
```
- 文字列中の部分文字列を検索：String.indexOf メソッド
  - 文字列の中に、部分文字列が含まれる場合には、  
その部分文字列が始まる番号(index)を返し、  
含まれない場合には -1 を返す  

```
int atMarkIndex = msg.indexOf("@"); //msg中の@がある番号を探す
System.out.println("@は" + atMarkIndex + "文字目から開始");
// 結果: @は8文字目から開始
```

# 文字列の番号 (index)

- 8文字目？ kから数えると9番目じゃないの？

```
String msg = "k25rs999@st.kyusan-u.ac.jp";
int atMarkIndex = msg.indexOf("@");
System.out.println("@は" + atMarkIndex + "文字目から開始");
//結果： @は8文字目から開始
```

- 文字列の開始位置、終了位置の番号は **0 から開始** する
  - Javaでは、文字列の最初の文字は0番目として数える

|       |   |   |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|---|---|
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 文字列   | k | 2 | 5 | r | s | 9 | 9 | 9 | @ | s |

- このケースだと、0から数え始めると @ の番号(インデックス)は8

# 文字列演算

- 文字列の一部分を切り取る

- **substring**メソッド

1文字目は0

- 対象の文字列**.substring(開始位置, 終了位置)**;  
開始位置から終了位置の手前までの文字列を切り取って取得する。

```
String target = "abcdefg";
String subStr0_4 = target.substring(0, 4);
System.out.println(subStr0_4); //結果: abcd
String subStr2_6 = target.substring(2, 5);
System.out.println(subStr2_5); //結果: cde
```

- 文字列の開始位置、終了位置の**番号は 0 から開始**する

# length, indexOf, substringの合せ技

- 例：メールアドレスのユーザ名だけを取得

```
String mailAddress = "k25rs999@st.kyusan-u.ac.jp";
int msgLength = mailAddress.length();
int atmarkIndex = mailAddress.indexOf("@");
```

0番目から @ のある位置 **未満** まで

```
String userName = mailAddress.substring(0, atmarkIndex);
System.out.println(userName);
```

|   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| k | 2 | 5 | r | s | 9 | 9 | 9 | @ | s | t  | .  | k  | y  | u  | s  | a  | n  | -  | u  | .  | a  | c  | .  | j  | p  |

- 例：ドメインだけを取得

```
String domainName = mailAddress.substring(atmarkIndex+1, msgLength);
System.out.println(domainName);
```

切り取りたいのは、  
@の次の文字からなので+1

msgLength(文字列の長さ)は26、  
最後の文字のindexは25

# 文字列 → 数値変換

- 文字列を数値に変換(キャスト)する
  - 文字列 → 整数値 : `Integer.parseInt ("文字列")`
  - 文字列 → 少数値 : `Double.parseDouble ("文字列")`
- ```
String str = "20";  
int strToInt = Integer.parseInt (str);  
double strToDouble = Double.parseDouble(str);
```
- 数値に出来ない文字列を変換しようとするとうエラーになることに注意

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "a"  
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)  
    at java.base/java.lang.Integer.parseInt(Integer.java:660)  
    at java.base/java.lang.Integer.parseInt(Integer.java:778)  
    at Main25RS999.main(Main25RS999.java:10)
```

演習

- Eclipse起動
- Javaプロジェクト作成
- クラス作成
- キーボード入力受取

統合開発環境

- プログラムを開発するために必要なものが揃っているアプリケーション
 - コードエディタ（テキストエディタ）
 - コンパイラ
 - デバッガー
- EclipseやProcessingは統合開発環境にあたる
 - それ一つインストールしておけば、プログラム開発を行える
- その他の統合開発環境（一例）
 - Visual Studio(C,C++,C#など)、Unity、IntelliJ IDEA、Android Studio、XCode、Thonny（Python）、PyCharm、その他色々...

テキストエディタ

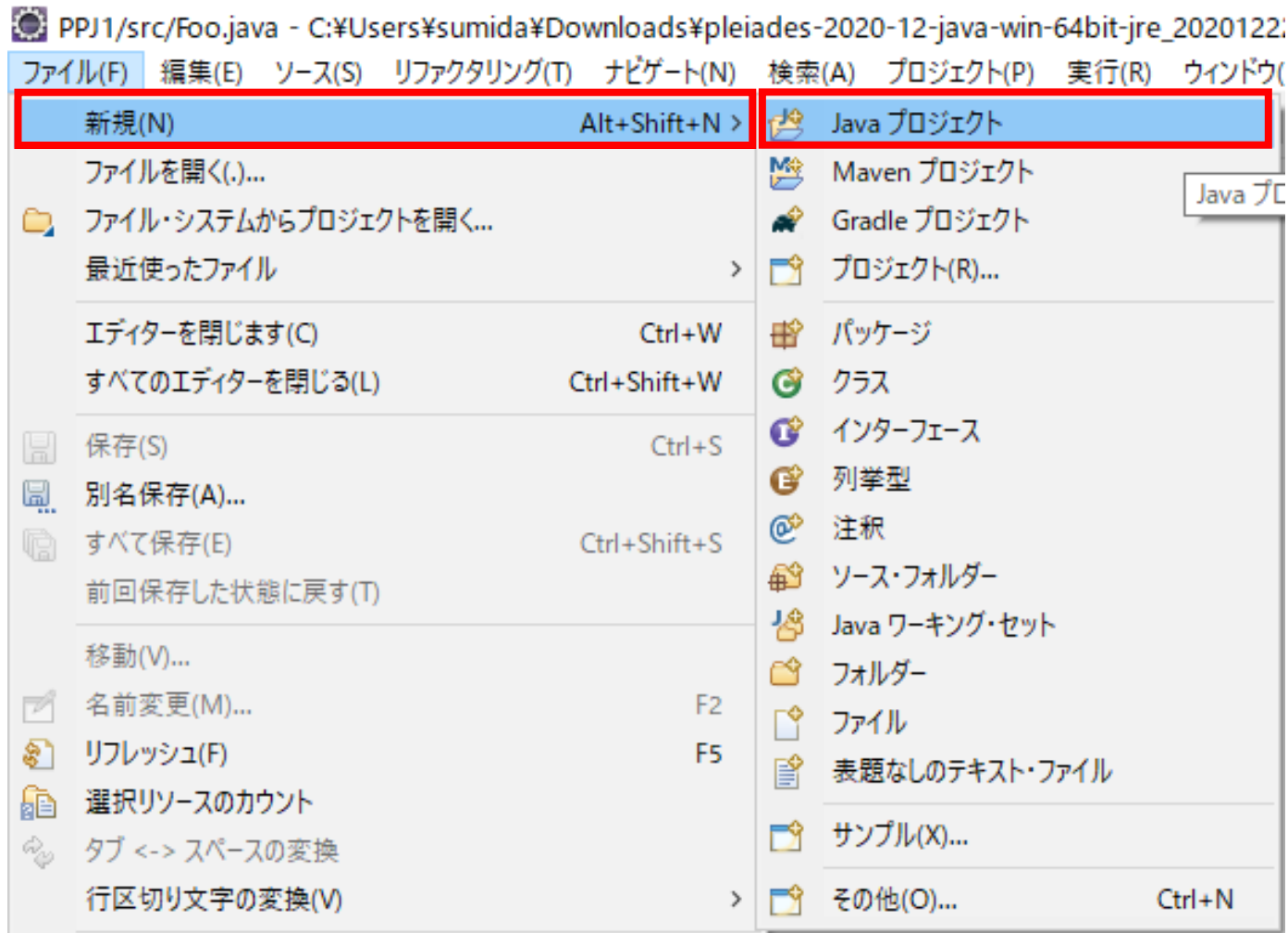
- テキスト(コード)を書くことに特化したアプリケーション
 - ソースコードの入力や編集は出来るが、
それ単体だとプログラムを動かすことが出来ない
 - 統合開発環境と比べると、動作が軽量（であることが多い）
- テキストエディタの例(Windows向け)
 - Visual Studio Code
 - メモ帳(Notepad)、Notepad++、Tera Pad、サクラエディタ、等

Visual Studio Codeについて

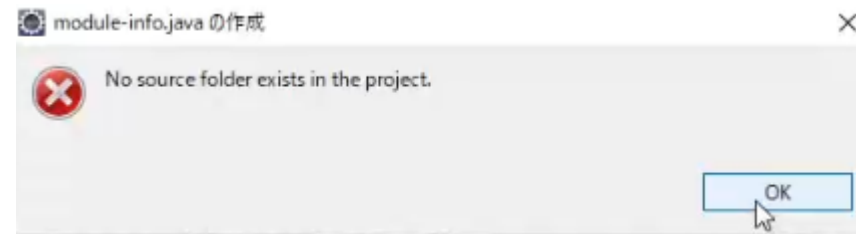
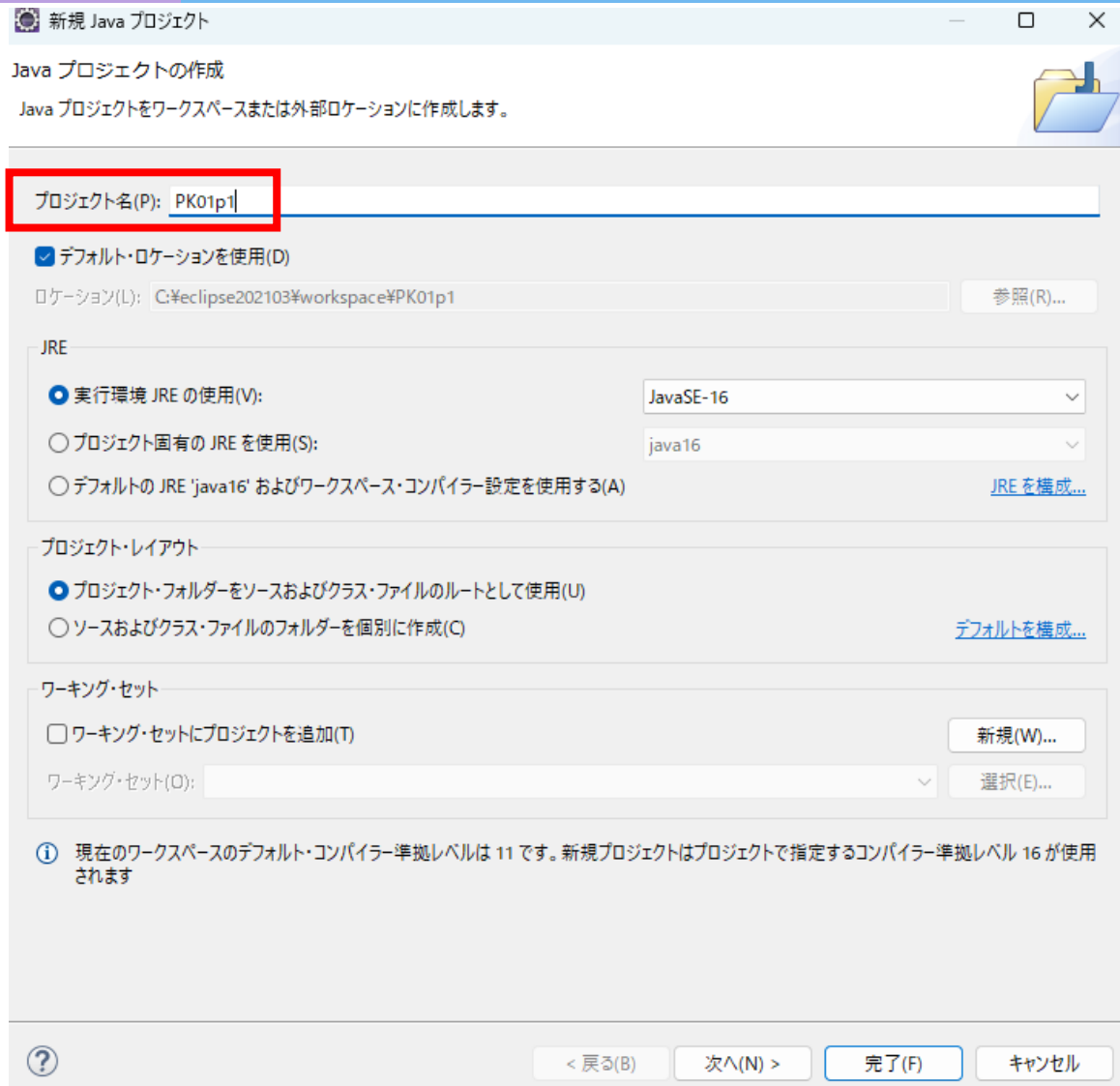
- Visual Studio Code (VS Code)はテキストエディタ
 - ただし、様々な拡張機能をインストールすることで、統合開発環境として動作するようになる
 - インストールしただけだと、プログラムを実行したり、デバッグすることは出来ない
 - その代わり、拡張機能を組み合わせることが出来る
 - 研究テーマに合わせてアレンジなど、自分好みの開発環境を作れる

Javaプロジェクトを作成

- 「ファイル」→「新規」→「Javaプロジェクト」



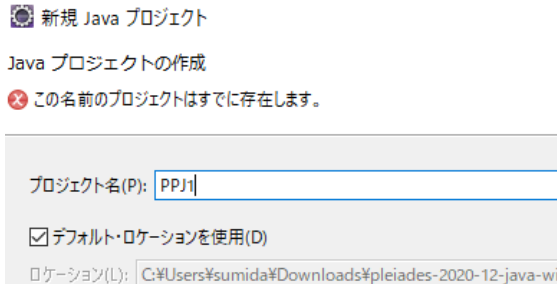
プロジェクト名を入力して、完了



このダイアログが出てきたら「OK」

こんなときは

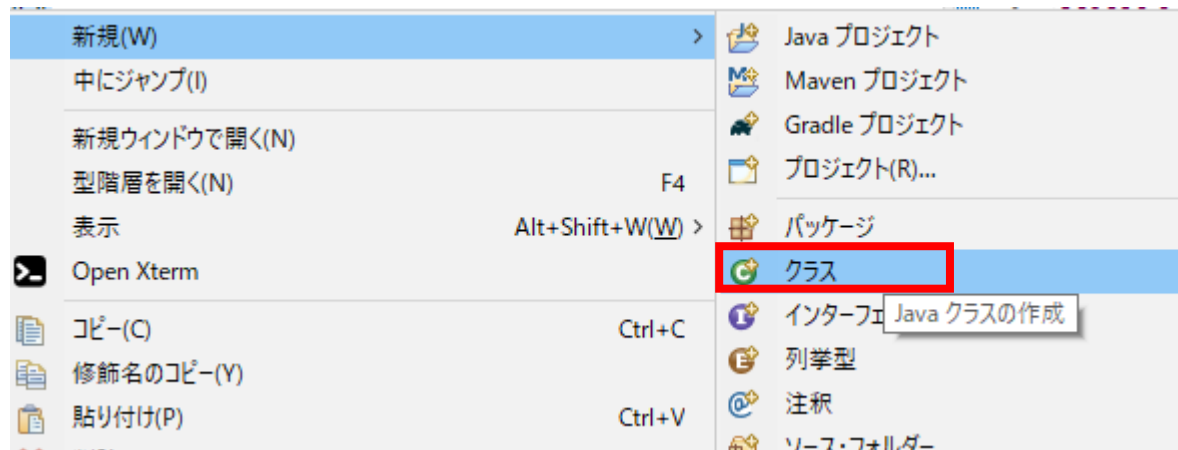
- 「この名前のプロジェクトはすでに存在します。」



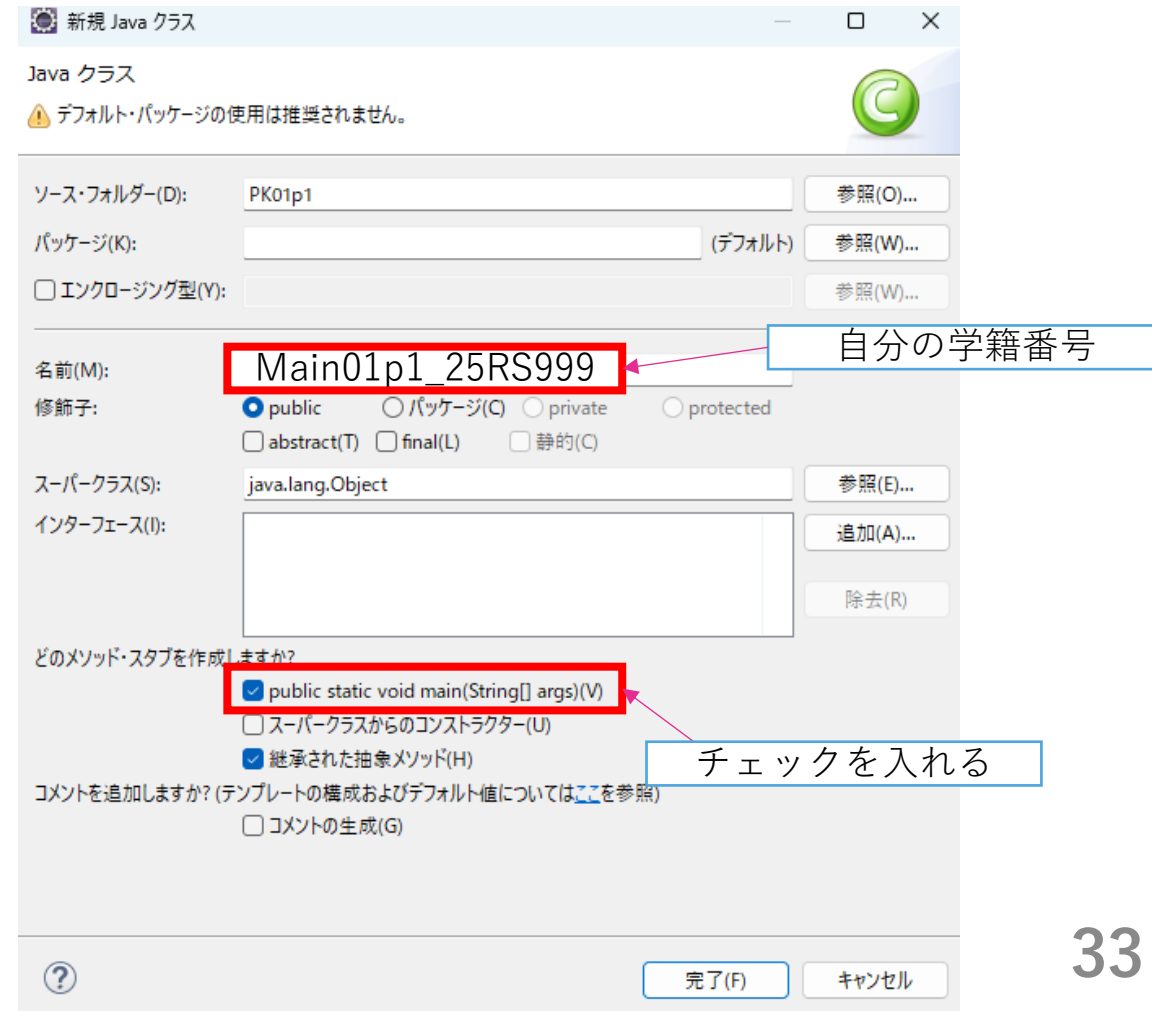
- 原因：一度、同じ名前のプロジェクトを作ったことがある
 - パッケージ・エクスプローラーに表示されていなくても、ワークスペースに同名名前のフォルダが存在すると、同名プロジェクトは作成できない
- 解決方法（授業中に質問すれば確実に）
 1. 元のフォルダを削除する（C:\eclipse202003\workspace）
 2. 後ろに番号を付けた別名プロジェクトを作成する
 1. PK01p1が作れない場合は、PK01p1_2 など

Javaプロジェクトに新規クラスを作成

- 「ファイル」→「新規」→「クラス」



Main01p1_学籍番号 (例:Main01p1_25RS999)



Eclipseでプログラムを記述

クラス名 (.java ファイル名と同じになる)

```
public class Main01p1_25RM999 {
```

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

mainメソッド
(必ず main)

ここに処理を記述

```
}
```

```
}
```

- JavaのプログラムはProcessingと異なり、すべてクラスの中に書く
- Javaのプログラムの本体は、mainメソッドの中に書く
- Eclipseを使うことで、このページに記述されているような内容は自動で生成されるので、手動で入力する必要はない

```
Test.java
1
2 public class Test {
3
4     public static void main(String[] args) {
5         // TODO 自動生成されたメソッド・スタブ
6         ← この行からプログラムを書いていく
7     }
8
9 }
10
```

プログラムはmainメソッドの中 `{ }` の中に書く
(第7回以降はmainメソッドの外に書く場合もあるが、
それまでは全てmainメソッド内に記述する)

キーボードから入力を受け取る（数値）

- Scannerを用いることで、キーボードから値を入力することができるようになる 教科書P.32 リスト3.16

```
import java.util.Scanner;
public class Main01p1_25RM999 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("円柱底面の半径を入力:");
        int radius = sc.nextInt();
        System.out.print("円柱の高さを入力:");
        int height = sc.nextInt();

    }
}
```

Scannerを使う準備 1

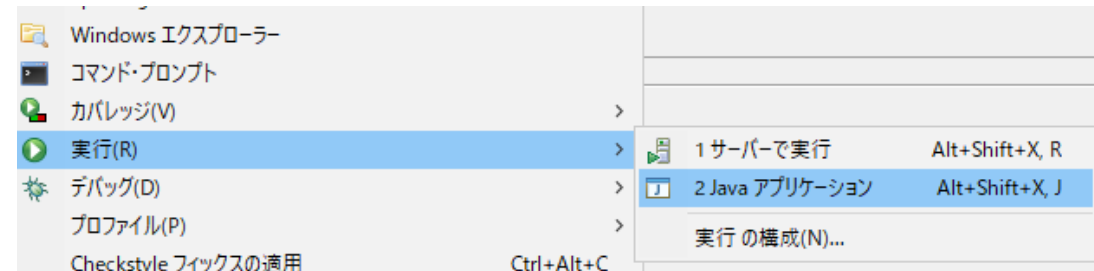
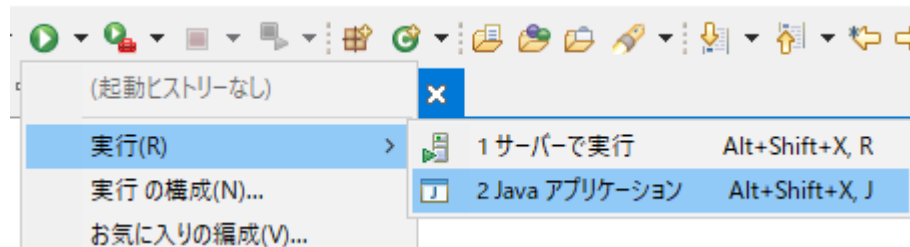
Scannerを使う準備 2

キーボード入力を受付
nextInt()は整数値を一つ受け取る
next()だと文字列を受け取る

Eclipseでプログラムを実行

- ソースコードエディターを右クリック
 - プログラムを書いている場所
- 「実行」→「Javaアプリケーション」 をクリック

- 実行ボタンから実行



- Alt + Shift + X → J キーでも実行可能

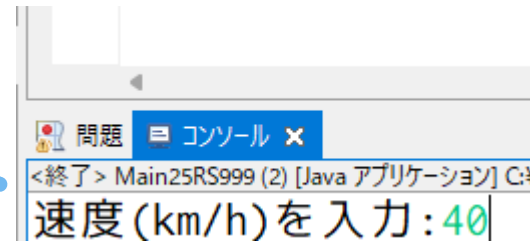
その他の実行方法

- 2回目以降は**Ctrl+F11**キーを押すと、
「**最後に実行したプログラム**」が実行される
 - 新しいプログラムを作成した場合は、前ページの方法などで一度実行
- その他、実行ボタンから実行、パッケージ・エクスプローラーのクラスファイルを右クリックして実行など・・・

Eclipseでのキーボード入力操作

- 「コンソール」をクリックしてから値を入力（半角全角に注意）

コンソール内をクリックしてから入力する



練習問題 1 : 変数、代入、初期化

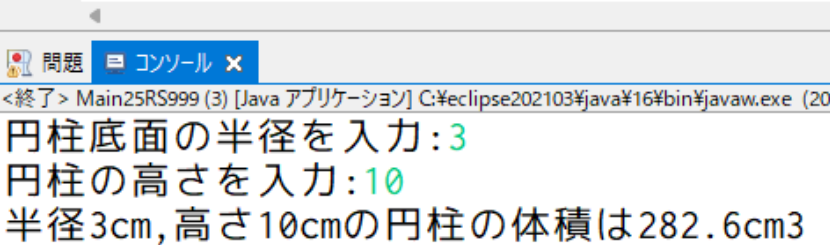
プロジェクト名:PK01p1 クラス名: Main01p1_学籍番号 (例:Main01p1_25RS999)

• 下記の問題のプログラムを記述

- 問題1:double型の変数 volume を宣言,円柱の体積を計算して代入
- 問題2:「半径3cm,高さ10cmの円柱の体積は282.6cm³」のように radius,height,volumeの値を表示

```
1 import java.util.Scanner;
2 public class Main25RS999 {
3
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.print("円柱底面の半径を入力:");
7         int radius = sc.nextInt();
8         System.out.print("円柱の高さを入力:");
9         int height = sc.nextInt();
10
11         // 問題1:double型の変数 volume を宣言,円柱の体積を計算して代入
12         [REDACTED]
13         // 問題2: 「半径3cm,高さ10cmの円柱の体積は282.6cm3」のように radius,height,volumeの値を表示
14         [REDACTED]
15     }
16 }
```

実行結果例



なるべく、自分で考えて問題を解こう (回答は動画を参照)

練習問題 1 ヒント

- 円柱の体積 = 底面積 * 高さ
 - 底面積 = 半径 * 半径 * 3.14
- `import java.util.Scanner;`
の記述忘れに注意

練習問題 1：入出力チェック

- 入力と出力が正しいことを確認する
 - 全パターン試して、同じ結果になることを確認しておくこと

入力 1	入力 2	出力
3	10	半径3cm,高さ10cmの円柱の体積は282.6cm ³
5	8	半径5cm,高さ8cmの円柱の体積は628.0cm ³
7	2	半径7cm,高さ2cmの円柱の体積は307.72cm ³

キーボードから入力を受け取る（文字列）

- 文字列を受け取る場合には、
nextInt()ではなく、next()メソッドを使用する

```
import java.util.*;

public class Sample0304_Input {
    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        Scanner sc = new Scanner(System.in);
        System.out.print("Please input a number: ");
        String s;
        s = sc.next();
        System.out.println("The string is : " + s);
    }
}
```

Scannerを使う準備 1

Scannerを使う準備 2

文字列を受け取る場合は
sc.next()

練習問題2：文字列演算

プロジェクト名:PK01p2 クラス名: Main01p2_学籍番号 (例:Main01p2_25RS999)

- 入力されたメールアドレスから、学籍番号だけを切り取って表示するプログラム

mainメソッド内のプログラムのみ

```
Scanner sc = new Scanner(System.in);
System.out.print("メールアドレスを入力:");
//問題1:String型の変数 inputMailAddress を宣言, キーボード入力から文字列を受け取って代入

//問題2:int型の変数 atmarkIndex を宣言し、inputMailAddress の @ の位置を取得して代入

//問題3:String型の変数 studentNumber を宣言し、
//      inputMailAddress の 2文字目から@の一つ前までの文字列を切り取って代入

//問題4:学籍番号は25rs999 のようにstudentNumberを表示
```

練習問題2 の注意と入出力チェック

- メールアドレスは全て半角英数で正確に入力すること
 - 余計なスペースなども含まないように注意する
- substringメソッドで文字列を切り抜く際の、インデックスの指定に注意
 - 1文字目は 0, 2文字目は 1 になる

入力	出力
k25rs999@st.kyusan-u.ac.jp	学籍番号は25rs999
k24re000@st.kyusan-u.ac.jp	学籍番号は24re000
k25gjk111@st.kyusan-u.ac.jp	学籍番号は25gjk111

まとめ

- 変数
 - 型 変数名; で変数宣言
- 代入
 - 変数名 = 値; で変数に値を格納
 - 型 変数名 = 値; で変数を宣言し、初期値を代入することができる
- 演算
 - +(加算) -(減算) *(乗算) /(除算) %(剰余)
 - 複合代入演算子、インクリメント演算子、デクリメント演算子
- 文字列演算
 - String.length、String.indexOf、String.substring
 - 文字列長の取得、文字列内検索、文字列切り取り