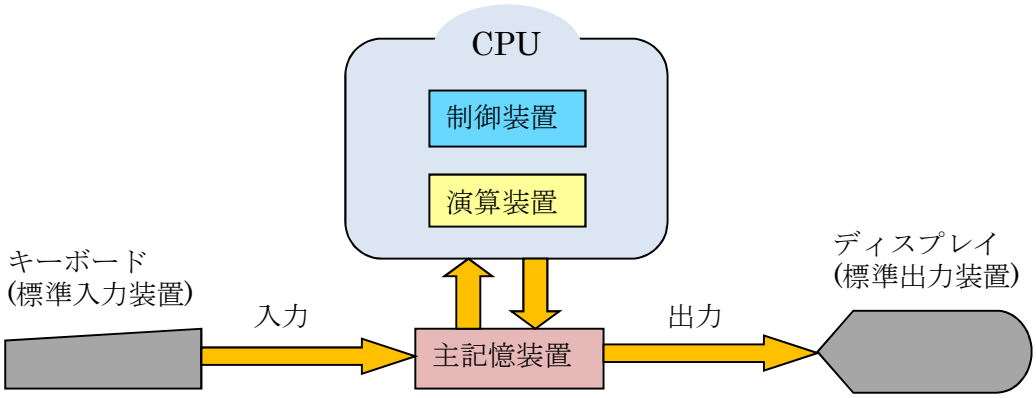
2018 年 4 月 12 日（木）実施

コンピュータのハードウェアとプログラム

機械（ハードウェア）としてのコンピュータの特徴

現在，普通に使われているコンピュータは**ディジタルコンピュータ**（2 値論理に基づくコンピュータ）であり，次の様な特徴を備えている。

1. 電子デバイス（主に半導体集積回路を用いた装置）により，データ処理を行う。
2. 2 値論理に基づいてデータを扱う。（電圧の高低，電荷や窪みの有無，磁化の向き等）
3. 主記憶装置（メインメモリ）に一時的に読み込んだプログラムを中央処理装置（CPU）で解釈・実行することにより，データ処理を行う。　【プログラム記憶方式】



注）ディスプレイには，ビデオ RAM（VRAM）に書き込まれたデータが表示される。

プログラムとは（機械語，アセンブラ，高級言語）

プログラムとは，**コンピュータに処理の手順を指令するもの**である。コンピュータが動作するための命令の種類とその書式は，CPU の設計段階で決定される命令セット（Instruction Set Architecture）に基づく。この命令セットを用いて作成されたプログラムを**機械語プログラム**と呼び，命令及びデータは 2 進数で表される。コンピュータが動作可能なプログラムは，機械語で記述されたもののみである。なお，命令セットは CPU ごとに異なるので，ある CPU 用の機械語プログラムは異なる設計の CPU 用には流用できない。

また，2 進数でプログラムを作成していくのは困難なので，命令に名前付けをした，**アセンブラ**（アセンブリ言語）が用いられる。アセンブラプログラムは機械語プログラムと 1 対 1 に対応するので，やはり CPU の設計に依存する。

通常のプログラミング（プログラム作成）には，CPU の設計に依存しない，**高級言語**と呼ばれるプログラム言語を用いることが多い。高級言語の多くは命令を英単語やその組み合わせまたはその省略形で表す人工的な言語である。高級言語で記述されたプログラムの各行は機械語に 1 対 1 には対応していないので，翻訳（コンパイル）・編集（リンク）という作業を通じて，コンピュータが実行可能な機械語プログラムに変換される。この作業は大変複雑なため，それぞれ**コンパ**

イラ，**リンカ**というプログラムを通じて行われる。この場合，元の高級言語プログラムは**ソースプログラム**と呼ばれる。（ソースプログラムの各行を解釈・実行していく**インタプリタ**と呼ばれるプログラムも存在する）

この授業では，高級言語としては最近よく用いられている，**Java 言語**のプログラミングについて初歩から学ぶ。

Java 言語の概要

Java 言語の特徴

Java 言語は，多くのアプリケーション開発に用いられており，次の様な特徴を備えている。

1. **オブジェクト指向**言語である。

オブジェクト： データと手続きとをひとまとめにした対象を指す。
オブジェクト指向： プログラムをオブジェクト間の相互作用により構成する。

2. 可搬性が高い。（機種による依存性が低い）

Java 言語で記述されたプログラムを実行する際には，先ず，ソースファイル（ソースコード，拡張子**.java**）をコンパイルしてクラスファイル（バイトコード，拡張子**.class**）という中間段階のコードを作成する。Java 実行環境では，プログラム中で利用されるクラスライブラリを備え，クラスファイル+クラスライブラリを Java 仮想マシンというインタプリタで解釈して実行する。

3. 開発支援環境が無償で提供されている。

1) Java 実行環境： JRE (Java Runtime Environment)　　Java 仮想マシンを含む。
2) ソフトウェア開発キット： SDK (Software Development Kit)　コンパイラ，JRE を含む。
3) 統合開発環境： IDE (Integrated Development Environment)　エディタ，SDK を含む。

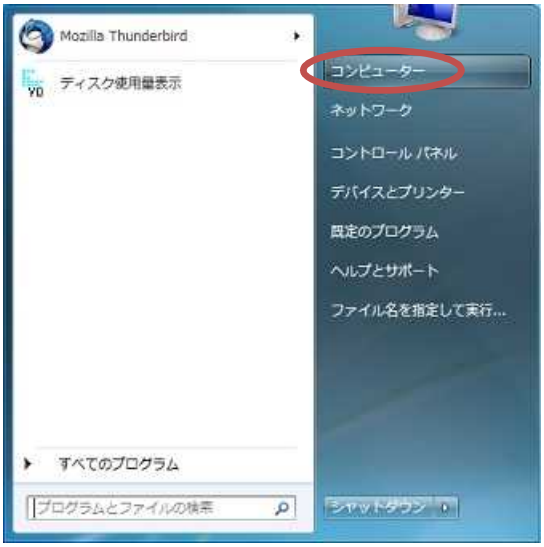
＊ この授業では，IDE として Eclipse を用いる。Eclipse は実際のソフトウェア開発現場でよく利用されている。

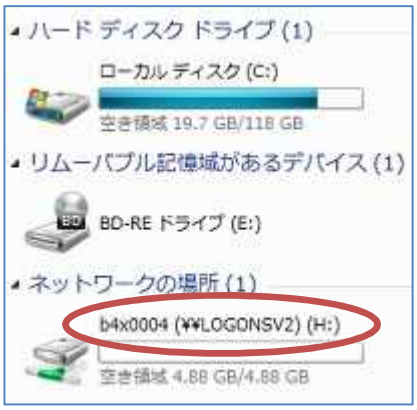
授業の準備

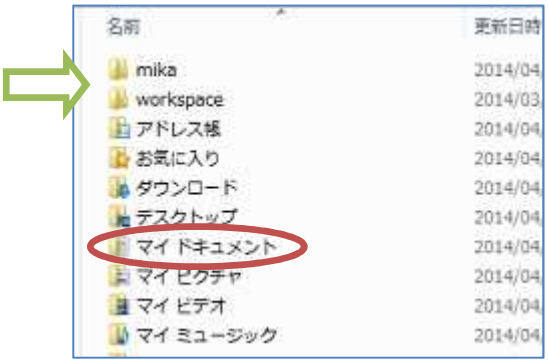
1) 教材ダウンロード用のフォルダ作成

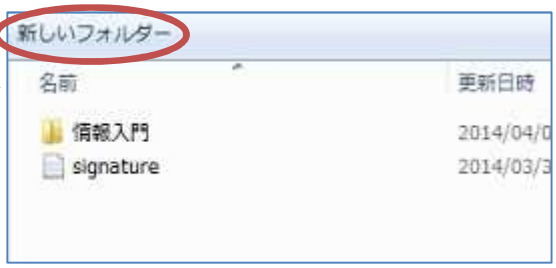
情報入門で学んだ様に，プログラミング１の教材ダウンロード用として，H ドライブのマイドキュメントの中に『Prog1』という名前のフォルダを作成する。

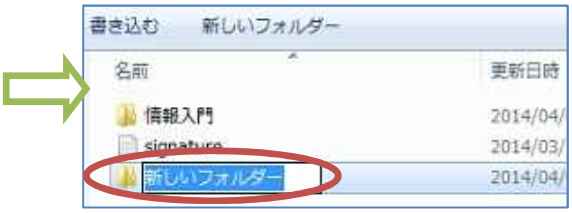
次に，『神保雅人 教材集』（http://www.cuc.ac.jp/~jimbo/textbooks.html）から教材をダウンロードする。

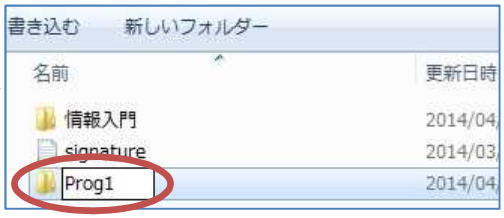


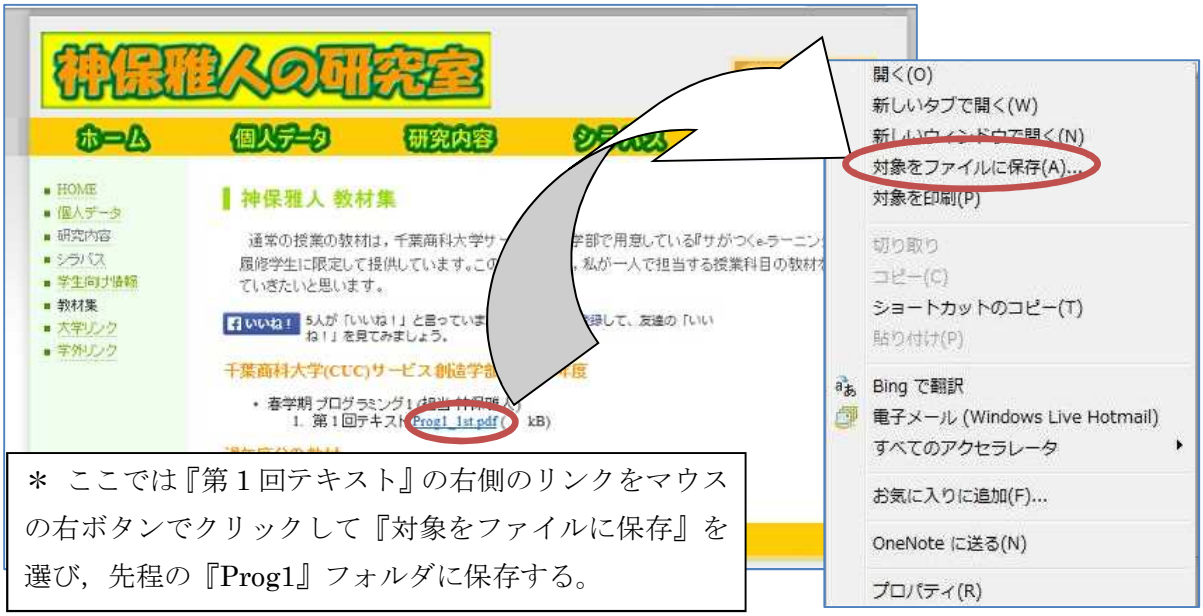










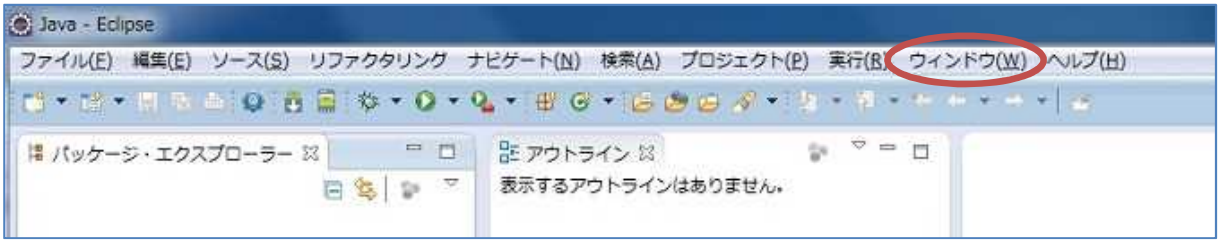


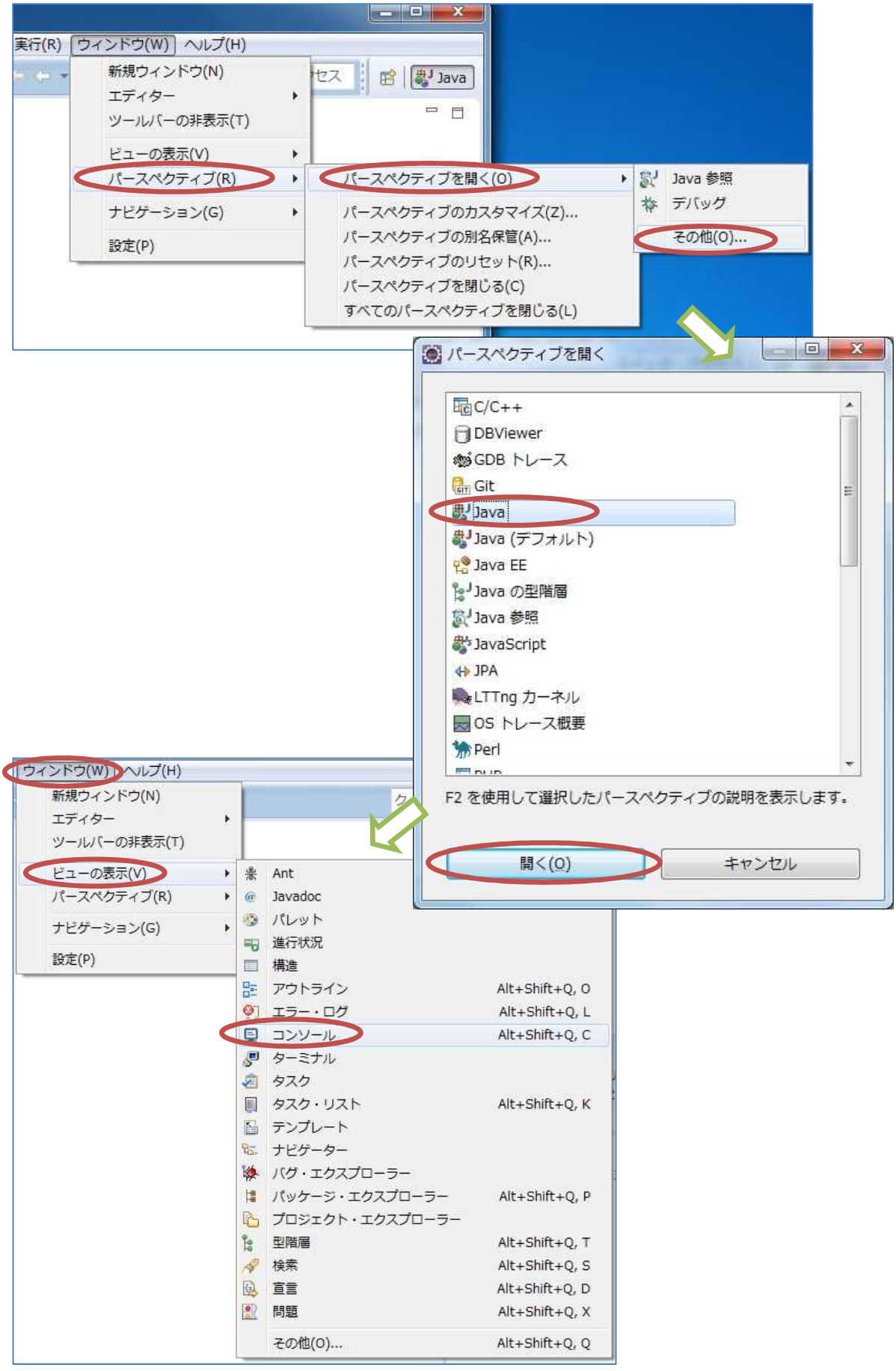
＊ ここでは『第１回テキスト』の右側のリンクをマウスの右ボタンでクリックして『対象をファイルに保存』を選び，先程の『Prog1』フォルダに保存する。

2) Eclipse の初期設定

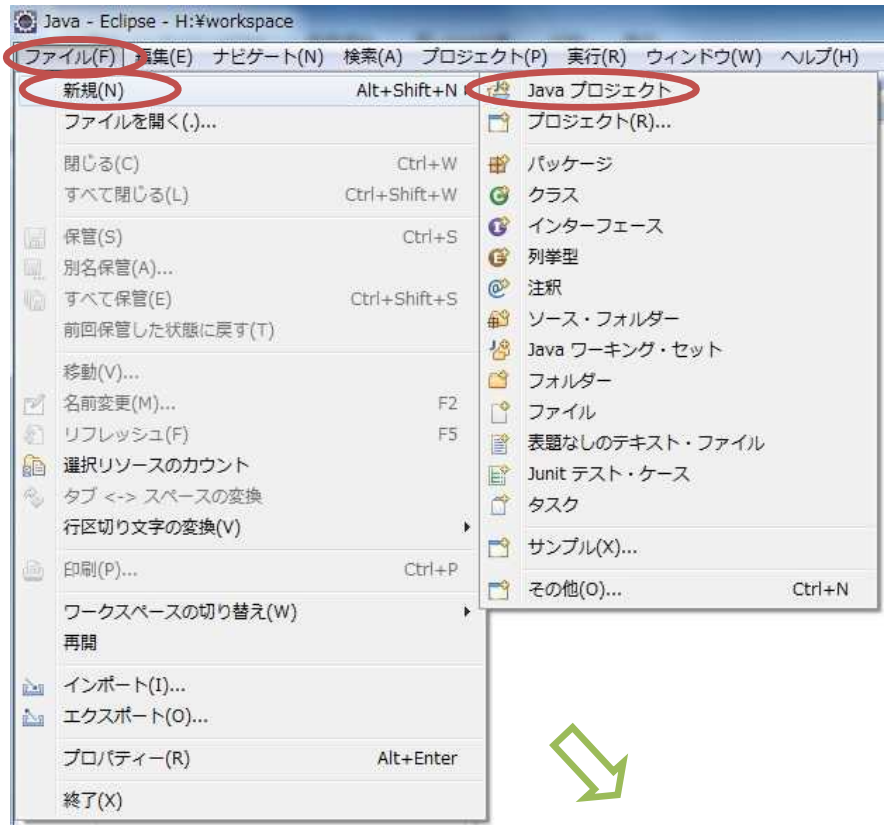
［スタートボタン］→［すべてのプログラム］→［eclipse］と辿って，Eclipse を起動する。

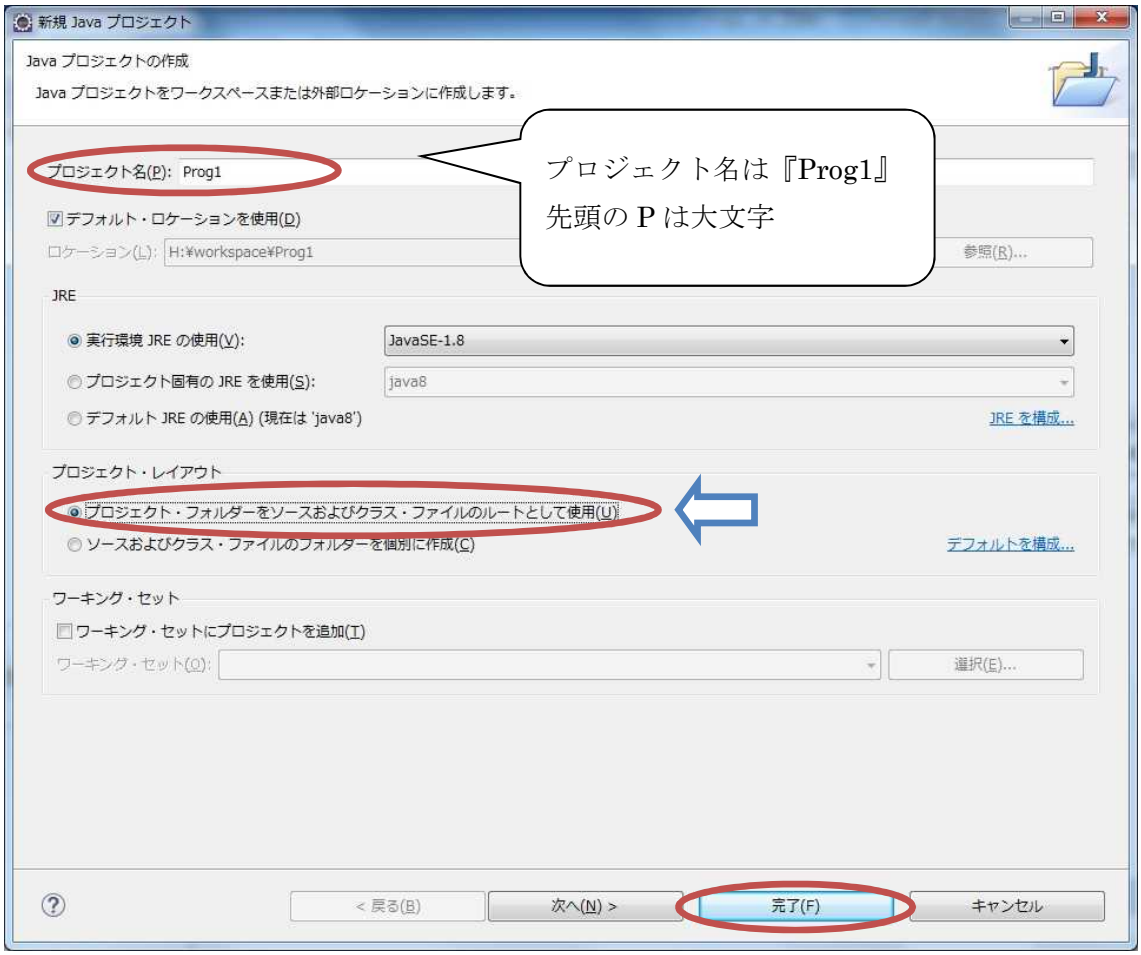
起動が完了したら，Eclipse の外観に関して，初期設定を開始する。




実行(R)
ウィンドウ(W)
ヘルプ(H)
新規ウィンドウ(N)
エディター
ツールバーの非表示(T)
ビューの表示(V)
パースペクティブ(R)
ナビゲーション(G)
設定(P)
パースペクティブを開く(O)
パースペクティブのカスタマイズ(Z)...
パースペクティブの別名保管(A)...
パースペクティブのリセット(R)...
パースペクティブを閉じる(C)
すべてのパースペクティブを閉じる(L)
Java 参照
デバッグ
その他(O)...
Java
パースペクティブを開く
C/C++
DBViewer
GDB トレース
Git
Java
Java (デフォルト)
Java EE
Java の型階層
Java 参照
JavaScript
JPA
LTTng カーネル
OS トレース概要
Perl
F2 を使用して選択したパースペクティブの説明を表示します。
開く(O)
キャンセル
ウィンドウ(W)
ヘルプ(H)
新規ウィンドウ(N)
エディター
ツールバーの非表示(T)
ビューの表示(V)
パースペクティブ(R)
ナビゲーション(G)
設定(P)
Ant
Javadoc
パレット
進行状況
構造
アウトライン Alt+Shift+Q, O
エラー・ログ Alt+Shift+Q, L
コンソール Alt+Shift+Q, C
ターミナル
タスク
タスク・リスト Alt+Shift+Q, K
テンプレート
ナビゲーター
バグ・エクスプローラー
パッケージ・エクスプローラー Alt+Shift+Q, P
プロジェクト・エクスプローラー
型階層 Alt+Shift+Q, T
検索 Alt+Shift+Q, S
宣言 Alt+Shift+Q, D
問題 Alt+Shift+Q, X
その他(O)... Alt+Shift+Q, Q

次に授業用のプロジェクトを作成しておく。





Java 言語プログラムの構成

Java 言語ではプログラムの基本は**クラス**と呼ばれる単位である。クラスの定義は，一般に次の形式となる。

```
修飾子 class クラス名 {
  型 フィールド名;
  修飾子 戻り値のデータ型 メソッド名( 引数 ){
    メソッドの定義
  }
}
```

この中で，**メソッド**に手続きを記述する。特に，プログラム動作時に最初に呼び出されるクラスには，**main メソッド**が必須である。

標準出力装置への文字列の表示

Java 言語で標準出力装置（p. 1 参照）に文字列（複数の文字の並び）を出力するには，元々用意されている println 等のメソッドを利用する。

コンピュータ内部では，文字には**文字コード**と呼ばれる番号が付けられていて，その番号によって扱われる。画面に文字を出す場合には，個々の文字に対して複数の文字の形（**フォント**）が点の集まりとして定義されていて，その中の一つのフォントのデータをビデオ RAM に書き込むと表示される仕組みとなっている。

『ペイント』で『あ』を描いて拡大したもの

例題１　**（標準出力装置への文字列の表示）**

次のプログラムは標準出力装置に文字列を表示させるものである。これを入力して，実行せよ。ここで，クラス名は Sample1_1，ソースファイル名は Sample1_1. java とする。なお，〇〇には自分の名前を入れる。Eclipse では**ビルド**（高級言語のプログラムを翻訳して実行出来る様にすること）から実行までは，ボタン操作一つで出来る。

```
public class Sample1_1 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        System.out.println("今日は、サービス創造学部の〇〇です。");
    }

}
```

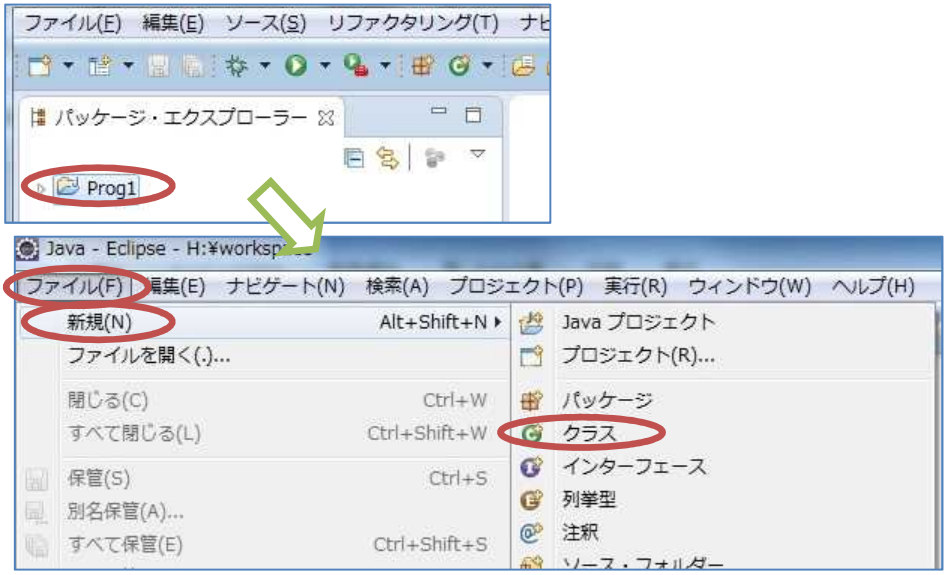
この行を入力

【解説】

1. //から右は注釈であり，プログラムの翻訳の際には無視され，動作に影響しない。注釈が何行に亘ってもよい。(/* から始めて */ で終わる注釈も使える）ここでは Eclipse が自動的に生成している注釈をそのまま表示している。
2. クラス名は大文字から始める習慣となっている。ここでは Sample1_1 がクラス名である。
3. Sample1_1 クラスにはフィールドはなく，main メソッドのみから成る。
4. 修飾子 public は，そのクラスやメソッドが利用可能な範囲を指定する**アクセス修飾子**の一つで，全てのクラスからの利用が可能であることを表す。
5. 修飾子 static は，そのメソッドがクラスから直接呼び出して使える**クラスメソッド**であることを表す。
6. void は戻り値のないメソッドであることを表す。
7. String[] args は，プログラム引数を文字列として格納する配列 args を用意する。
8. System.out.println はクラス名.フィールド名.メソッド名の構成となっていて，括弧内に与えた値を標準出力装置に出力するメソッドである。（Eclipse 内で実行する場合には，そのコンソールに出力する）なお，System クラスはパッケージ java.lang の中に用意されている。(out は PrintStream クラスのフィールドであり，println は PrintStream クラスのメソッドであるが，ここでは詳述しない)

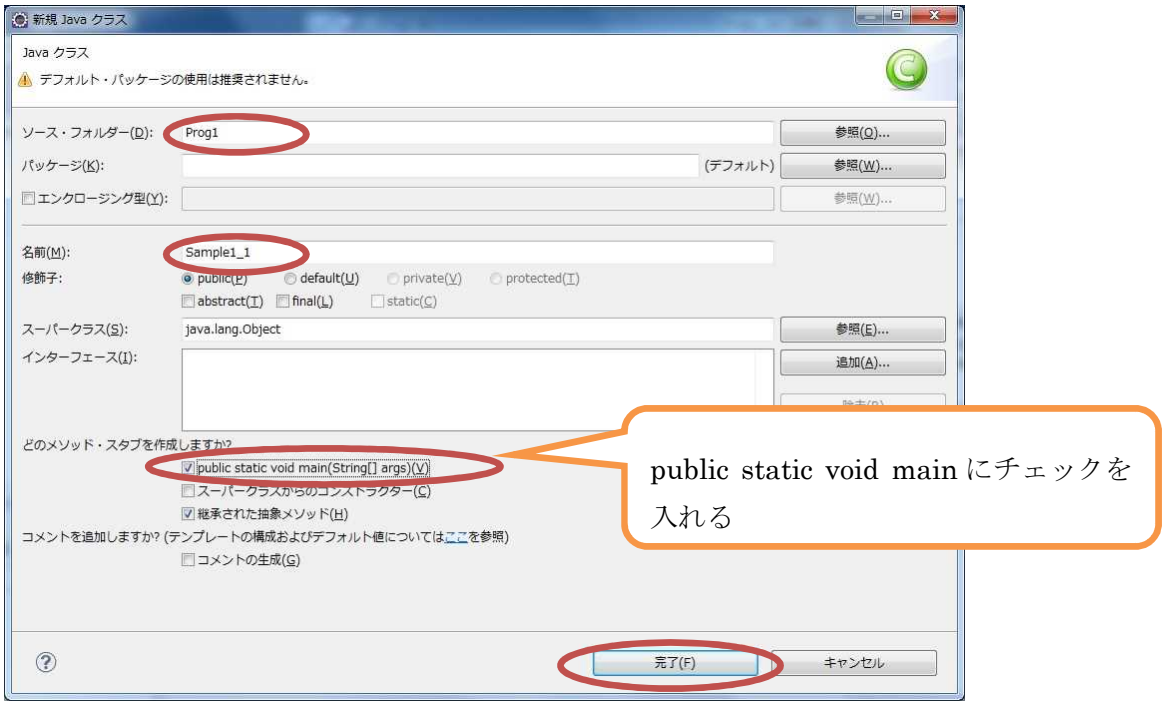
プログラムの作成から実行まで

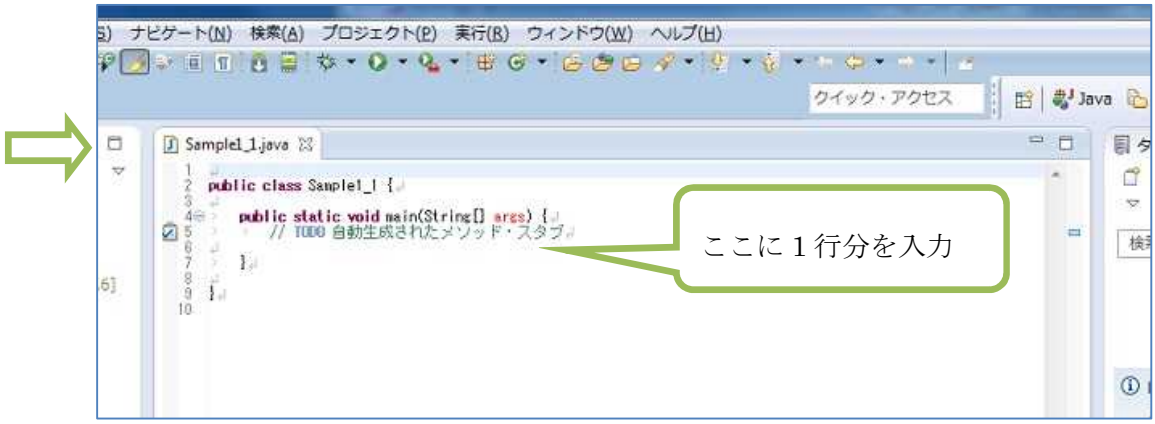
Eclipse の『パッケージ・エクスプローラ』に先程作成したプロジェクト『Prog1』が表示されている箇所をクリックして選択しておき，［ファイル］→［新規］→［クラス］の順で選択する。



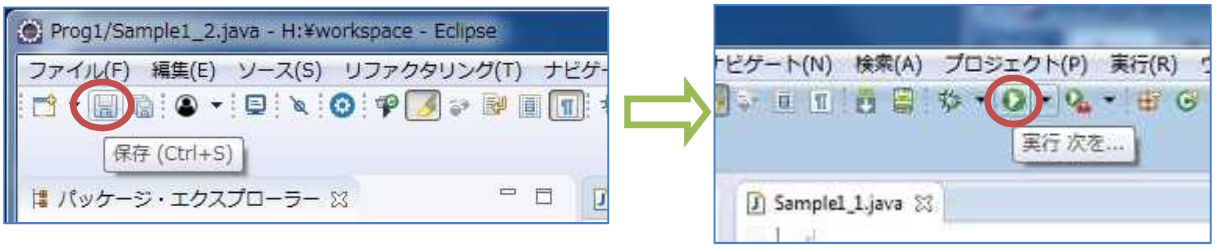
次ページの図の様なダイアログが出てきたら，

・『ソース・フォルダー』欄に『Prog1』が表示されていることを確認する。
・『名前』欄に『Sample1_1』を入力する。（この箇所はプログラム毎に別の名前となる）
・『public static void main(String[] args)』欄をクリックして，チェック（✓）を入れる。
・『完了』ボタンをクリックする。

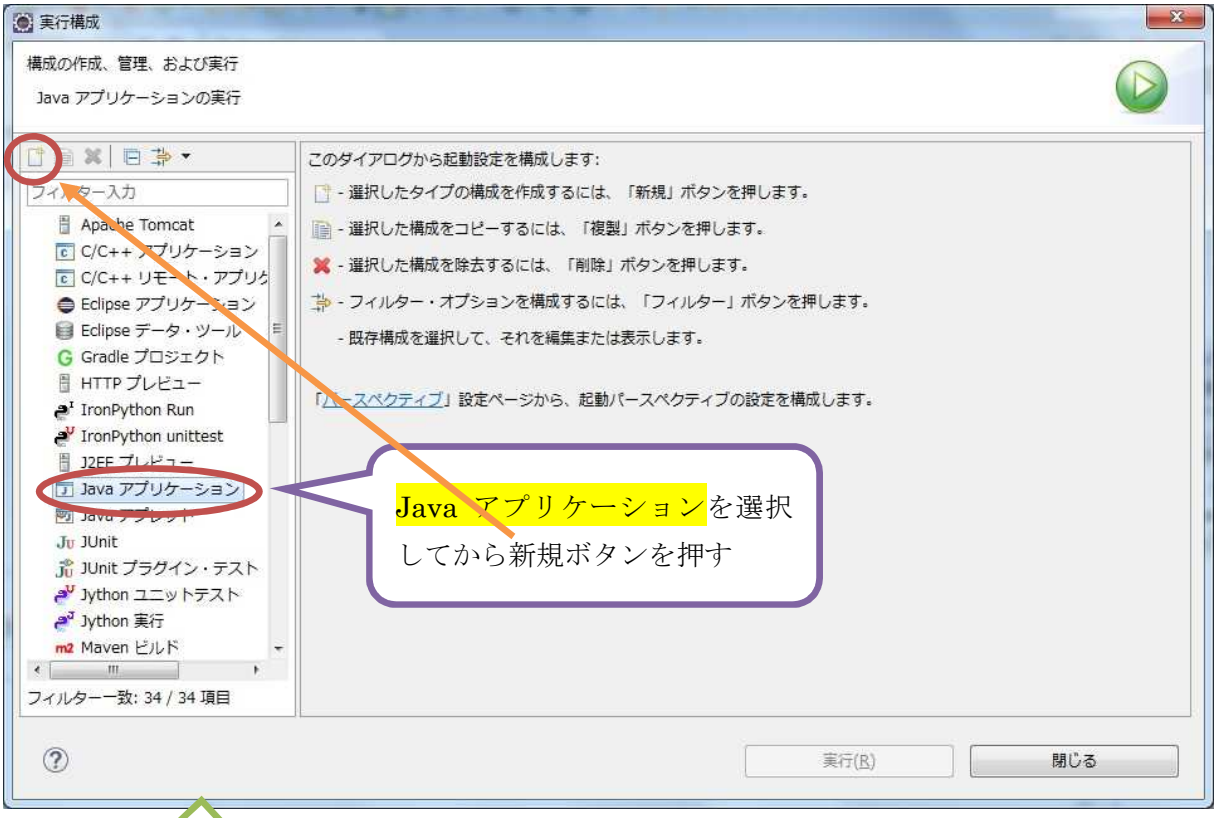

新規 Java クラス
Java クラス
デフォルト・パッケージの使用は推奨されません。
ソース・フォルダー(D): Prog1
パッケージ(K):
エンクロージング型(Y):
名前(M): Sample1_1
修飾子: public(P) default(U) private(V) protected(T)
abstract(T) final(L) static(C)
スーパークラス(S): java.lang.Object
インターフェース(I):
どのメソッド・スタブを作成しますか?
public static void main(String[] args)(V)
スーパークラスからのコンストラクター(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)
参照(O)...
参照(W)...
参照(E)...
追加(A)...
完了(F)
キャンセル
public static void main にチェックを入れる

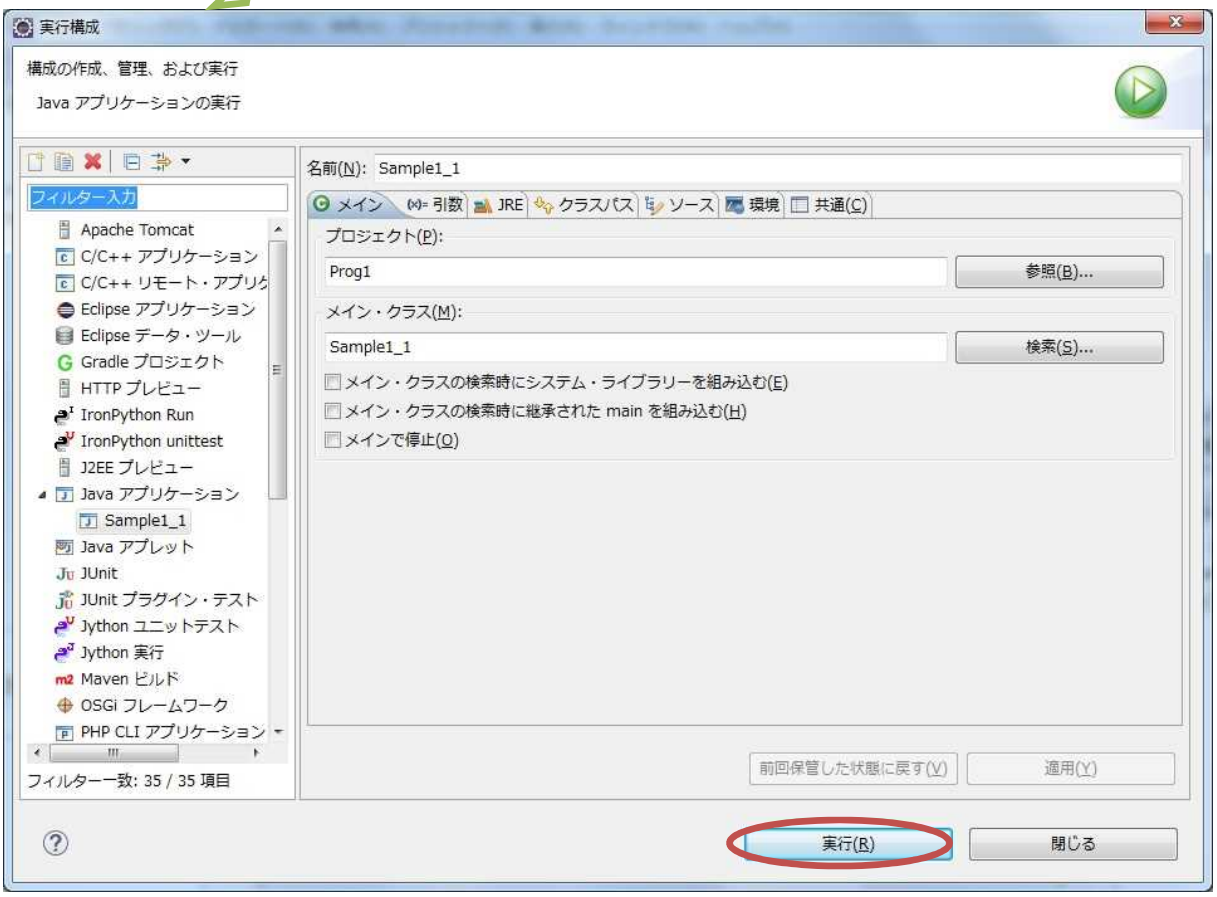

ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
クイック・アクセス
Java
Sample1_1.java
public class Sample1_1 {
public static void main(String[] args) {
// TODO 自動生成されたメソッド・スタブ
}
}
ここに１行分を入力

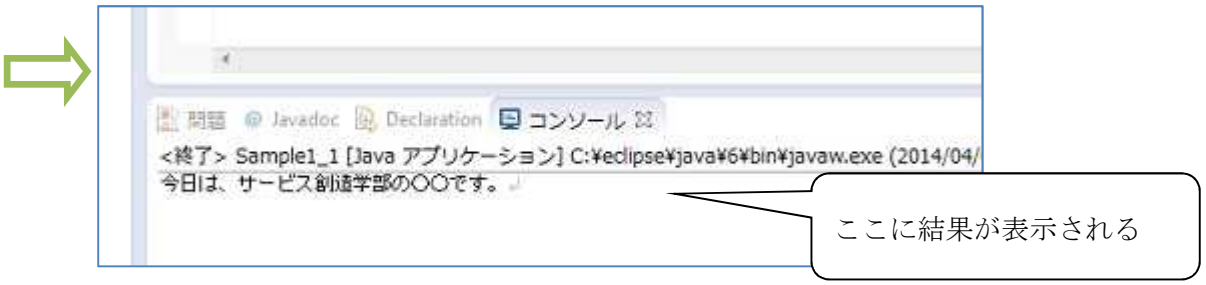

*Sample1_1.java
public class Sample1_1 {
public static void main(String[] args) {
// TODO 自動生成されたメソッド・スタブ
System.out.println("今日は、サービス創造学部の○○です。");
}
}


Prog1/Sample1_2.java - H:¥workspace - 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
保存 (Ctrl+S)
パッケージ・エクスプローラー
ナビゲート(N) 検索(A) プロジェクト(P) 実行(R)
実行 次を...
Sample1_1.java

【初回の対応】

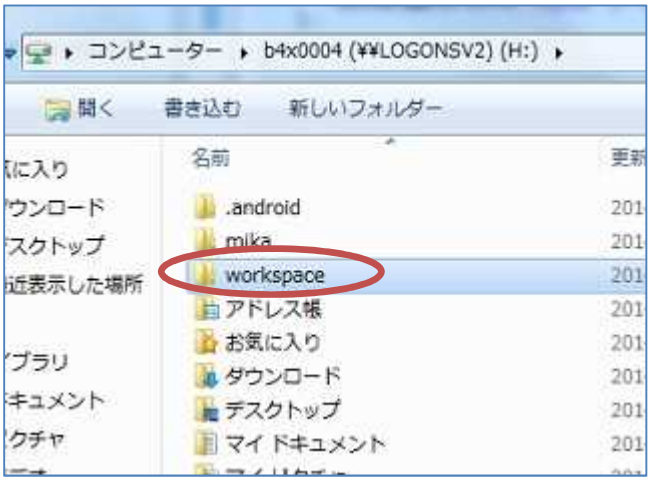

実行構成
構成の作成、管理、および実行
Java アプリケーションの実行
このダイアログから起動設定を構成します:
- 選択したタイプの構成を作成するには、「新規」ボタンを押します。
- 選択した構成をコピーするには、「複製」ボタンを押します。
- 選択した構成を除去するには、「削除」ボタンを押します。
- フィルター・オプションを構成するには、「フィルター」ボタンを押します。
- 既存構成を選択して、それを編集または表示します。
「パースペクティブ」設定ページから、起動パースペクティブの設定を構成します。
Java アプリケーション
JUnit
JUnit プラグイン・テスト
Jython ユニットテスト
Jython 実行
Maven ビルド
フィルター一致: 34 / 34 項目
実行(R)
閉じる
Java アプリケーションを選択してから新規ボタンを押す

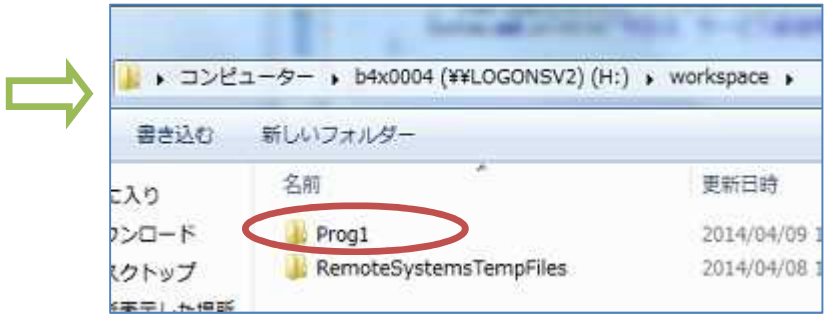

実行構成
構成の作成、管理、および実行
Java アプリケーションの実行
フィルター入力
Java アプリケーション
Sample1_1
Java アプレット
JUnit
JUnit プラグイン・テスト
Jython ユニットテスト
Jython 実行
Maven ビルド
OSGi フレームワーク
PHP CLI アプリケーション
フィルター一致: 35 / 35 項目
名前(N): Sample1_1
メイン
引数
JRE
クラスパス
ソース
環境
共通(C)
プロジェクト(P):
Prog1
参照(B)...
メイン・クラス(M):
Sample1_1
検索(S)...
メイン・クラスの検索時にシステム・ライブラリーを組み込む(E)
メイン・クラスの検索時に継承された main を組み込む(H)
メインで停止(O)
前回保管した状態に戻す(V)
適用(Y)
実行(R)
閉じる

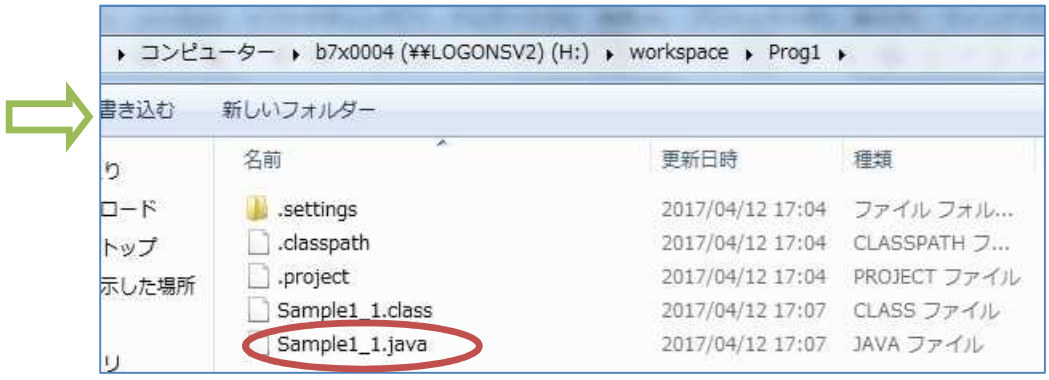


プログラムの保存場所

作成したプログラムは H ドライブの『workspace』フォルダの中に出来た，『Prog1』フォルダに保存されている。







＊ エディタで作成して保存したファイル（ソースファイル）名は Sample1_1.java となる。

変数と型

変数

プログラム中の命令を通じて，メインメモリ上にデータを格納する領域を確保し，必要に応じてその場所に格納されるデータを上書きすることが可能である。このような領域を**変数**という。Java 言語では，変数の取り扱いに関して，次の様な特徴がある。

1. プログラム中で用いる変数は必ず**宣言**しておく。 ⇒ メインメモリ上にデータを格納する領域を確保せよ，という命令に相当する。

2. 変数の宣言時には，**データ型**を指定する。 ⇒ データ型毎にデータを格納する領域のサイズが固定されている。

   例） int x;　// 整数（integer）のデータ型の変数 x を宣言

3. 変数にデータを格納するには，**代入**を行う。

   例） x=10;　/* 変数 x に定数 10 を代入（=の右辺を評価し，その値を左辺の変数に格納）
   なお，宣言時に**初期化**することも可能である。 ⇒ int x=4; */

4. 代入式の左辺以外に変数名を用いると**参照**が行われ，変数に格納されたデータが取り出されて利用される。

   例） System.out.println(x);　// 変数 x の中身を画面に表示

データ型

Java 言語には数値や文字のデータを扱う**基本型(プリミティブ型)**と，クラス等を扱う**参照型**とがある。基本型には，次のものがある。

分類	型名	サイズ	内容
整数型	char	16 ビット	文字型とも呼ばれる　Unicode 文字（\u0000 ～ \uffff）
	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（-2^{31} ～ $2^{31}-1$ = -2147483648 ～ 2147483647）
	long	64 ビット	整数（-2^{63} ～ $2^{63}-1$）
実数型	float	32 ビット	単精度浮動小数点数
	double	64 ビット	倍精度浮動小数点数
論理型	boolean	1 ビット	論理値（true または false）

算術演算

Java 言語で，2 個の変数 a，b の間で加減乗除の演算を行うには，a+b，a−b，a*b，a/b の様に，加減演算子（+，−），乗除演算子（*，/）を用いる。また，乗除演算子には a を b で割った余りを求める際に用いられる『%』もある。

なお，整数を整数で割った場合には，小数点以下は切り捨てられ，結果は整数部のみ残る。

例題 2　（**標準出力装置への計算結果の表示**）

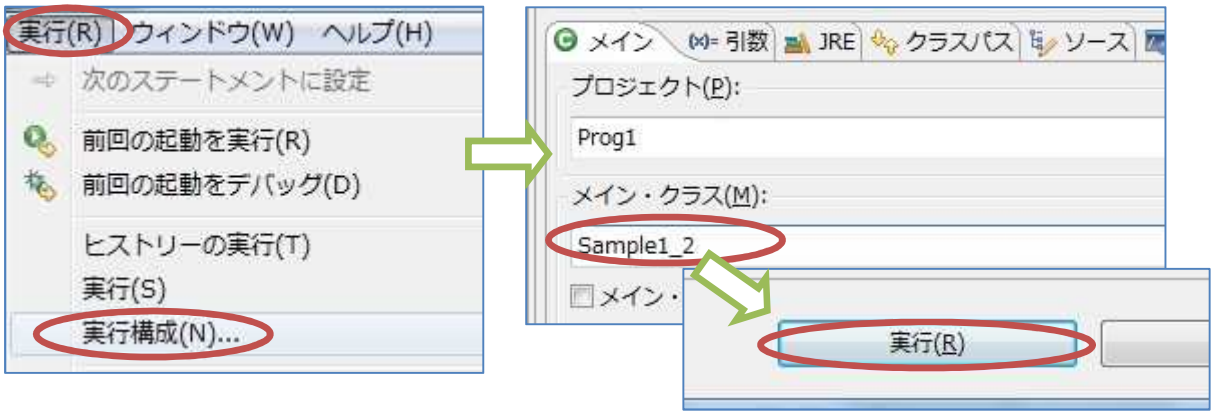
次のプログラムは標準出力装置に計算結果を表示させるものである。これを入力して，実行せよ。ここで，クラス名は Sample1_2，ソースファイル名は Sample1_2.java とする。

```
public class Sample1_2 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        int wa, sa, seki, shou, amari;
        int a=4, b=10;

        wa=a+b;
        sa=a-b;
        seki=a*b;
        shou=a/b;
        amari=a%b;

        System.out.println("a と b との和は" + wa);
        System.out.println("a と b との差は" + sa);
        System.out.println("a と b との積は" + seki);
        System.out.println("a と b との商は" + shou);
        System.out.println("a を b で割った余りは" + amari);
    }
}
```

【解説】 System.out.println の括弧の中で，二重引用符で挟まれている文字列（"a と b との和は" 等）と変数名（wa 等）との間にある『+』は文字列を連結する演算子として扱われる。

＊ プログラムごとにクラスを新規作成する（pp. 8-9 の手続きが必要）。

＊＊ メイン・クラスの欄を書き直して『実行』ボタンを押す。



演習

例題２のプログラムを実行すると，「a と b との商は 0」と表示されるが，本日学んだ内容から，その理由を考察せよ。

提出物：

1）例題２のプログラムのコンソールへの出力結果をコピーしてメール本文に貼り付ける。
2）演習で考察した内容をメール本文に記述する。
＊ メールの件名は『プログラミング １　第１回課題』（鍵括弧は要らない）とし，宛先メールアドレスは授業内で伝える。

コンソールからの出力結果の取り出し

コンソールに表示されている文字列は，マウスで**ドラッグ**（左ボタンを押したままマウスを動かすこと）して範囲選択し，これをマウスの右ボタンでクリック（以下，右クリックと表記する）して『コピー』を選択することにより，**クリップボード**と呼ばれるメインメモリ上の領域にコピーすることが出来る。メールの本文で右クリックして『貼り付け』を選択すると，**クリップボードの内容が貼り付けられる。**