

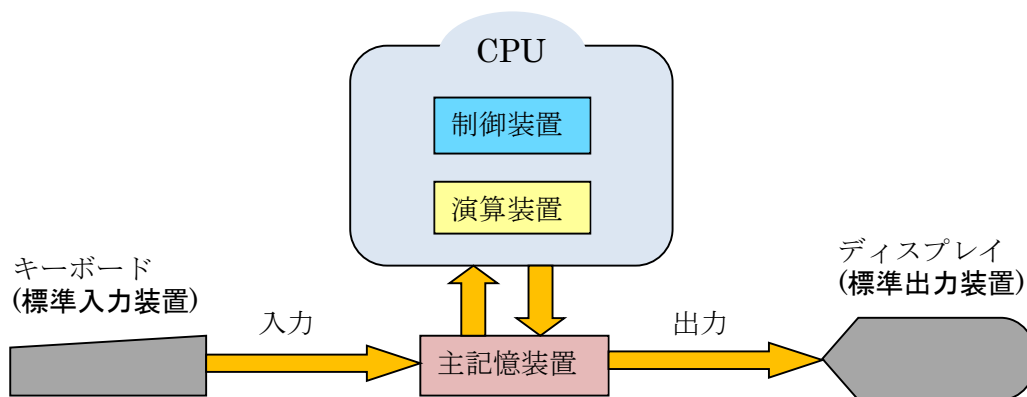
2023 年 10 月 12 日 (木) 実施

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順序を指令するもの**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 用には流用出来ない。

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、まず、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、プロクラスファイル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。

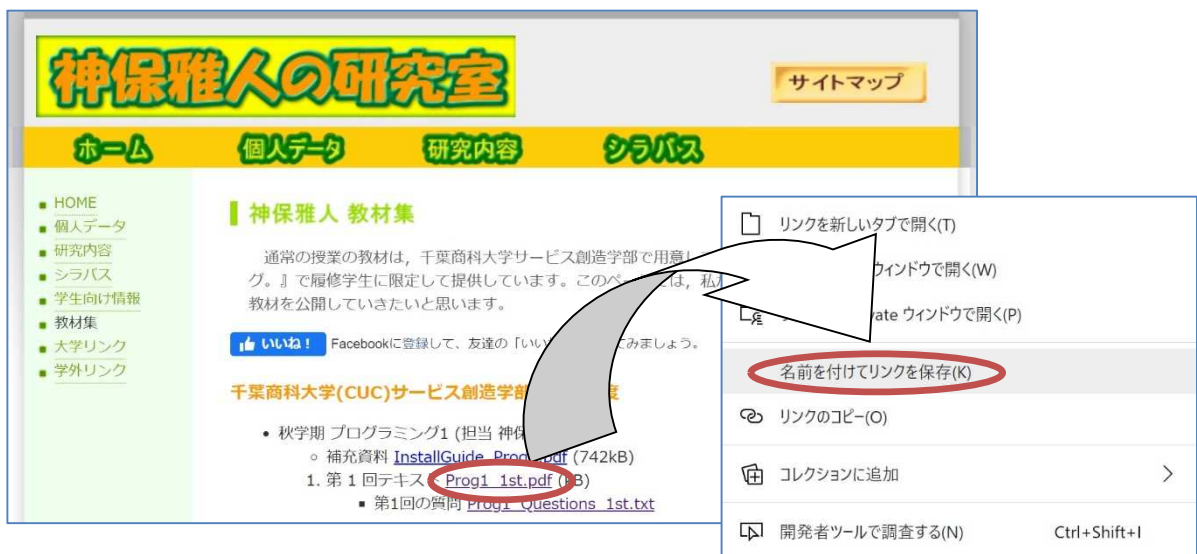
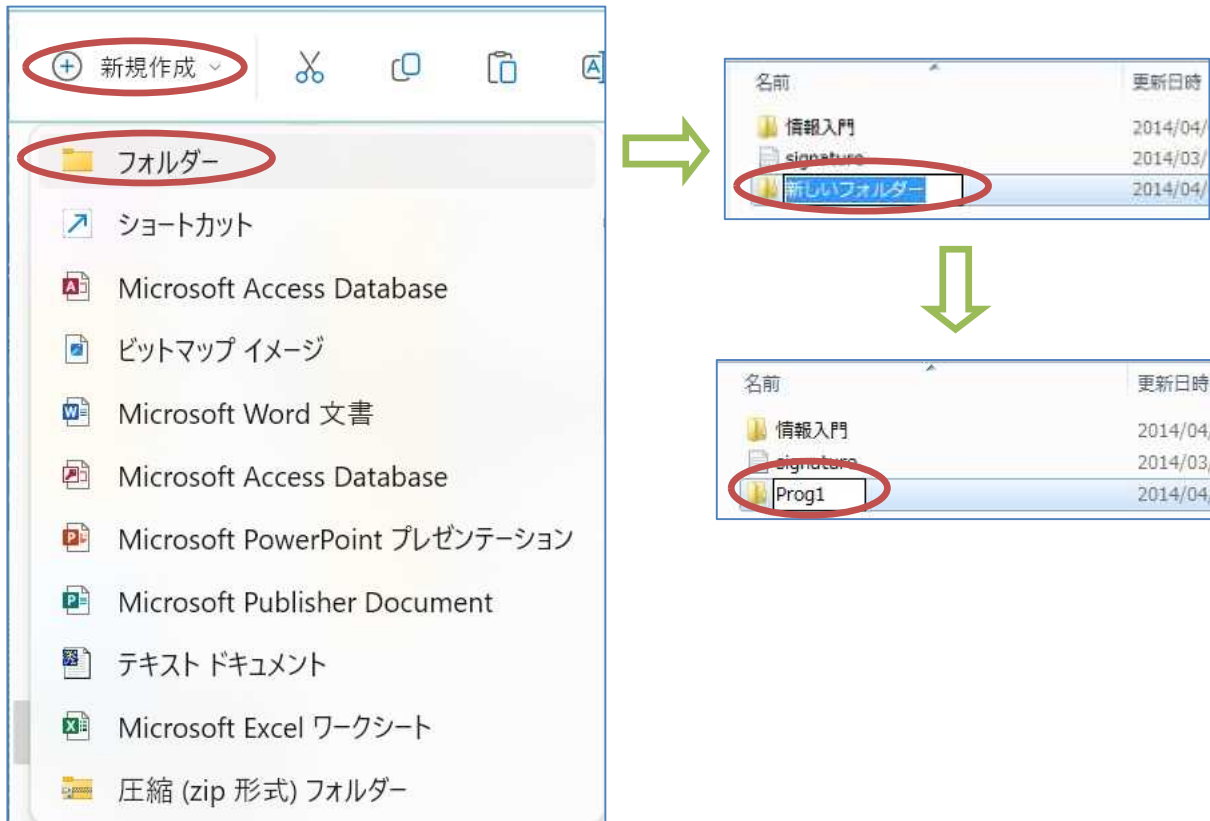
* この授業では、IDE として **Eclipse** を用いる。**Eclipse** は実際のソフトウェア開発現場でよく利用されている。

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、『**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://www3.cuc.ac.jp/~jimbo/textbooks.html>) から教材をダウンロードする。

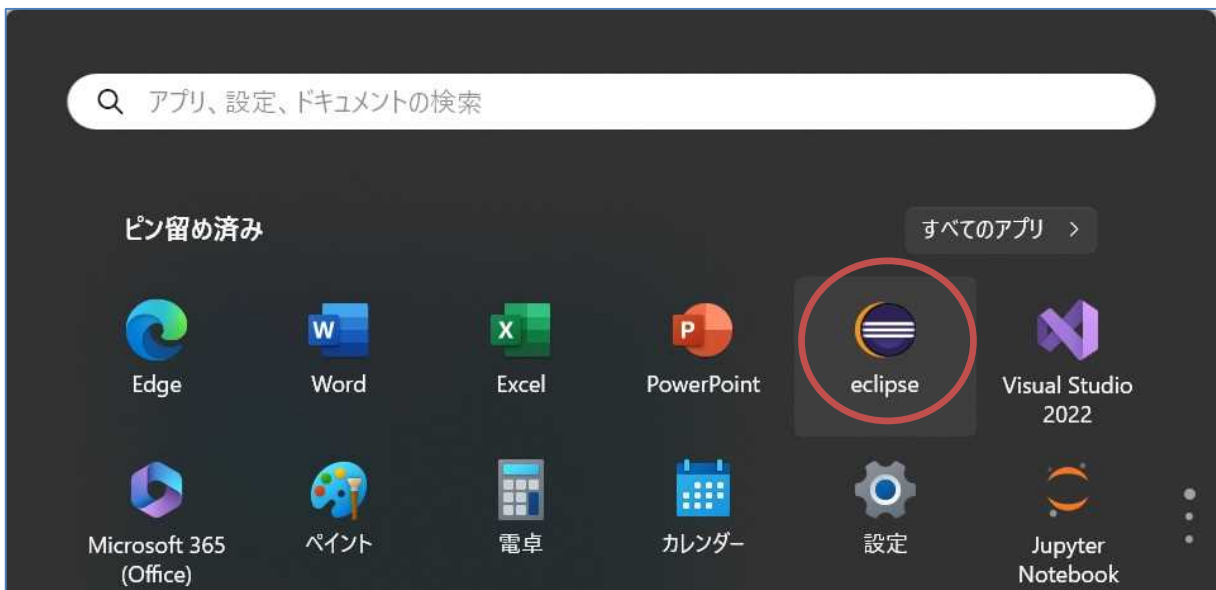


* ここでは『第 1 回テキスト』の右側のリンクをマウスの右ボタンでクリックして『名前を付けてリンクを保存』を選び、先程の『Prog1』フォルダに保存する。

* 第 1 回の例題 2 以降は例題のプログラムの内容や必要となるファイルも教材集に掲載するので、テキスト以外のファイルも同様に『Prog1』フォルダに保存しておく。

2) Eclipse の初期設定

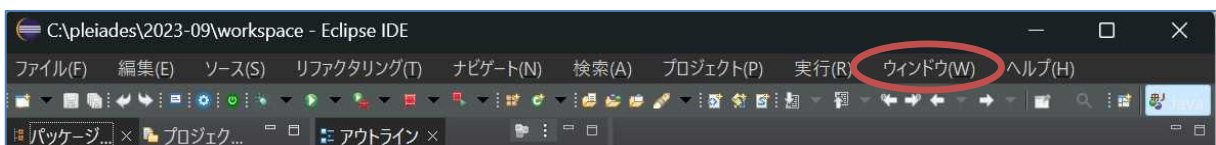
Eclipse を起動する (『プログラミング開発環境について.pdf』を参照)。



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。

【オプションの設定】

- ・『ルック&フィール』の初期状態が「ダーク」となっているので、**白地の画面にしたい場合には**「クラシック」または「ライト」に設定し直す (教材はこれ以降「クラシック」の画面)。
- ・『ウィンドウ』を選択



- ・『設定』を選択

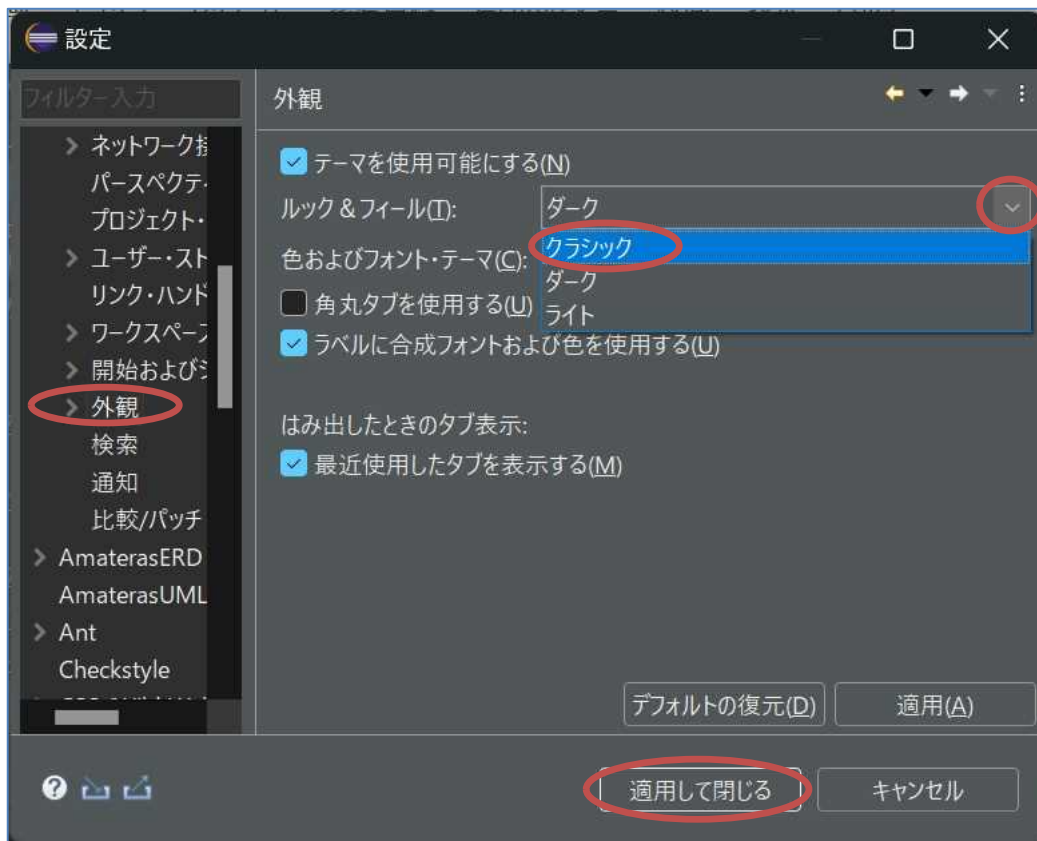


- ・『一般』を選択

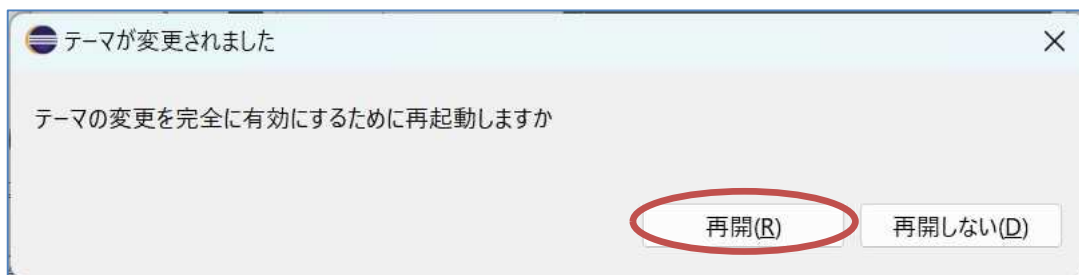


- ・『外観』を選択

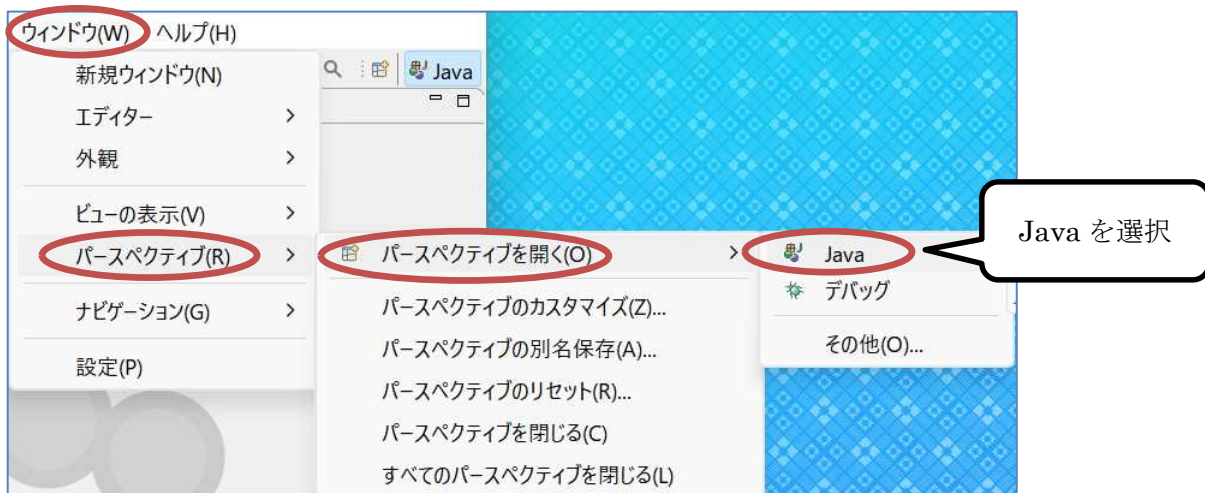
(図は次のページ)

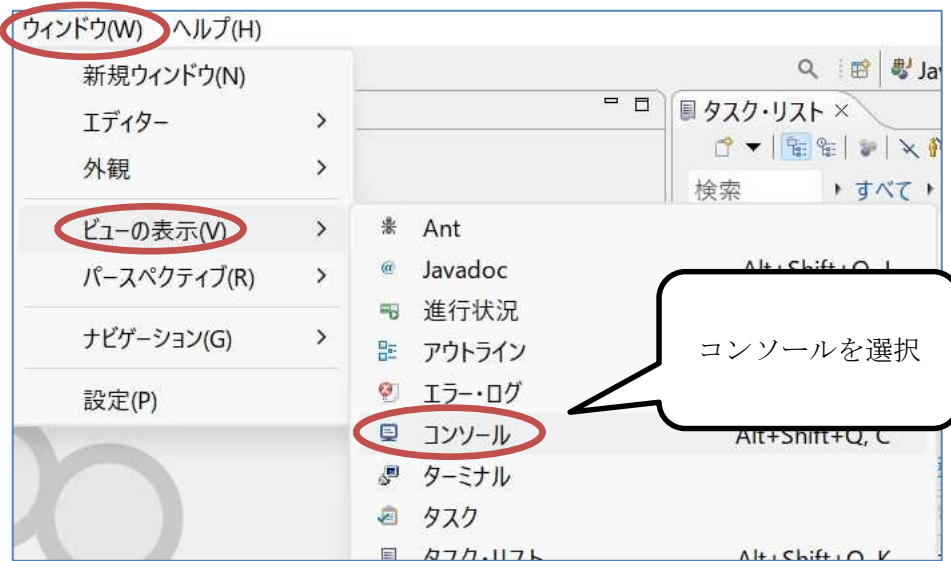


・『適用して閉じる』をクリックする。Eclipse を終了させて、再度起動する。

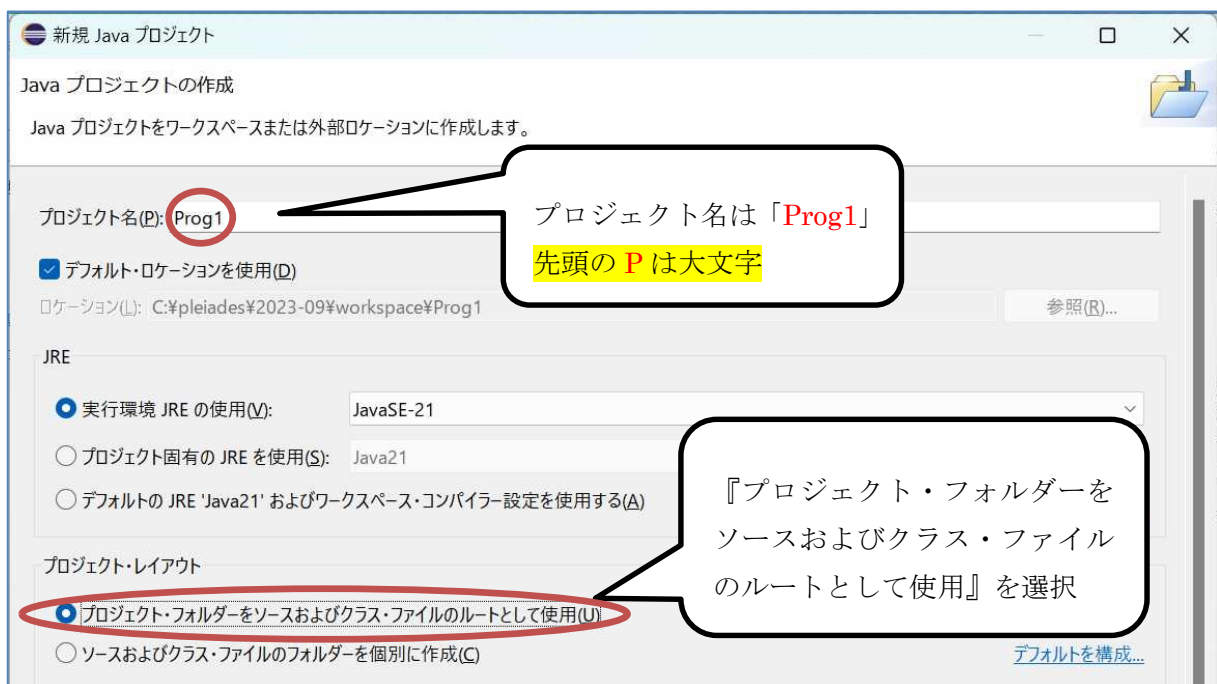
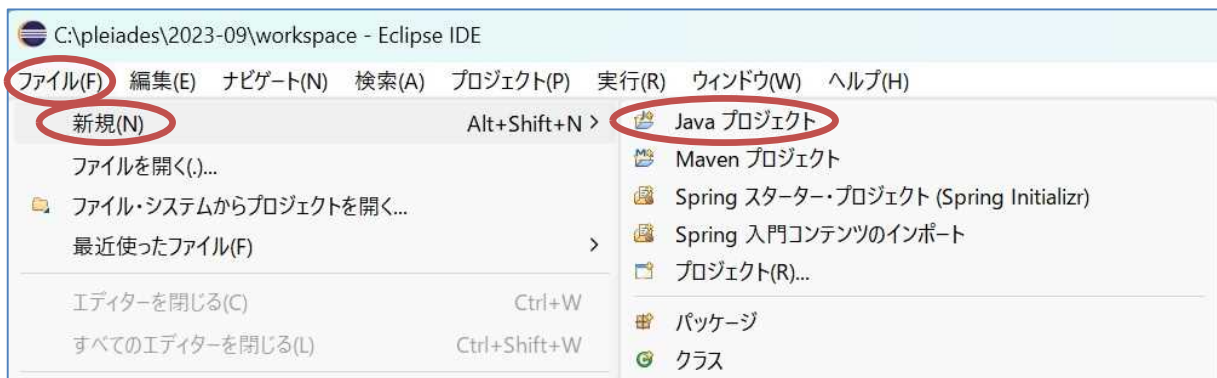


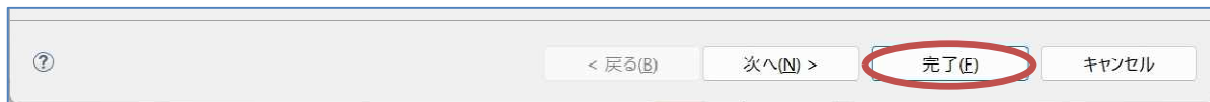
【必須の設定】





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。これを作成して、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、**赤色の下線の箇所は自分の名前に置き換える**。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) 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神保です。");  
    }  
}
```

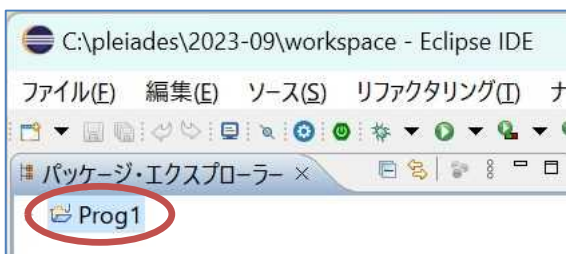
この 1 行を入力

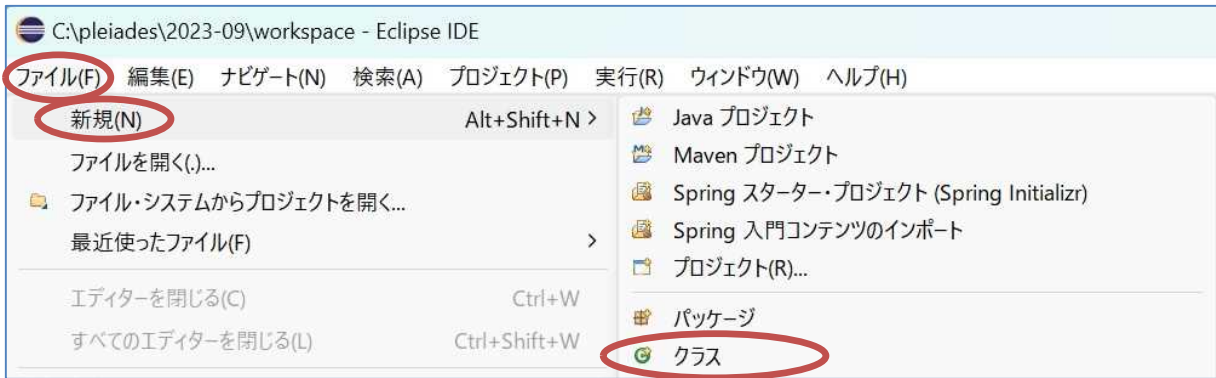
【解説】

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。

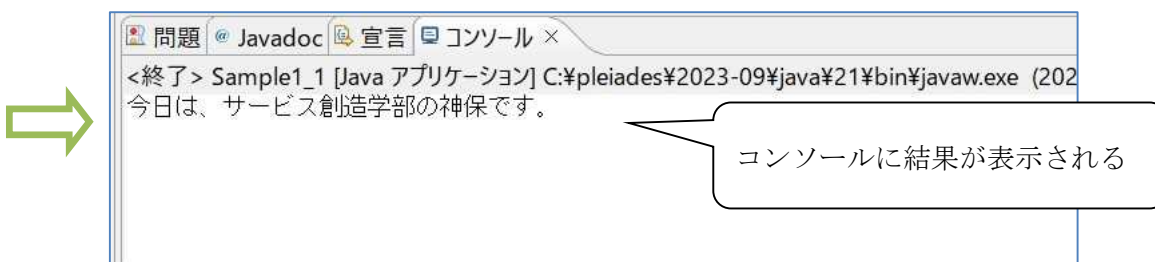
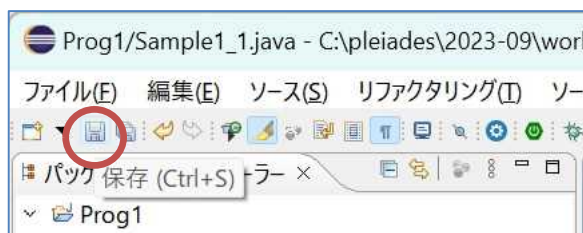
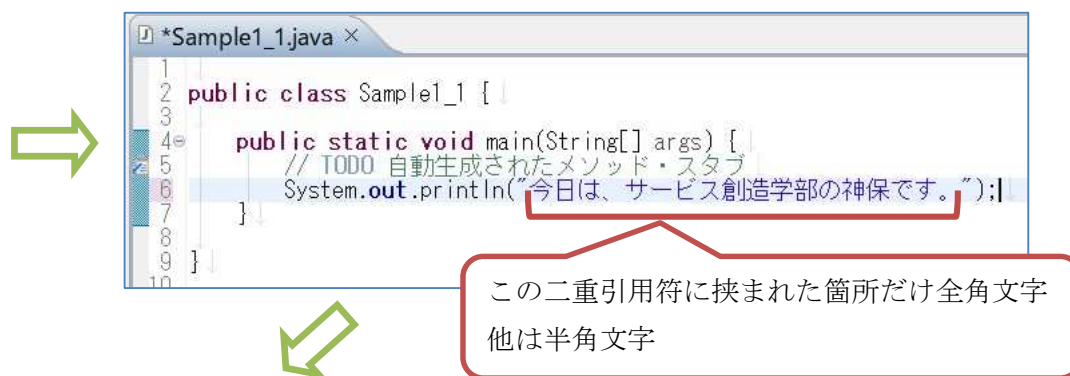
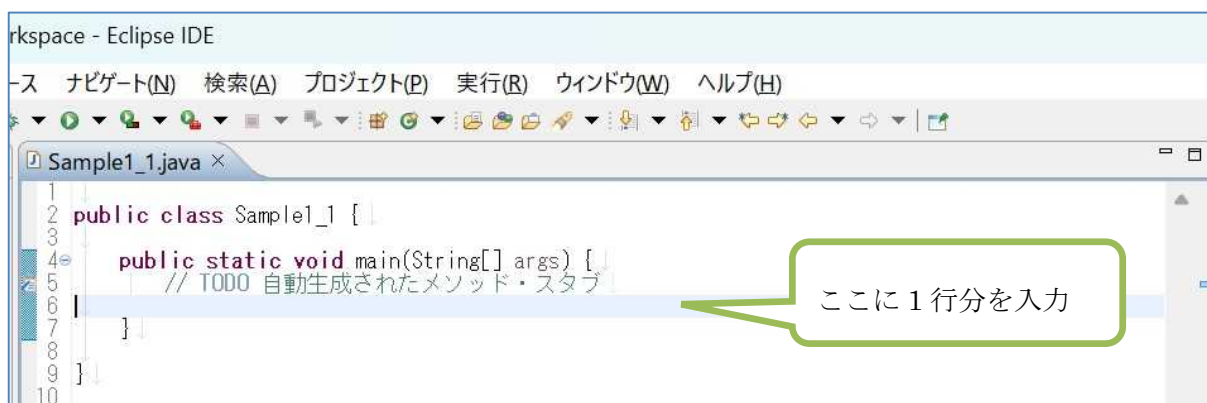




次の様なダイアログが出てきたら、

- ・『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。



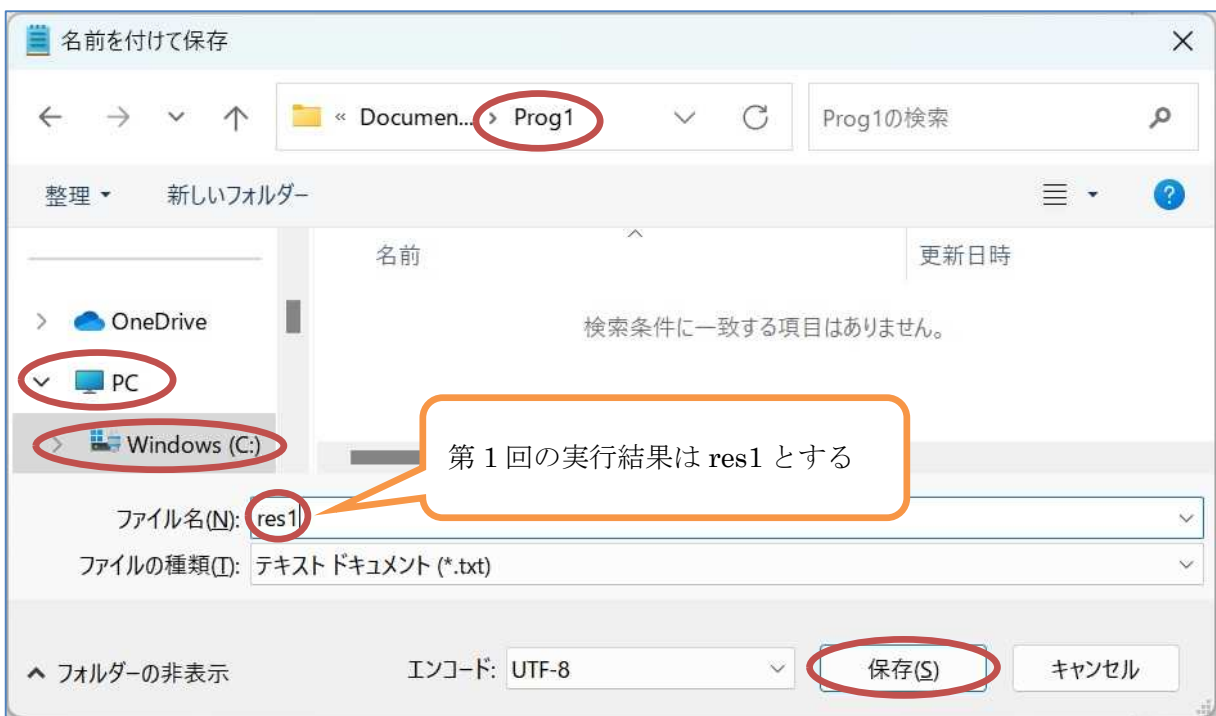
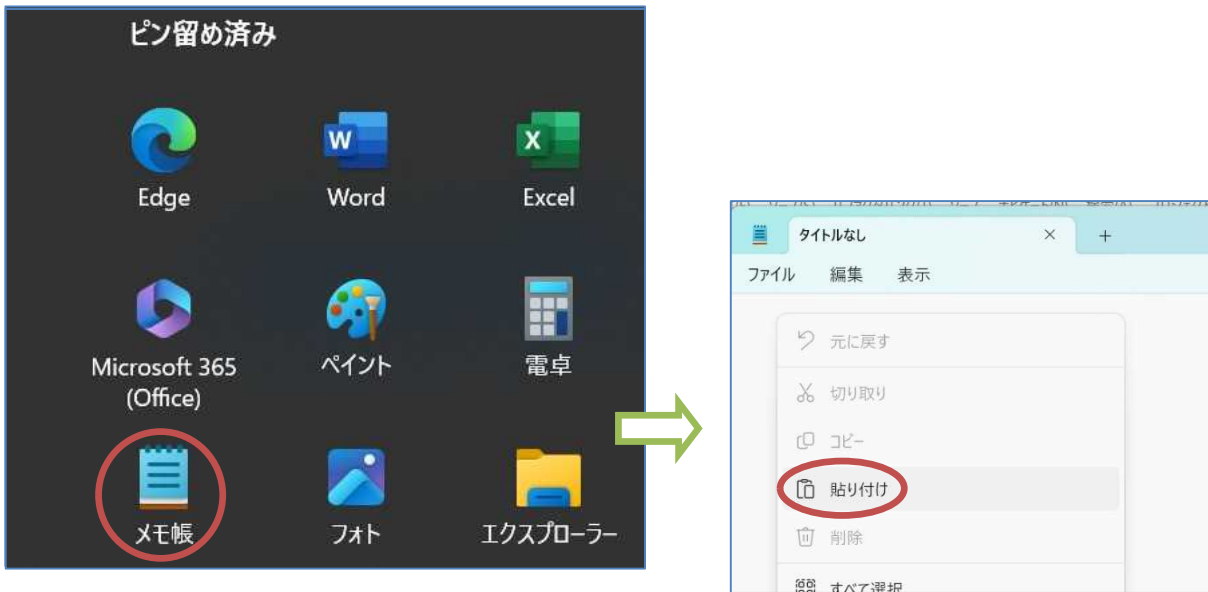
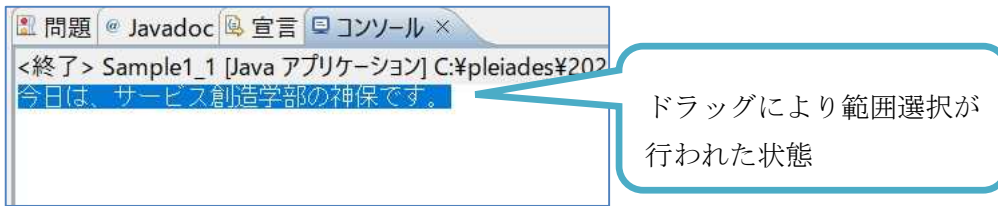


* 例題及び演習課題のプログラムの実行結果は後で提出するので、スタートメニューからメモ帳等のテキストエディタを起動し、コンソールで範囲選択してコピーした内容を貼り付けて、**res1.txt** の名前でテキストファイルとして保存しておく。

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

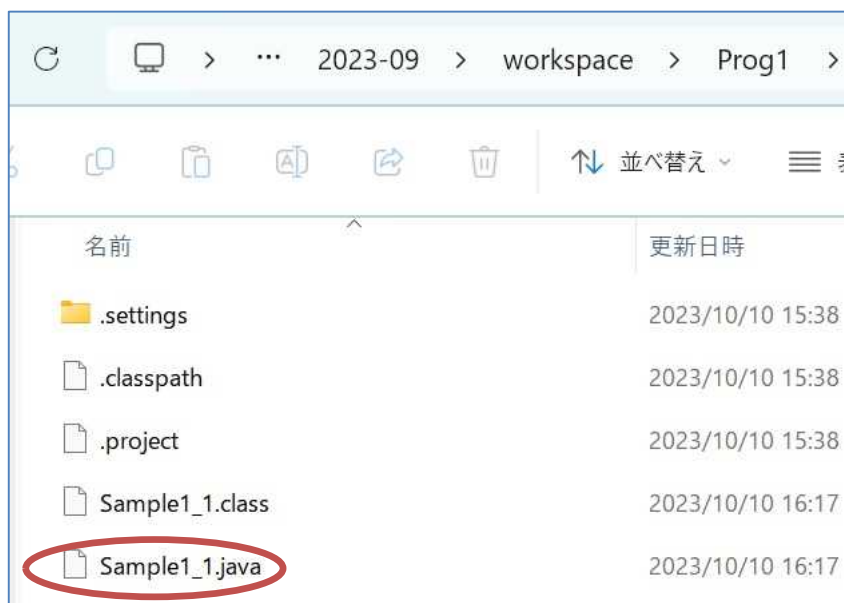
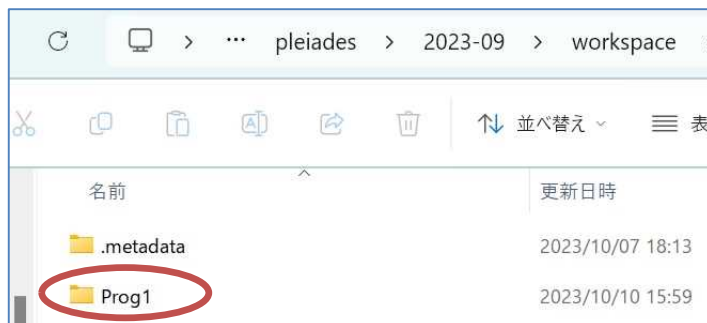
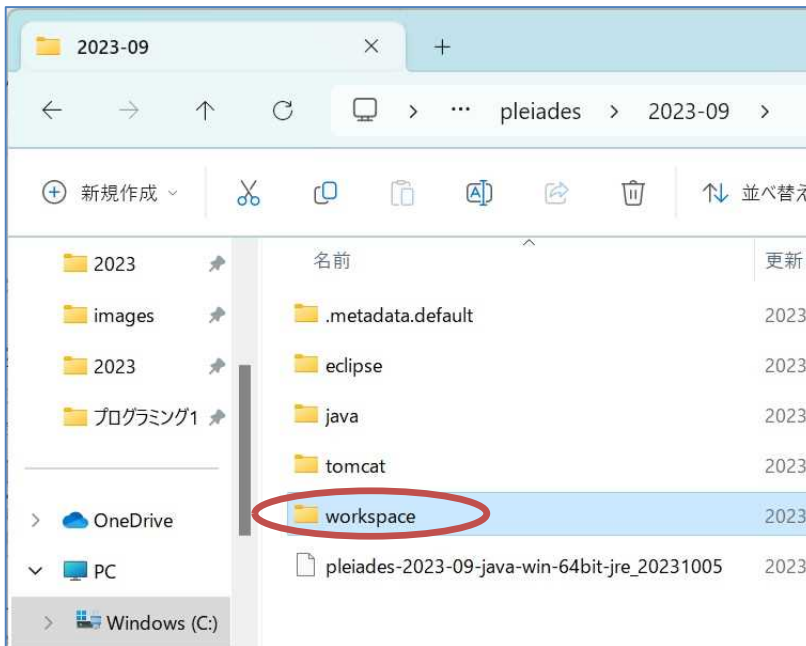
一することが出来る。メモ帳を起動して、右クリックして『貼り付け』を選択すると、クリップボードの内容が貼り付けられる。



* このファイルに、この後に続く例題や演習の実行結果を追記していき、上書き保存をする。

プログラムの保存場所

作成したプログラムはPleiadesを自分のPCにインストールした際に作成された『workspace』フォルダの中に、今回プロジェクトを作成した際に出来た『Prog1』フォルダに保存されている。



* 拡張子がjavaのものがソースプログラムのファイルなので.classのもの間違えない様に！

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 ($-2^{31} \sim 2^{31} - 1 = -2147483648 \sim 2147483647$)
	long	64 ビット	整数 ($-2^{63} \sim 2^{63} - 1$)
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。これを新規に作成して、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。

なお、これから先の例題のプログラム作成には教材集からダウンロードしたテキストファイルの中身を利用することにする。

```
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。

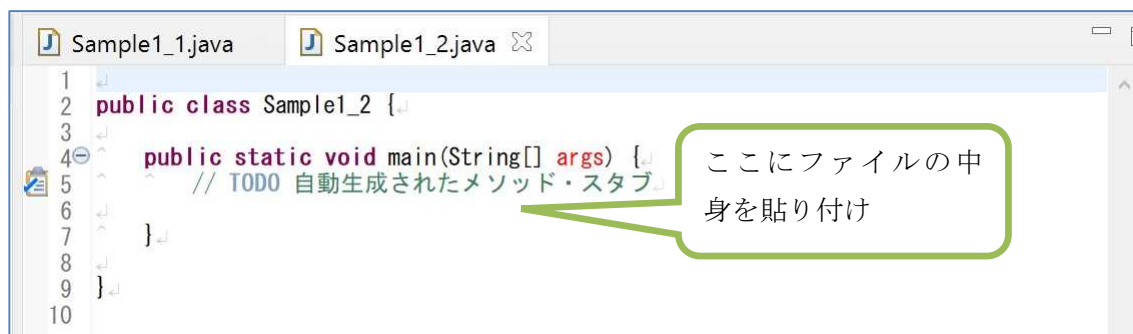
【Sample1_2_main.txt の利用法】

ここから先は、例題のプログラムに関しては、`main` メソッドの中身をテキストファイルとして教材集の Web ページに置いている。その利用法について解説する。

この教材の pp. 10-11 と同様にして、[ファイル] → [新規] → [クラス]の順で選択し、クラス名を `Sample1_2` とする。

クラスのブロック及びその中の `main` メソッドのブロックが生成されるので、その中 (次のページの図では 6 行目) にカーソルを置き、ここに `Sample1_2_main.txt` の中身をコピーして貼り付ける。

(図は次のページ)



* 具体的には、ドキュメントフォルダ内の Prog1 のフォルダを開き、Sample1_2_main.txt をダブルクリックすると、メモ帳が起動する。メモ帳側で、[編集] → [すべて選択]を選択した上で、[編集] → [コピー]を選択する。⇒ Eclipse 側で、[編集] → [貼り付け] を選択する。(保存を忘れない様に)

文字型変数と String クラス

文字型変数

Java 言語では、1 文字を格納するには、**char 型**の変数を用いる。その値は **Unicode** で表される文字コードの数値で、**16 ビットの符号無し整数**である。

例) char c; // 文字 (character) のデータ型の変数 c を宣言
char c1 = 'A'; // 文字 A の代入 (一重引用符で挟む)
char n1 = '神'; // 文字 神 の代入 (Java 言語では全角文字も 1 文字)

String クラス

Java 言語では、文字列を扱う String クラスがパッケージ java.lang の中に用意されている。String クラスは、**作成後に変更できない文字列**を表す。

例) String str = "CUC";

右辺の "CUC" の様に、文字列を半角の二重引用符で挟んだものは**文字列リテラル**と呼ばれ、String クラスの**インスタンス**("実体")への参照として扱われるが、このインスタンスに関しては後述する。

例題 3 (標準出力装置への文字及び文字列の表示)

次のプログラムは、標準出力装置に文字及び文字列を表示させるものである。これを作成して、実行せよ。ここで、クラス名は **Sample1_3**、ソースファイル名は **Sample1_3.java** とする。

```
public class Sample1_3 {
    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        char n1 = '神', n2 = '保';
    }
}
```



```
String str1 = "Service";
String str2 = " Innovation";

System.out.println("n1 に格納されている文字は: " + n1);
System.out.println("n2 に格納されている文字は: " + n2);
System.out.println("両者をつなげて表示すると: " + n1 + n2);

System.out.println("str1 に格納されている文字列は: " + str1);
System.out.println("str2 に格納されている文字列は: " + str2);
System.out.println("両者をつなげて表示すると: " + str1 + str2);
}
}
```

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 1 のソースプログラムのファイル `Sample1_1.java` をメールに添付する。
- 2) 演習で考察した内容をメール本文に記述する。
- 3) 例題 1, 例題 2 及び例題 3 のプログラムのコンソールへの出力結果をコピーして貼り付けたテキストファイル `res1.txt` をメールに添付する。
- 4) 第 1 回の理解度確認用の質問ファイル `Prog1_Questions_1st.txt` をダウンロードして、ダブルクリックしてメモ帳で開き、解答を記入して上書き保存し、メールに添付する。

* 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、メールの本文の書き方は情報入門に準拠する。