コンピュータプログラミング
（Java 初級編）

大阪アクティブラーニングスクール

■もくじ

■第１章 Java 編

Lesson0　開発環境を理解しよう

これから皆さんは、プログラミングについて学習していきます。プログラミングにはたくさんの言語があり、それぞれに特徴があり、開発方法も異なります。皆さんがこれから学習する言語は Java 言語です。Java を学ぶ上で必要な用語と開発環境について学びましょう。

1.　プログラムとは

そもそもプログラムとはなんでしょう。電子機器は基を正せば電圧のオンオフしか理解しません。これをもう少し複雑にしたものが機械語になります。人間には機械語が理解できませんので、電子機器への命令を人がわかる言葉で書き表したものが必要になります。これがプログラムです。

2.　プログラム設計書

プログラムを作成していくうえで、いきなりプログラミングをしていくということはありません。通常はプログラム設計を行い、プログラミングをしたのち、プログラムテストを行います。

プログラム設計書は、要求定義を基にシステム要件や基本設計をこれから作成しようとするプログラムの詳細設計において機能ごとに分割したプログラム１本１本の構造や処理手順を詳細にまとめたものです。簡単に言うと、プログラミングを行う上での流れを日本語で記述した設計書ということになります。プログラマーは、このプログラム設計書に基づいてコーディングしてくことになります。このことをプログラミングといいます。

3.　コンパイルとは

プログラムを記述したファイルのことをソースファイルといいます。この時点ではまだ機械はプログラムの内容を理解することができません。そこでプログラムを機械に理解できるようにコンパイルする必要があります。Java ではソースファイル（.java）をコンパイルするとクラスファイル（.class）が作成されます。このクラスファイルを実行することで電子機器を制御することができます。

4.　Java とは

> *「Java は、1995 年に Sun Microsystems から初めてリリースされたプログラミング言語およびコンピューティングプラットフォームです。ユーティリティ、ゲーム、ビジネスアプリケーションなど、最先端のプログラムの基礎となっているテクノロジです。Java は、世界中の 8 億 5000 万台を超える個人用コンピュータや、世界中の何十億台ものデバイス（モバイルデバイスや TV デバイスなど）で動作しています。」(java.com より抜粋)*

5. オブジェクト指向とは

「オブジェクト」とは英語で「もの」という意味です。「オブジェクト」は「インスタンス」と同義に扱われることがあります。インスタンスは実体を指します。オブジェクト指向とは、ソフトウェアを「オブジェクト」としてとらえオブジェクト同士のやり取りをプログラムで表現して組み立てる方法を指します。プログラムの内部構造や操作手順の詳細を知ることなく利用できるというメリットがあります。Java はこのオブジェクト指向で作られた言語です。

6. クラス・メソッド・フィールド

Java の場合、プログラムをただ単に記述するのではなく、「クラス」と「メソッド」と「フィールド」というものを作る必要があります。

クラスというのは、ひとまとまりになったプログラムの最小単位のようなものです。プログラムは、たいてい細々とした機能の組み合わせになっています。例えば普通のアプリケーションでも、**ウィンドウ**とか**ボタン**とか**メニュー**とかいった部品の組み合わせになっています。Java では、プログラムはこうした1つ1つの部品を定義していき、それらを組み合わせて作るようになっています。このプログラムの部品となるのが「**クラス**」なのです。クラスには、この後に登場するメソッドとフィールドをその中に持つことができます。

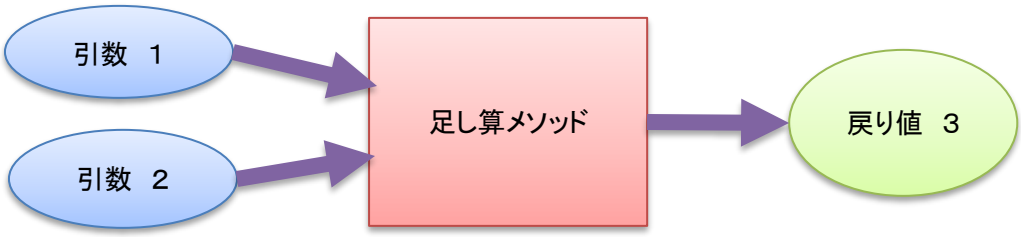
メソッドは、クラスの中に用意することができる細々とした「処理」です。例えばウィンドウのクラスなら「ドラッグした時の処理」とか「中の表示を描く処理」とか「ボタンなどの部品を組み込んだときの処理」とか、そうした細々とした処理がたくさん用意されているはずですね。これが「**メソッド**」なのです。クラスを作るときは、このメソッドというもので様々な処理を作っていきます。これが、Java におけるプログラミングなのです。

クラスは、さまざまな処理だけでなく、「情報」も持つことができます。わかりやすくいえば、「データ」、つまり「いろんな値」を保管しておくことができるのです。これを**フィールド**と呼びます。これらの用語は後半の Android アプリ制作にも関係しますのでしっかり理解してくださいね。

では、このクラスというのはどうやって書くのでしょうか。基本的な書き方をまとめると、こういう風になります。

```
class クラス名 {
    種類 フィールド名;
    種類 フィールド名;
    ……略……
戻り値 メソッド名( 引数 ){
        ……実行する処理……
  }          ……略……}
```

- 戻り値：そのメソッドを呼び出した元に何かの結果を返すためのものです。例えば複雑な計算を行うメソッドを定義したとすると、最後に「計算結果はこれです」というのを渡さないといけませんね？　これが戻り値です。戻り値は、プログラムに返す値の種類（int とか String とか）を指定して書きます。
- 引数：何かの処理をさせるメソッドでは、そのために何かの値が必要となることもあります。「これを実行するには、これとこれの値を下さい」というのを指定しないといけないことがあるわけです。こういうときに、メソッドに必要な値を渡すためのものが引数です。これは、値の種類（int とか String とか）と、それを収める変数を書きます。またいくつもある場合は、それぞれをカンマで区切って書きます。



7.　インスタンス

次に**インスタンス**について。インスタンスというのは、そのクラスを元にしてメモリー内に作成された「操作できるオブジェクト」のことです。Java　では、クラスを利用する際には、通常、こうやってインスタンスを作り、それを操作するのです。

もし、クラスを作って直接それを操作するとしたら、常にクラス１つしかオブジェクトが作れないことになります。例えば、ウィンドウのクラスがあったとしても、それ１つしか使えないのでは困るでしょう。複数のクラスを使いたい時には、また同じようなウィンドウのクラスを作らないといけなくなってしまいます。

そこで、クラスは最初に「インスタンス」というものを作り、このインスタンスにあるメソッドやフィールドを呼び出して操作するようにした、というわけです。インスタンスは、いわば「クラスのコピー」です。このように、クラスを利用する際には必ずそのインスタンスを作りインスタンスの中のメソッドなどを操作すればいいのです。

8.　開発環境

この演習で Java の開発を行う上で必要なアプリケーションは Eclipse というアプリケーションを利用します。Eclipse は IBM によって開発された統合開発環境（IDE）の一つで、高機能でありながらオープンソースであるため、プラグインを導入することで Java をはじめとした C++や PHP といったいくつかの言語に対応しています。このアプリケーションはオープンソースとなっている為、無償で入手することが可能です。もともと商用ベースに作られたアプリケーションですから非常に安定しています。

Ecripse を使ってプログラミングを行っていくにはワークスペースを用意する必要があります。ワークスペースはその名の通り作業領域ですので、作成したプログラムの保存場所と理解しておきましょう。ワークスペースの作成先は、各自でしっかり考え保存先に注意しましょう。その保存場所にクラス（クラス名.java）を作成し、そこにプログラムを記述していきます。クラスは複数用意することができます。それぞれが独立した一つのプログラムですが、それぞれのクラスで連携を取りながらパッケージ内のプログラムとして作成することも可能です。

それでは実際にアプリケーションを使って操作をしてみましょう。

※この Lesson については環境により変わりますので講師の指示に従ってください

まずは、プロジェクトを用意します。

① eclipse アイコンを W クリックする

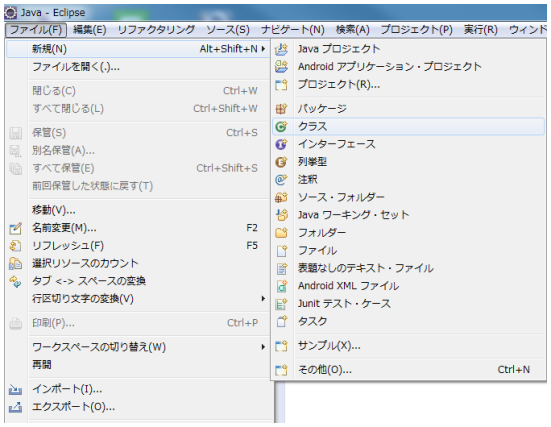
② eclipse プログラムが起動するので、ファイルメニューから新規→Java プロジェクトで次へ

③ プロジェクト名に好きな名前を入れて「完了」をクリックする

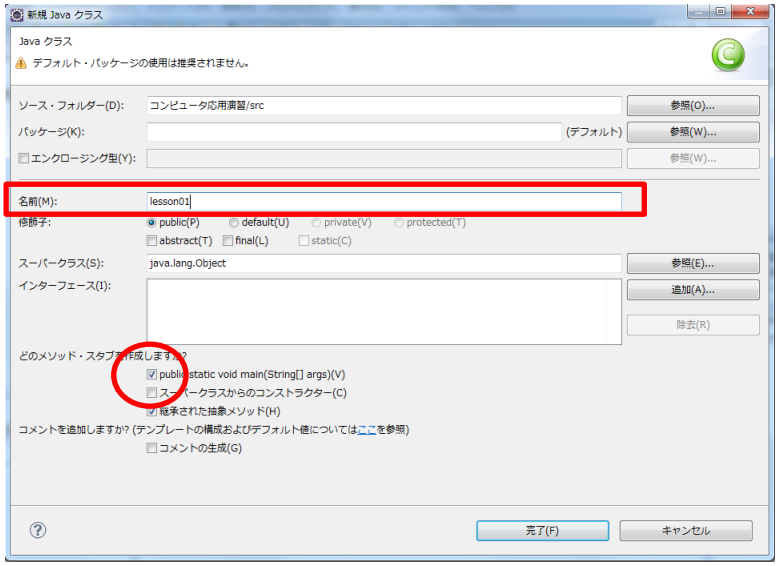
C:¥java¥workspace¥にプロジェクトが作成されます。

これでプロジェクトが作成されます。次にクラスを用意します。

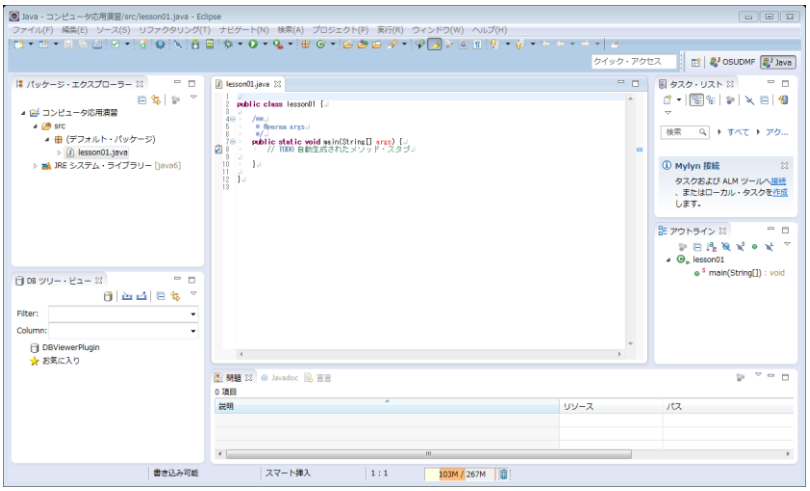
④ ファイル→新規→クラスで新規 Java クラスダイアログ表示



⑤　名前(M):に好きな名前(クラス名になります**※必ず半角入力**)を入れて、public static main(String[]args)にチェックを入れる



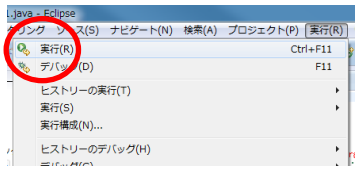
⑥　完了　プログラム記述を入力できる画面が出てくる



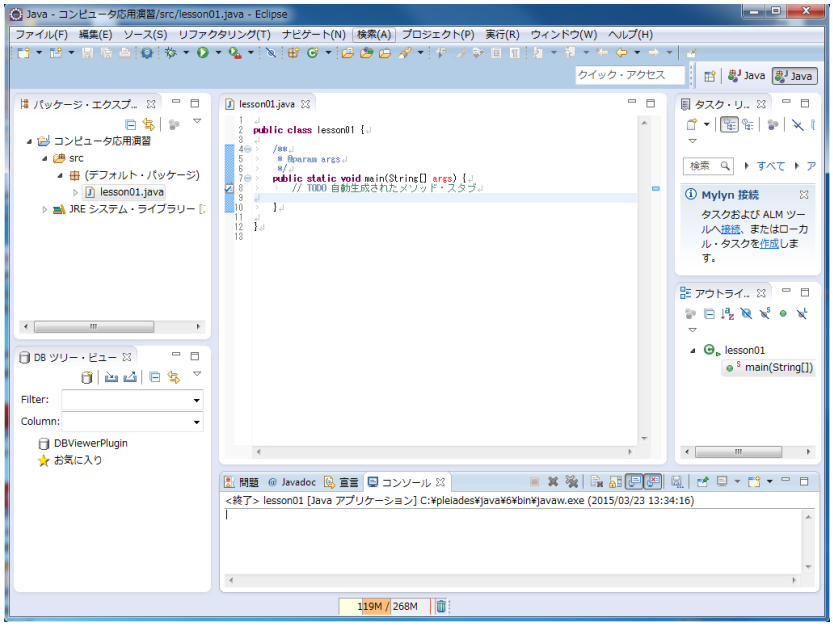
これでクラスが作成され、プログラミングを始める準備が整います。

⑦　ソース(プログラム記述)を入力する

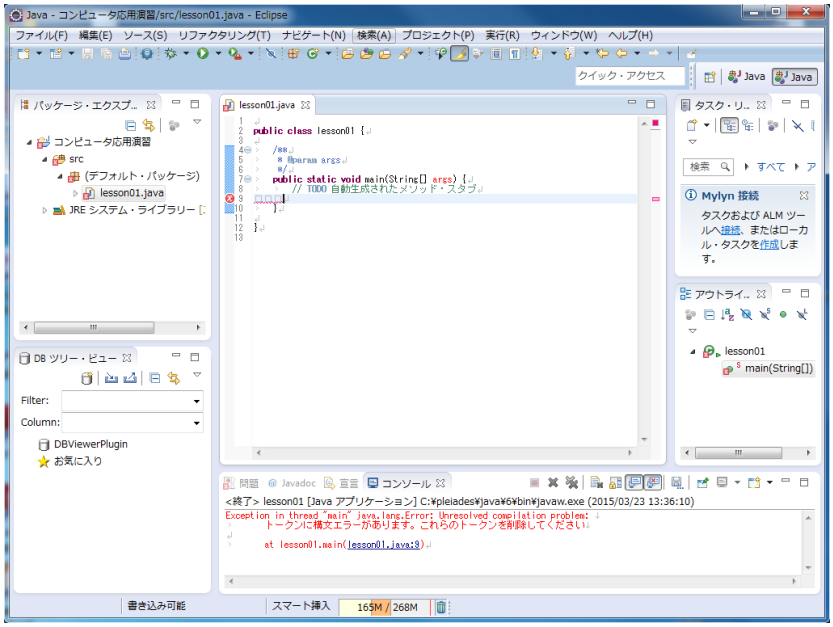
⑧　実行→実行でエラーがなければ実行結果が、エラーがあればエラーメッセージが表示される



（実行結果）



（エラーになった場合の実行結果）

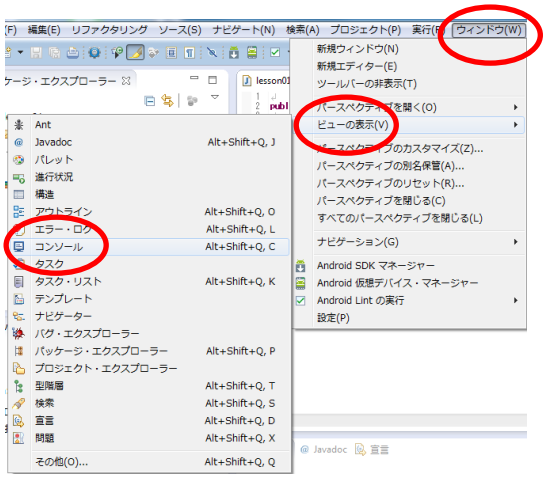


ワンポイント！

コンソールが出てないときの出し方は？

メニューのウィンドウ→ビューの表示→コンソールで表示できる。

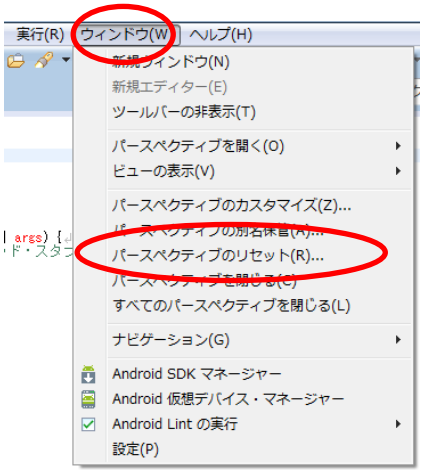
又は**実行**をしたら出る。



ワンポイント！

ビューがバラバラになってしまったときの戻し方は？

メニューのウィンドウ→パースペクティブのリセットで初期状態に戻すことができる。



Lesson1　簡単なプログラムを作ってみよう

1.　基本的なルール

では、実際にプログラムを作ってみましょう。プログラミングを行っていくうえで、プログラミングに必要な基本的な記述のルールを学びましょう。

(1) 命令の文字

命令は必ず半角で「abc」という風に記述します。「abc」と「ABC」は人間にとっては同じですが、コンピュータにとっては別のものと認識しています。入力をする際は十分に注意しましょう。また、「ａｂｃ」というような全角文字は画面に文字を表示するときやコメントを入れる時以外は使いません。スペースも文字とみなします。特に全角のスペースは絶対に用いてはいけません。見た目でわかりにくくなるため、実行した場合、エラー数がありえないほど多く出ます。

(2) コメントのルール

コメントはそこに書かれたプログラムが何について書かれているのかなどを明確に示してくれるので、他のプログラマーが見てもわかりやすいように表示しておくと便利です。また、自分で作ったプログラムであっても長くなってくると初めに書いたものが何かわかりにくくなってきます。その際コメントを適切に利用していれば、その後の処理をスムーズに行うことができます。

コメントの入れ方は 2 種類あります。一つ目は、「 // 」を入れてその後ろに説明文を入れる方法です。この際、コメントの有効範囲は、行の終わりまでになります。二つ目は、「 /* 」と「 */ 」の間に説明文を入れる方法です。この方法では、複数行にわたってコメントを入力したい場合に利用します。

コメントは命令ではないのでどんな文章でも問題はありません。積極的にコメントを入れて、あとから見てわかるようにしましょう。

例１）　// 画面から文字を入力する処理
例２）　/* 画面から文字を入力する処理 */

(3) main メソッドは必須

Java は、public static void main という予約された名前の main メソッドから実行されますので、必ず作成します。メソッドの記述ルールとしては下記のようになります。

［修飾子］戻り値のデータ型 メソッド名（引数 1，引数 2，．．．．）{
}

最初は次の記述で構いません。　例）public static void main(String args[])

(4) ブロックの区切り

ブロックや命令が継続するものは{と}で囲む必要があります。これを忘れるエラーが初心者では多く見受けられます。必ず始まりの「 { 」と終わりの「 } 」の数を合わせましょう。

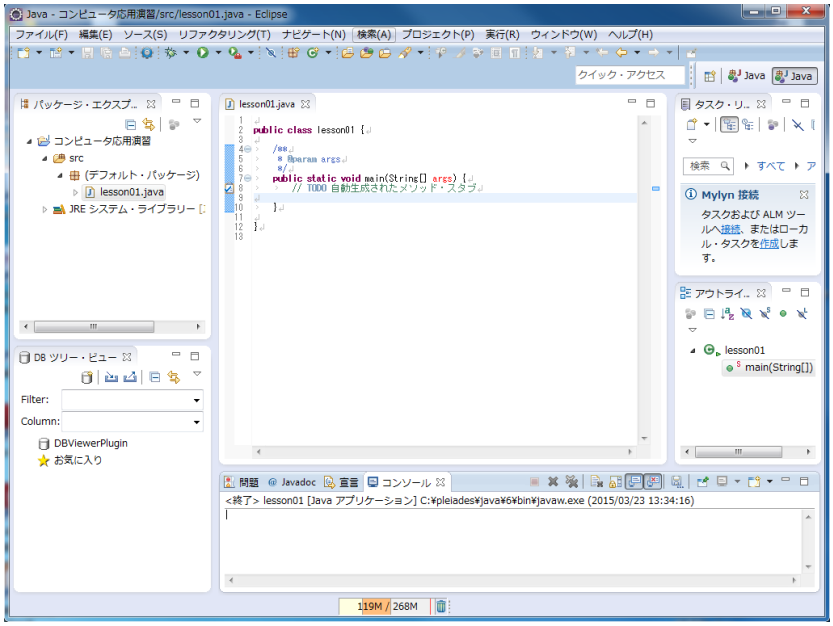
(5) コードの末尾

ブロック内では、コードの末尾にセミコロン；を付けます。コロン：とはあらわす意味が異なりますので注意してください。

以下のソースは簡単なクラスの記述です。

```
/* 最も単純なクラス */
public class lesson01{
    public static void main(String[] args)
    {
    }
}
```

（実行結果）



前述のプログラムは実行してもまったく何も行わないが、プログラムであることは確かです。

また、改行は個人のセンスが問われるものです、最初のうちはツールなどが勝手にしてくれる改行を守りましょう。

【注意事項】全角と半角は忠実に守りましょう。ここまでの記述は全部半角文字です。特にスペースの全角は、エラー数を極端に増やす原因になるので、十分に注意することが必要です。

2.　文字を表示させよう

それでは、文字を表示するプログラミングを行っていきましょう。プログラムに命令を与えるには、クラス名やメソッド名の後にある｛ と ｝の中に命令を記述する必要があります。

まず、画面に表示する命令は、

System クラス.out フィールド.print メソッド(引数);

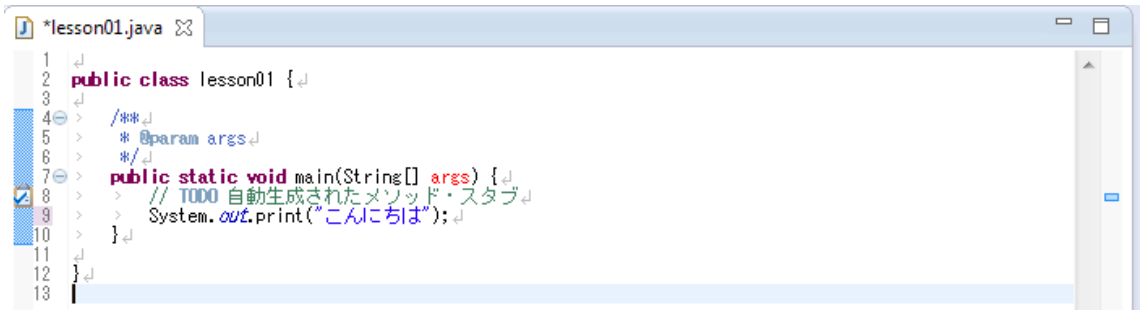
と記述します。Java には System というクラスがあって、それには out というフィールドが用意されています。この out には、値を出力するためのクラス（PrintStream というもの）のインスタンス（オブジェクト）が入っているのです。このインスタンスの中にある「print」というメソッドを呼び出していた、というわけです。

例えば、画面に**「こんにちは」**と表示するには以下の様に記述します。最後にセミコロン；が必ず必要になるので注意しましょう。文字の表示には全角文字が利用可能です。全角文字を利用するには ” こんにちは ” のように「 ”（ダブルクォート）」で囲みます。

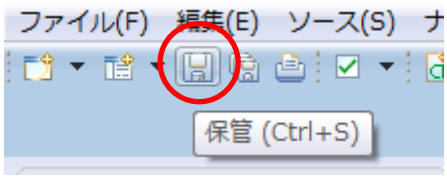
System.out.print ("こんにちは") ;

「System」「out」「print」といった3つの要素があって、その後に引数の（ ）が付けられていることがわかります。

それでは実際に入力をしてみましょう。下図のように入力します。



9 行目に「System.out.print(“こんにちは”);」を追加します。入力ができたらツールバーにある「保管」をクリックしておきましょう。いわゆるファイルの保存を eclipse では保管と呼びます。

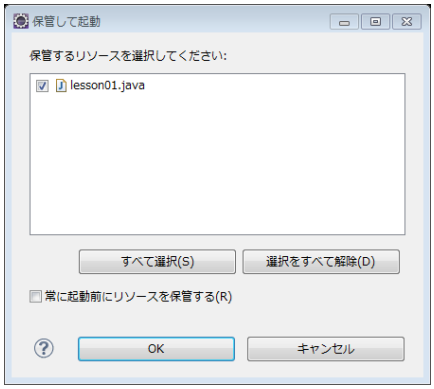


3. 作ったプログラムを実行してみよう

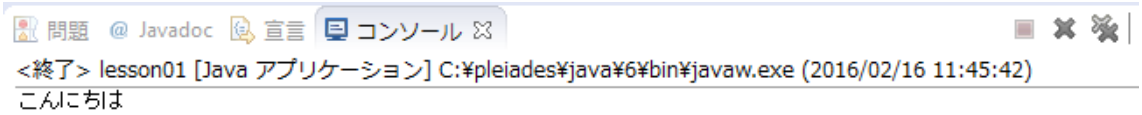
折角プログラミングを行ったのですから、今度はそのプログラムを実行してみましょう。プログラムの実行は、メニューの「実行(R)」から「実行(R)」をクリックします。または、ツールバーにある

をクリックすることで実行できます。

※下記のような画面が出る場合がありますが、これはプログラムが保管（保存）されていない場合に保管をしてから実行してくださいというメッセージですので、そのまま「OK」をクリックすれば、保管して実行してくれます。



（実行例）



4. エラーの修正

問題なく表示できましたか？プログラムは文法上の間違いがあるとそもそもコンパイルをしてくれない場合もありますが、コンパイルができていても意図した結果に結びつかないことが多々あります。このエラーをとる作業が、プログラミングの一番大事な作業です。

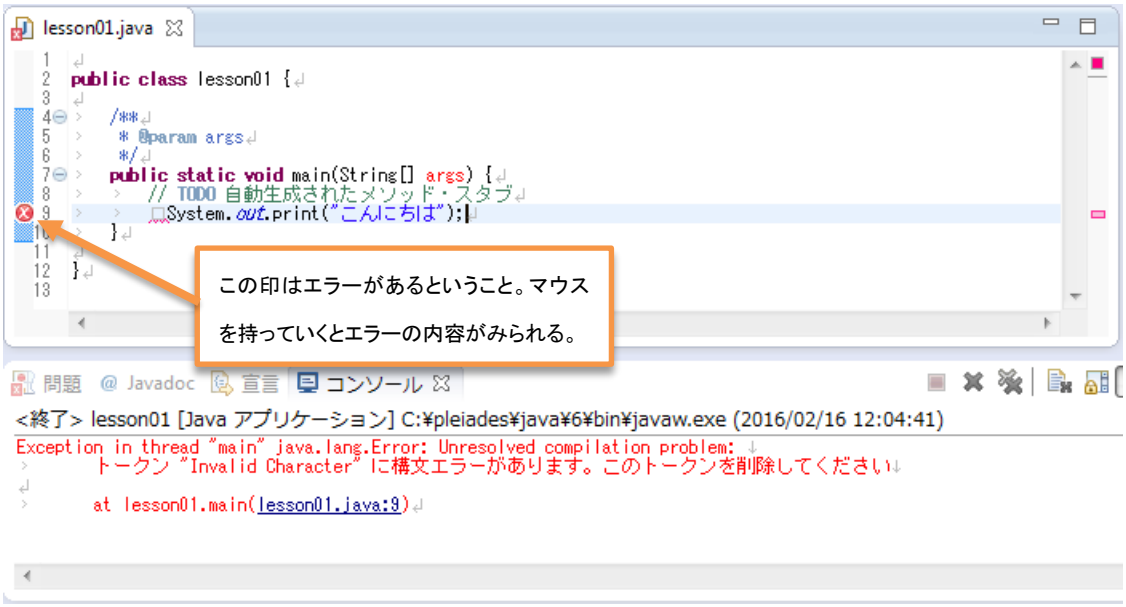
うまくいかない人は、「打ち間違い」又は「初期設定の間違い」の可能性があります。以下のことをチェックしてみてください。

- クラス名を全角で入力していないか？ → クラスを作り直す。
- プログラムを全角で入力していなか？ → 英字や記号を見直す。
- 全角のスペースが入っていないか？ → 入っていたら消す。
- ダブルクォーテーション（ “ ）はちゃんと対になっているか？
- 文末にセミコロン（ ； ）は入っているか？
- コメントアウトは正しく定義されているか？ →「 /* ～ */ 」の構造を見直す。
- スペルミスはないか？ → 正しいスペルで入力しなおす。

何事も最初が肝心です、毎回同じミスをしていると、気力も失せてしまいます。特に全角のスペースのエラーは数も多量にでますから注意しなくてはいけません。

※Eclipse では構文エラーと表示されます。

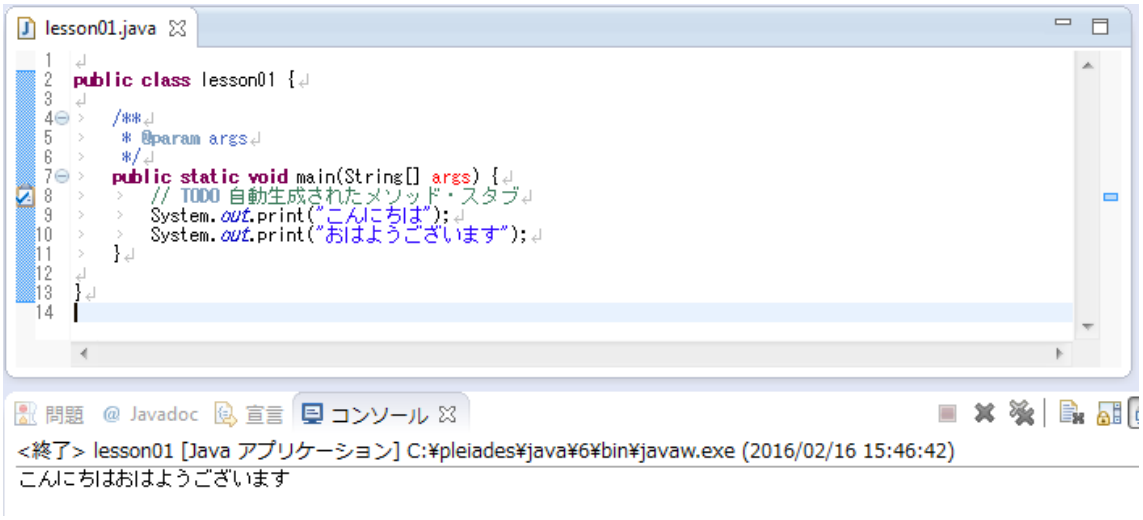
（全角が入ったらエラーになる例）



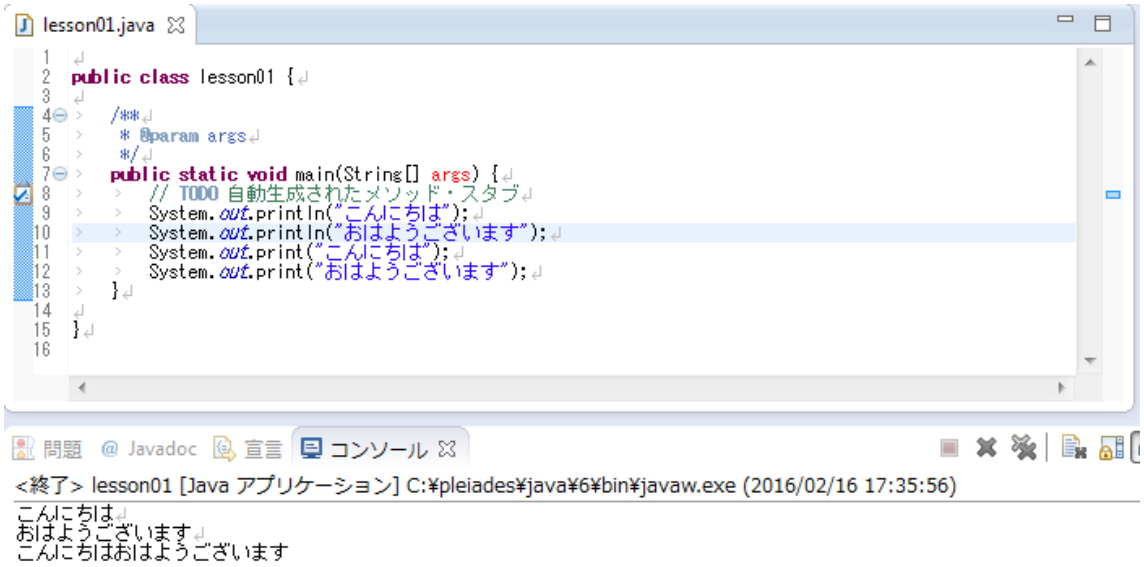
5. プログラムの再編集（print()と println()とエスケープシーケンス）

正しく表示をしたらもう少しこのプログラムを編集してみましょう。

プログラムを下図のように修正し実行してみましょう。



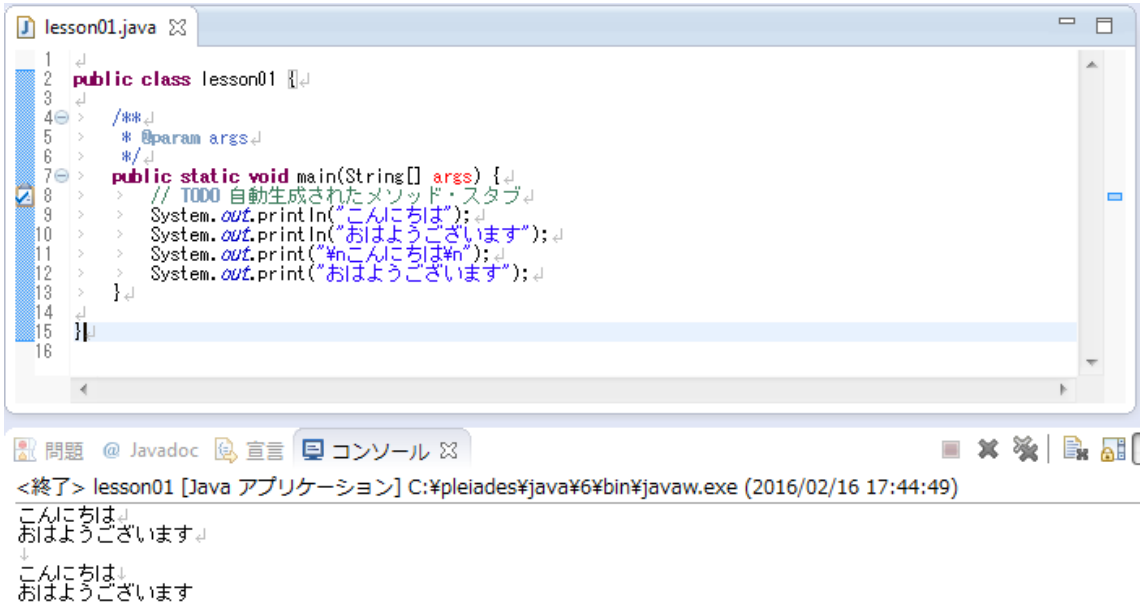
実行結果は、文字列を表示しており、一見問題ないように見えますが、プログラム上は 2 行に分けて表示しており、コンソールでは 1 行で表示されています。これは、改行がされていない状態になる為、このような表示になります。文の終わりで改行する為には、print ではなく println を使います。



print()は内容を表示するだけのメソッドで、println()は指定した内容の後ろに改行をつけるメソッドになります。2 つのメソッドをしっかり使い分けられるようにしましょう。

また、途中で改行したい時には、エスケープシーケンスを入れてやります。エスケープシーケンスとは、キーボードから入力できない文字を他の文字の組み合わせで表現したもののことです。例えば、改行は「 ¥n 」、タブは「 ¥t 」というように表現します。その他シングルクオーテーションやダブルクオーテーションも「 ¥' 」「 ¥" 」と表現します。また、円マークは「 ¥¥ 」と表現します。

下図のように修正して、実行してみましょう。



どうですか？正しく改行できましたか？

Java 言語の基本はここにあるのでしっかり理解しておきましょう。

【練習問題】

新しいクラス（rensyu01）を用意し、画面表示命令を用いて以下の様な表示をさせてください。

```
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊＊＊＊＊　Yes. we can!!　　＊＊＊＊＊
＊＊＊＊＊　　Obama as Nocchi　＊＊＊＊＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
```

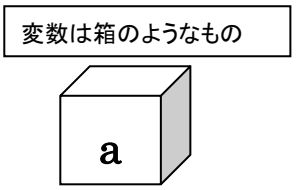
【記述例】

Lesson2　変数を理解しよう

前回は画面に文字を表示させることを学びました。しかし、表示をするときには何も固定の文字の表示だけではありません、状況に応じて様々な文字列や数値などを表示します。そんなときには変数というものが必要になってきます。

変数というのは固定で数字や文字を使うのではなく、何が来るかわからない時や変動する数などを扱う時の為に使うもので、数字や文字を入れるための**容器**のようなものと考えればよいかと思います。

具体的には以下の様に記述します。



例）int a;　→中に整数が入る a という名前の変数

前についている **int** は変数が扱える範囲を示すもので、この場合には整数値を扱うことができるものです、うしろの　**a**　は変数名で、あらかじめ決まったルールというのはないので、自分でわかりやすい名前を付けて良いのですが、あまり長い名前を付けると面倒なので必要最小限の長さで良いと思います。また、命令として予約されている言葉は使えないので注意が必要です。範囲を示すものは変数の型といいます、型のいろいろを以下に示すのでしっかり学習しましょう。

■表　Java の基本的なデータ型

タイプ	型名	サイズ	意味	範囲
整数型	byte	8	8 ビット整数	-128～127
	short	16	単長整数	-32768～32767
	int	32	整数	約±21 億 4700 万
	long	64	倍長整数	約±9223372 兆
浮動小数点型	float	32	単精度浮動小数点整数	有効桁 7
	double	64	倍精度浮動小数点整数	有効桁 16
文字型	char	16	文字	16 ビット文字コード
論理型	boolean	8	論理値	true,false

ちなみに String という文字列を扱うものはデータ型ではありません、String クラスというものになります。但し、扱い方が変数と同様なので、ここで学習します。

変数の宣言をしておかないとエラーとなるので注意しましょう。また、プログラムの中で使っていない変数は宣言しないようにしないと警告が出ます。しかし実行はできますから問題はありません。

宣言したからには使いたいでしょう？使うにはどうするかというと、先ほどの例でいくと以下の様に使えばよいのです。計算の詳細は次回に回します。

```
int a;      // 整数の変数  a  を定義する
a = 10;     // 変数 a  に 10 を代入する

System.out.println("変数 a に入っているのは"+ a + "です。");
```

プログラミングの世界では、「=（イコール）」は同じという意味ではありません。数学の世界と違い代入を意味します。つまり、この時点では **a という変数の中には 10 という数値が入る**ということになります。一般的に変数は定義した時点では中に何が入っているかはわからない（箱を用意しただけで中身が空かどうかも分からない）ので、初期化（中身を整理）しておくとわかりやすいでしょう。上記の例では固定値で計算するのであまり必要ではないと思うかもしれませんが、初期化する癖はつけておきましょう。

初期化のもうひとつの方法は定義と同時に行うことです。以下のようにします。

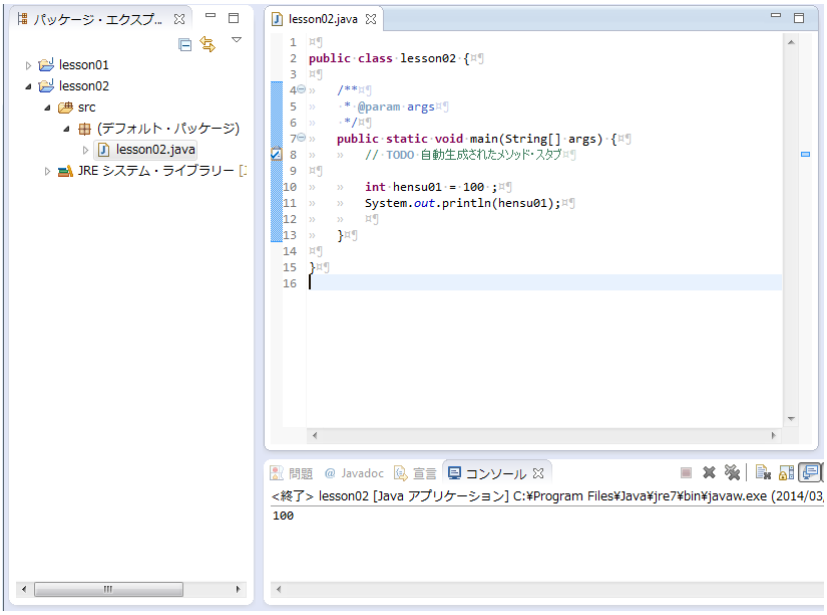
```
int a = 0 ;
```

それでは、変数 hensu01 を 100 で初期化してそれを画面に表示しましょう。

【記述例】

```
int hensu01 = 100 ;
System.out.println(hensu01) ;
```

（実行例）

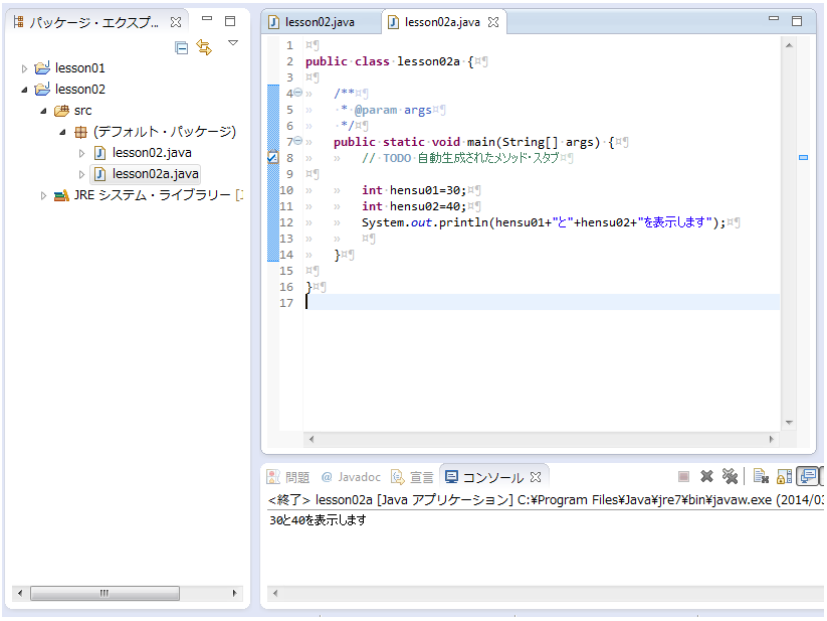


では次に、変数 hensu01 を 30 に、変数 hensu02 を 40 で初期化して、hensu01 と hensu02 の内容を表示します。

【記述例】

```
int hensu01 = 30 ;
int hensu02 = 40 ;
System.out.println(hensu01 + “と“+ hensu02 + “を表示します。” ) ;
```

（実行例）



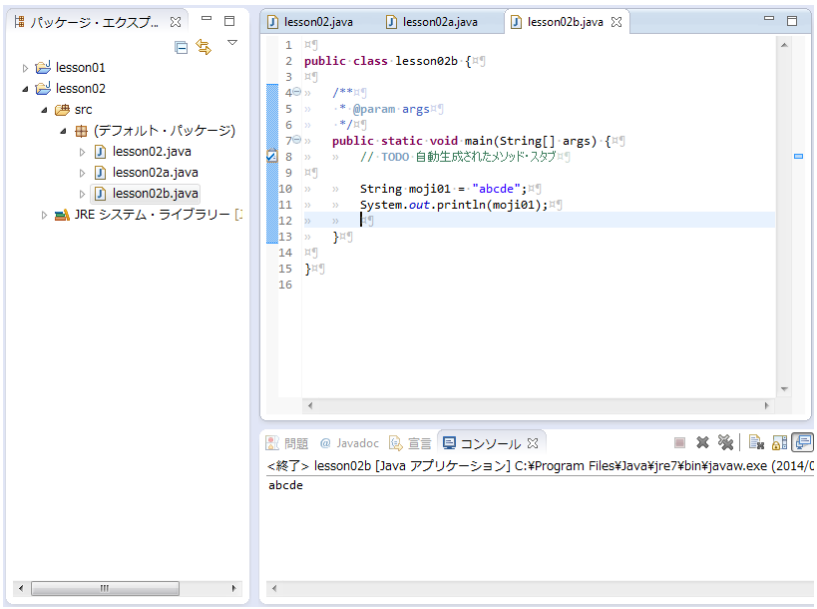
上記が変数に数値を入れた場合の扱い方です。変数は数値だけでなく文字も扱えます。文字の場合は char 型を使用し、文字列の場合は、String クラスを利用します。

変数 moji01 を文字列 abcde で初期化して、moji01 の内容を画面に表示します。

【記述例】

```
String moji01 = “abcde” ;
System.out.println(moji01) ;
```

（実行例）



変数の定義と初期化の注意点

変数を使って数字を利用する場合は、

int a = 10 ;

と表記すれば問題ありません。変数に文字を使う場合は、

char b =’x’ ;

というように対照の文字を「’ (シングルクォート)」で囲む必要があります。

文字列は、文字通り、文字が、列を成している状態を指し、Java では文字列を扱う変数がないため String クラスを利用します。その際は、

String moji01 = “abc” ;

というように「” (ダブルクォート)」で囲みます。

【練習問題】

int 型の変数　nikaime　を定義して、0 で初期化して下さい。

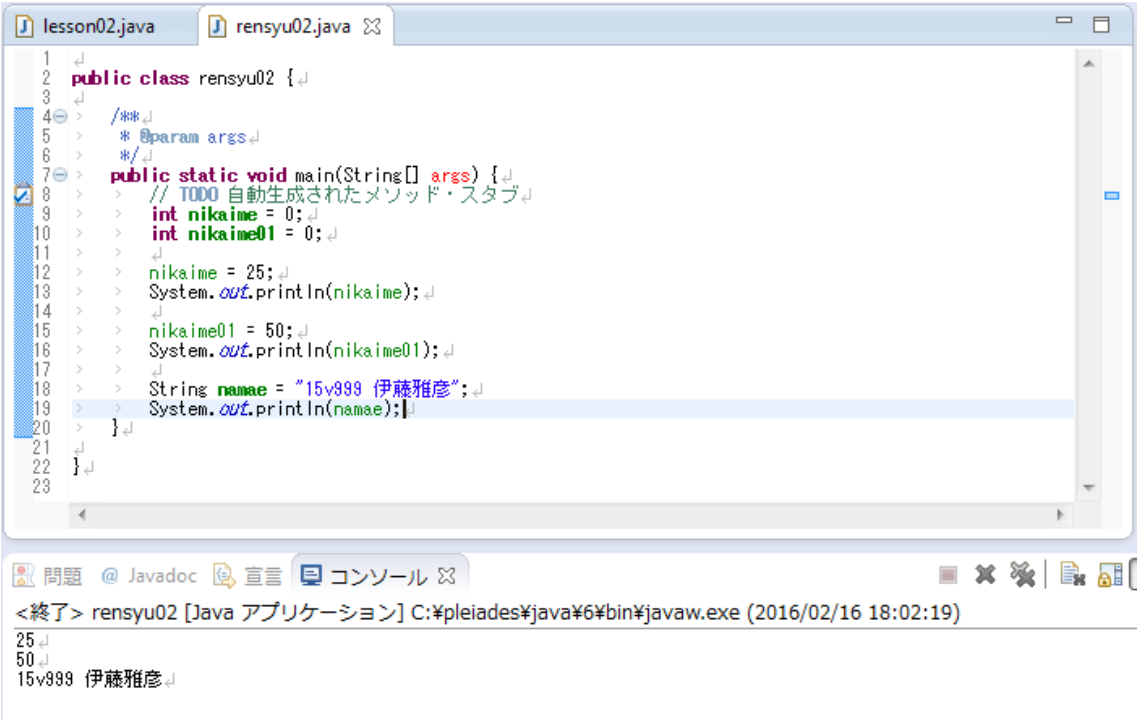
int 型の変数　nikaime01　を定義して 0 で初期化して下さい。

そして nikaime に 25 を代入し、nikaime を画面に表示して下さい。

nikaime01 に 50 を代入し、nikaime01 を画面に表示して下さい。

String クラスで何か変数を定義してその中に学籍番号と名前を入れて画面に表示して下さい。

【記述例】



Lesson3　計算をさせてみよう

1.　数値の計算

■数値の計算には以下の記号を使います。

＝	代入（比較演算子で等しいという意味ではありません）
＋	加算（たしざん）
－	減算（ひきざん）
＊	乗算（かけざん）
／	除算（わりざん）
％	剰余（わりざんのあまり）

足し算をする場合：c=a+b;
引き算をする場合：c=a-b;
掛け算をする場合：c=a*b;
割り算をする場合：c=a/b;
剰余算をする場合：c=a%b;

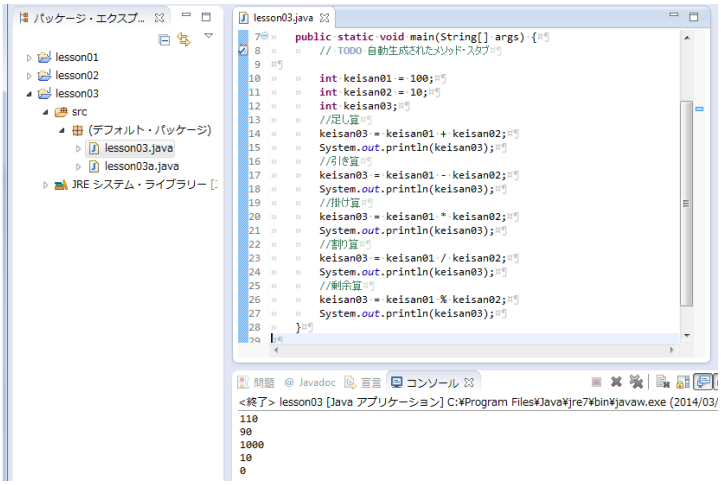
単純に1ずつ足し算する場合には　a++ と記述すれば　a=a+1 と同じになります。同様に引き算する場合も　a--と記述すれば　a=a-1 と同じになります。足し算をインクリメント、引き算をデクリメントと呼びます。

◎特に剰余は理解に苦しむかもしれません。以下の様なルールです。

例）10%3＝1　（意味）10を3で割ったらあまりが1できる。

剰余の利用はある数字が割り切れるかとかいう判断に用いる場合が多いです。

（様々な計算の例）



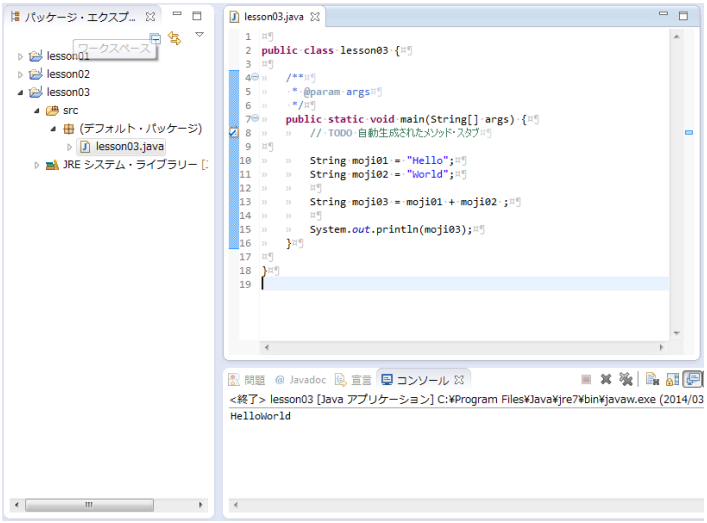
2.　文字の計算

■文字の足し算

String moji01 = ”Hello”;
String moji02 = ”World”;
String moji03 = moji01 + moji02 ;

この足し算の結果は”HelloWorld”となります。テキストをつなぐ方法です。

（文字をつなぐ例）



■要素を比較する場合の**論理演算子**は以下のものがあります

＞	より大きい	a>b　a は b より大きい
＜	より小さい	a<b　a は b より小さい
＞＝	以上	a>=b　a は b 以上である
＜＝	以下	a<=b　a は b 以下である
＝＝	等しい	a==b　a は b と等しい
！＝	等しくない	a!=b　a は b と等しくない

【練習問題】

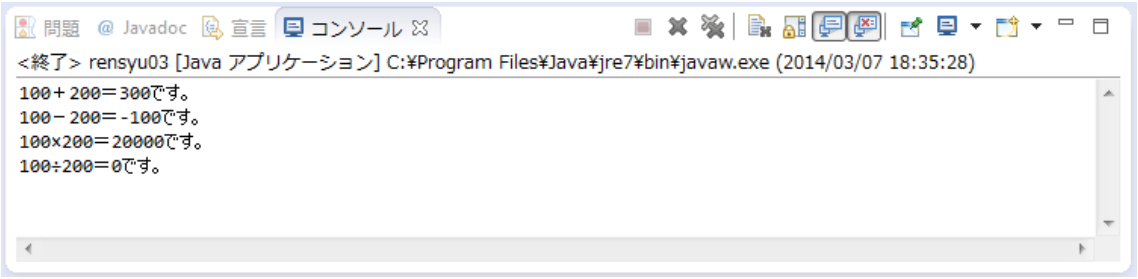
変数 a に 100 変数 b に 200 を代入してそれを四則演算して結果を表示してみよう。

処理手順（1）:int 型で変数を定義する。
処理手順（2）:ひとつの変数に 100 を代入する、もうひとつの変数に 200 を代入する。
処理手順（3）:画面表示（変数 a＋変数 b＝xxx です。）xxx には a+b の結果
処理手順（4）:画面表示（変数 a－変数 b＝xxx です。）xxx には a-b の結果
処理手順（5）:画面表示（変数 a×変数 b＝xxx です。）xxx には a*b の結果
処理手順（6）:画面表示（変数 a÷変数 b＝xxx です。）xxx には a/b の結果

【記述例】

```
public class rensyu03{
    public static void main(String args[])
    {
     int a ;
     int b ;
     int c ;
     int d ;
     int e ;
     int f ;

     a = 100 ;
     b = 200 ;
     c = a + b ;
     d = a - b ;
     e = a * b ;
     f = a / b ;

     System.out.println(a + “＋” + b + ”＝” + c + “です。” ) ;
     System.out.println(a + “－” + b + ”＝” + d + “です。” ) ;
     System.out.println(a + “×” + b + ”＝” + e + “です。” ) ;
     System.out.println(a + “÷” + b + ”＝” + f + “です。” ) ;
    }
}
```

（実行例）



Lesson4　キーボードからの入力

1.　文字列を入力する場合

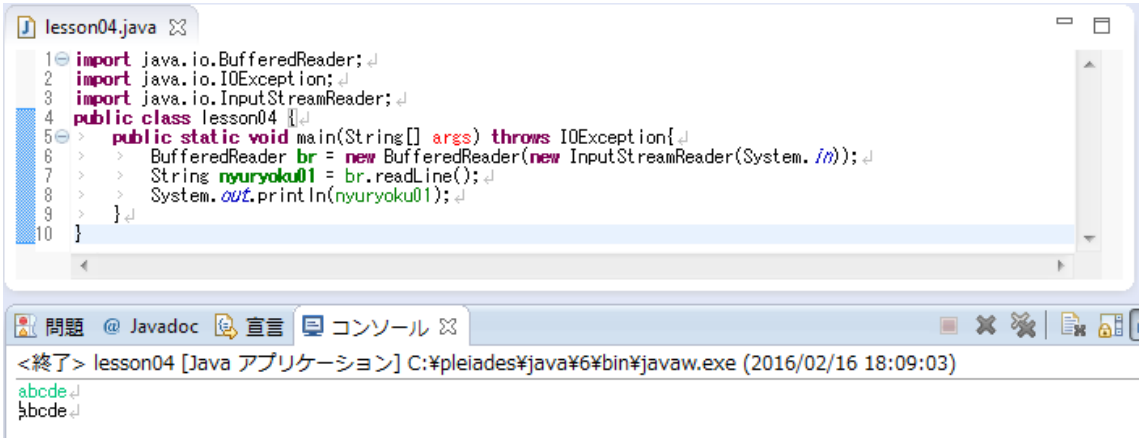
さて、今回はキーボードから数字や文字を入れる方法について。キーボードから入力する場合には以下の様にします。

【記述例】

```
import java.io.*;
public class lesson04{
   public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String nyuryoku01 = br.readLine();
        System.out.println(nyuryoku01);
    }
}
```

import 文の追加！

（実行例）



文字列の入力は入力ストリームで行われます。キーボードからの入力は System.in から入ってきます。InputStream の継承です。System.in を InputStreamReader クラスの引数としてオブジェクトを作成することで、文字ストリームに変換を行います。

＊ストリームとは・・・データの流れる道順です。
入力ストリームは、キーボード→プログラム　の矢印の部分を指します。

今回初めて出てきた import という文頭の記述ですが、これはクラスライブラリという予め用意されているクラスがあり、入出力など決まりきった処理を簡単に実現させてくれます。

- java.io はファイルの入出力などの機能を提供するものです。
- java.lang は java の基本的な機能を提供するもので省略できます。
- java.applet はアプレットの作成に必要な機能を提供します。

さらに throws も今回初めて出てきた記述になります。

Java には例外というオブジェクトが存在します。発生する可能性がある例外に対する処理を、あらかじめ記述しておくことで、プログラムがエラーで終了するのを避けられます。これは、例外が発生する可能性のあるメソッドでは、そのメソッドにおいて例外を補足するのか、そのメソッドを呼び出したメソッドに例外を任せるのかを選択する必要があります。メソッド内で例外の処理を行う場合は、「try - catch」を利用し、呼び出したメソッドに例外処理を任せる場合は「throws」を定義します。readLine() のような入出力メソッドを用いるときは、必ず「throws」か「try - catch」を使わないと、コンパイラで叱られます。

また、IOException は Stream クラスを使用している間に発生するファイル入出力関係の例外のスーパークラスになります。

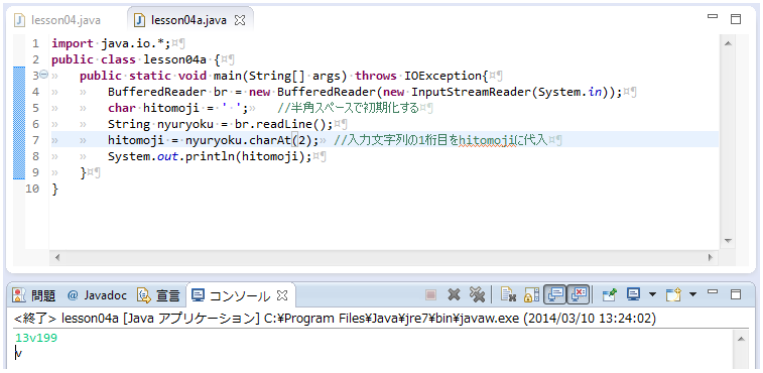
2.　文字列を入力しその何桁目かを利用する場合

入力された文字列から一文字だけを表示することも可能です。例えば、学籍番号を入力した場合三桁目にあたる学科記号だけを表示したいといった場合に利用します。

【記述例】

```
import java.io.*;
public class lesson04a{
   public static void main(String[] args) throws IOException{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        char hitomoji = ' ';          //半角のスペースで初期化する
        String nyuryoku = br.readLine();
        hitomoji = nyuryoku.charAt(2);     //入力文字列の3桁目を hitomoji  に代入
        System.out.println(hitomoji);
   }
}
```

（実行例）



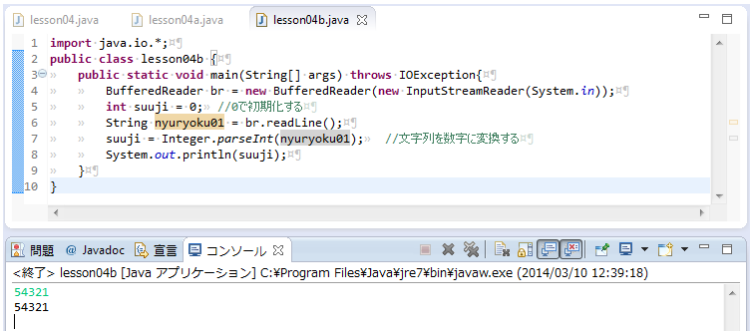
＊何桁目？の数え方：　0,1,2,3,4、・・・・9　つまり 0 が 1 桁目になる。

3.　入力された文字列を int 型に変換し整数として利用する場合

【記述例】

```
import java.io.*;
public class lesson04b {
   public static void main(String[] args) throws IOException{
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int suuji = 0;       //int 型の変数を用意し 0 で初期化する
      String nyuryoku01 = br.readLine();
      suuji = Integer.parseInt(nyuryoku01);       //文字列を数字に変換する
      System.out.println(suuji);
   }
}
```

（実行例）



＊初めの１桁に「0」を入力した場合は無視されます。これは、整数の先頭に「0」がつくことはないからです。

【練習問題】

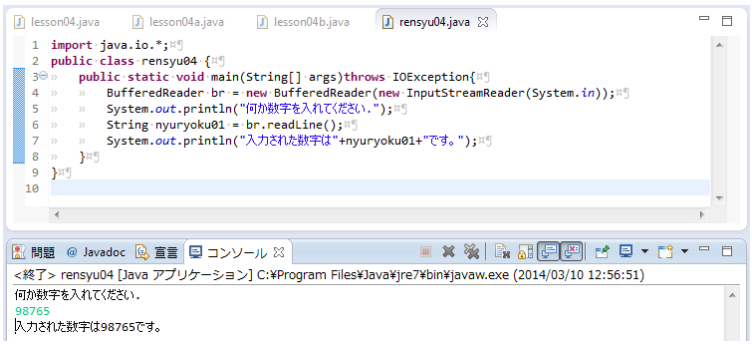
画面に「何か数字を入れてください」と表示して、キーボードから数字を入力させ、その入力された数字を再度画面に表示する。

処理手順（1）：画面に「何か数字を入れてください」と表示する。
処理手順（2）：キーボードから数字を入力する。
処理手順（3）：入力された数字を画面に表示する。
「入力された数字は XX です」

【記述例】

```
import java.io.*;
public class rensyu04{
   public static void main(String[] args) throws IOException{
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("何か数字を入れてください。");
        String nyuryoku01 = br.readLine();
        System.out.println("入力された数字は" + nyuryoku01 + "です。");
    }
}
```

（実行例）



Lesson5　条件分岐を学ぼう

　条件分岐はプログラムにおける重要な要素です。条件分岐がなければ単純な処理をするだけで、面白くもなんともありません。

　世の中には様々なケースで判断を迫られるケースがあります。ごく身近なところでは履修登録などの場合。

まず

「その科目が必須かどうか？」
「必須なら仕方ないから履修登録しよう」
「必須でないなら内容を確認しよう」
「内容はおもしろそうか？」
「おもしろそうなら履修登録しよう」
「おもしろくなさそうなら登録はやめておこう」

という風に判断をして履修登録をしますよね？まさか、手当たり次第に登録して取れたら儲けものなどというやり方はしてないでしょうね。

これを Java 言語風に書くとこうなります。

```
if(科目==必須){
　　履修登録;
} else if(内容==面白い){
　　履修登録;
}else {
　　登録見送り;
}
```

上の文章と見比べてください。条件分岐とはこういうものなのです。

1． if 単体の場合

まず、**if 文**
「もし～ならば～をする、でなければ～をする」の様に表現するものです。

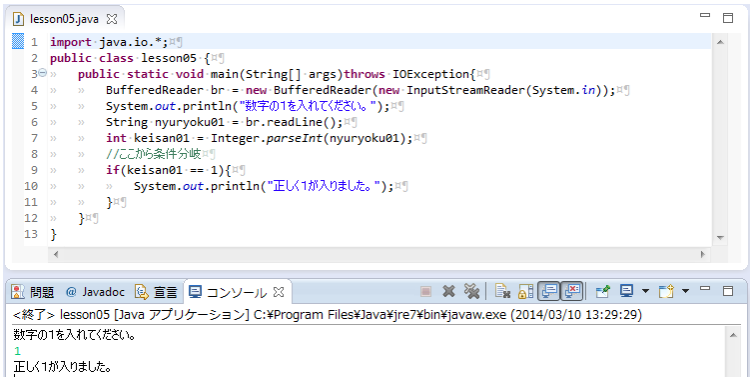
if（演算式）
{
演算式の値が、真の場合に実行される命令;
}

if で条件を設定して、それが OK ならば以降の処理をするものです。
具体的な記述例を挙げて見ます。キーボードから数字の1を入力させて、正しく1が打たれたかどうかを判断します。

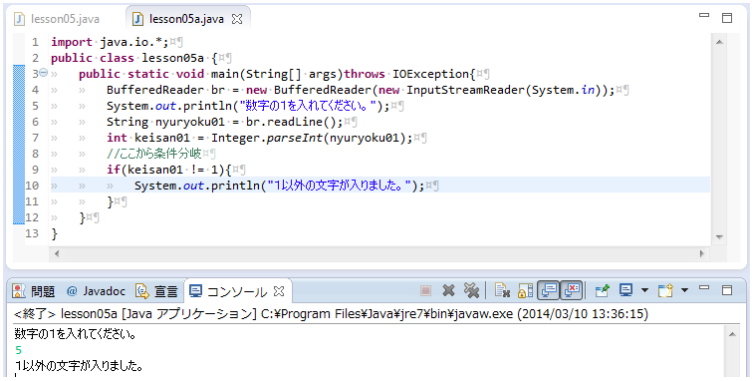
【記述例】

```
import java.io.*;
public class lesson05{
   public static void main(String[] args) throws IOException{
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("数字の１を入れてください。");
        String nyuryoku01 = br.readLine();
        int keisan01 = Integer.parseInt(nyuryoku01) ;
        if( keisan01 == 1 ){
              System.out.println("正しく１が入りました。");
        }
   }
}
```

（実行結果）



否定の場合には==のところを!=にすれば良いので試してみてください。



ここで注意することは、前回の LESSON でやった文字列から整数への変換をしなければいけません。それが、「int keisan01 = Integer.parseInt(nyuryoku01) ;」になります。

★普通に考えれば当たり前なのですが、コンピュータの処理はいちいち正しいか、そうでないかを判断しないといけません。

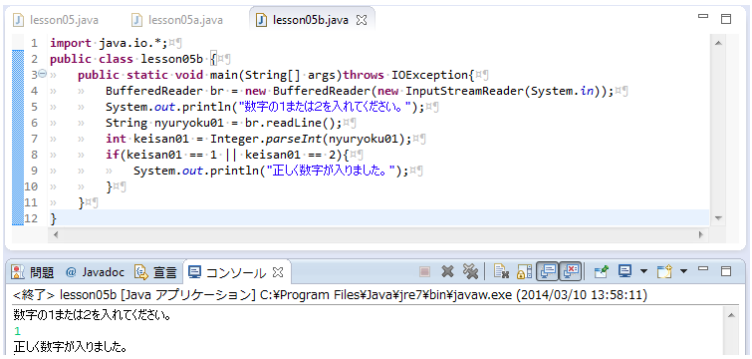
どうでしょうか？ここが理解できなければこの後のパターンは難しいので、しっかり理解出来るようにしましょう。

次に、複数の条件の場合。

【記述例】

```
import java.io.*;
public class lesson05b{
   public static void main(String[] args) throws IOException{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("数字の１または２を入れてください。");
        String nyuryoku01 = br.readLine();
        int keisan01 = Integer.parseInt(nyuryoku01) ;
        if( keisan01 ==1 || keisan01==2 ){
             System.out.println("正しく数字が入りました。");
        }
    }
}
```

（実行結果）



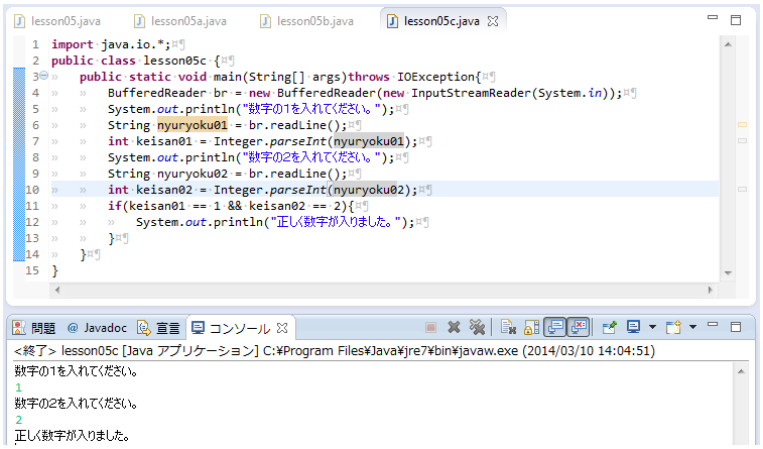
if 文の条件の対象が「1**または**2」などの場合には「 || 」でつなぎます。
また、if 文の条件の対象が「1**かつ**2」などの場合には「 && 」でつなぎます。

それが以下の文です。

【記述例】

```
import java.io.*;
public class lesson05c{
   public static void main(String[] args) throws IOException{
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("数字の１を入れてください。");
      String nyuryoku01 = br.readLine();
      int keisan01 = Integer.parseInt(nyuryoku01) ;
      System.out.println("次に数字の２を入れてください。");
      String nyuryoku02 = br.readLine();
      int keisan02 = Integer.parseInt(nyuryoku02) ;
      if( keisan01 ==1 && keisan02==2 ){
         System.out.println("正しく数字が入りました。");
      }
   }
}
```

（実行結果）



2. if-else　形式の場合

else は条件が成立しない場合の処理を示します。

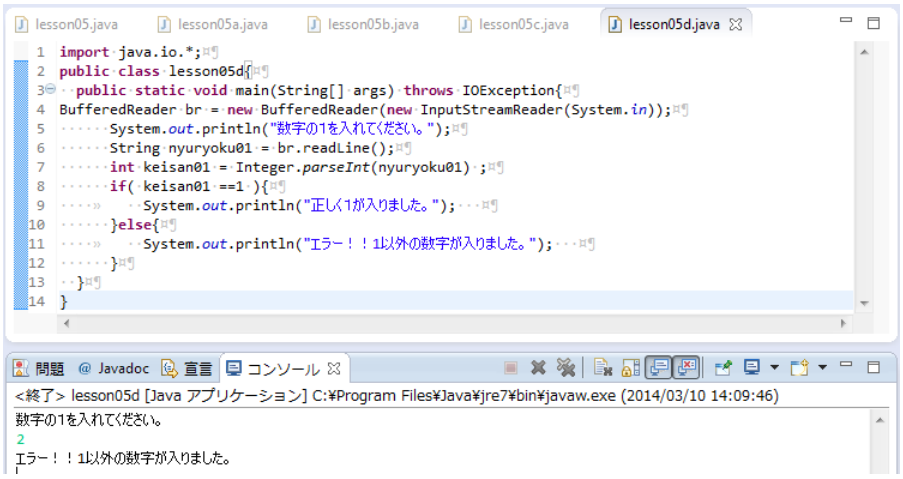
```
if(演算式)
{
    演算式の値が、真の場合に実行される命令;
}
else
{
    演算式の値が、偽の場合に実行される命令;
}
```

ではこれも具体例を挙げてみます、キーボードから入った数字が１以外の場合にエラー表示をするようにします。

【記述例】

```
import java.io.*;
public class lesson05d{
  public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("数字の１を入れてください。");
        String nyuryoku01 = br.readLine();
        int keisan01 = Integer.parseInt(nyuryoku01) ;
        if( keisan01 ==1 ){
            System.out.println("正しく１が入りました。");
        }else{
            System.out.println("エラー！！1 以外の数字が入りました。");
        }
  }
}
```

（実行結果）


lesson05.java
lesson05a.java
lesson05b.java
lesson05c.java
lesson05d.java
1 import java.io.*;
2 public class lesson05d{
3 public static void main(String[] args) throws IOException{
4 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
5 System.out.println("数字の1を入れてください。");
6 String nyuryoku01 = br.readLine();
7 int keisan01 = Integer.parseInt(nyuryoku01) ;
8 if(keisan01 ==1){
9 System.out.println("正しく1が入りました。");
10 }else{
11 System.out.println("エラー！！1以外の数字が入りました。");
12 }
13 }
14 }
問題
Javadoc
宣言
コンソール
<終了> lesson05d [Java アプリケーション] C:¥Program Files¥Java¥jre7¥bin¥javaw.exe (2014/03/10 14:09:46)
数字の1を入れてください。
2
エラー！！1以外の数字が入りました。

3. if-else if 形式の場合

ここはさらに複雑になります。条件がふたつ以上の場合にこのような記述を行います。

```
if(演算式1){
    演算式1の値が、真の場合に実行される命令;
}else if(演算式2){
    演算式1の値が偽であり、かつ演算式2の値が真の場合に実行される命令;
}else {
    演算式1の値が偽であり、かつ演算式2の値が偽の場合に実行される命令;
}
```

例えば数字の 1 が入力されたら OK ですが、それ以外にも数字の 3 が入力されても OK というような使い方をするのです。

最初の if で条件が合致しなくても次の else if で設定した条件に合致すれば OK というものです。今回の例は A または B という条件を満たすものを挙げてみました。

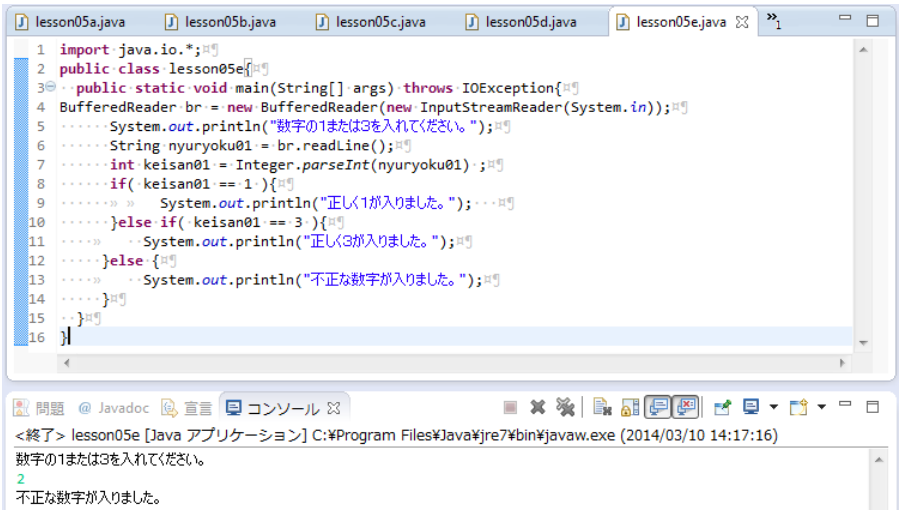
このあたりになると中かっこ{ の数のアンマッチで「予期しない EOF・・・」というエラーメッセージに悩む人が増えてくるのも事実ですね。

はじまりの中カッコ{ の数と終わりの中カッコ }の数を数えてみてくださいね。絶対に同じでなければいけませんよ。

では具体例を示します。

```
import java.io.*;
public class lesson05e{
   public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("数字の１または３を入れてください。");
        String nyuryoku01 = br.readLine();
        int keisan01 = Integer.parseInt(nyuryoku01) ;
        if( keisan01 == 1 ){
              System.out.println("正しく１が入りました。");
        }else if( keisan01 == 3 ){
             System.out.println("正しく３が入りました。");
       }else {
             System.out.println("不正な数字が入りました。");
       }
   }
}
```

（実行結果）



4. 文字列と文字列の比較

さて、ここまでの話では一文字、数字の比較を行ってきました。でも文字列同士の比較、例えば「Computer」と入れた文字が正しく「Computer」と入ったかどうかをチェックする場合には、単純な比較をするだけでは、アンマッチになります。それを解決するには以下の様な手順で記述する必要があります。

```
if(文字列1.equals(文字列2)){
    真の場合の処理
}
else{
    偽の場合の処理
}
```

この他にも文字列の比較には以下のものがあります。

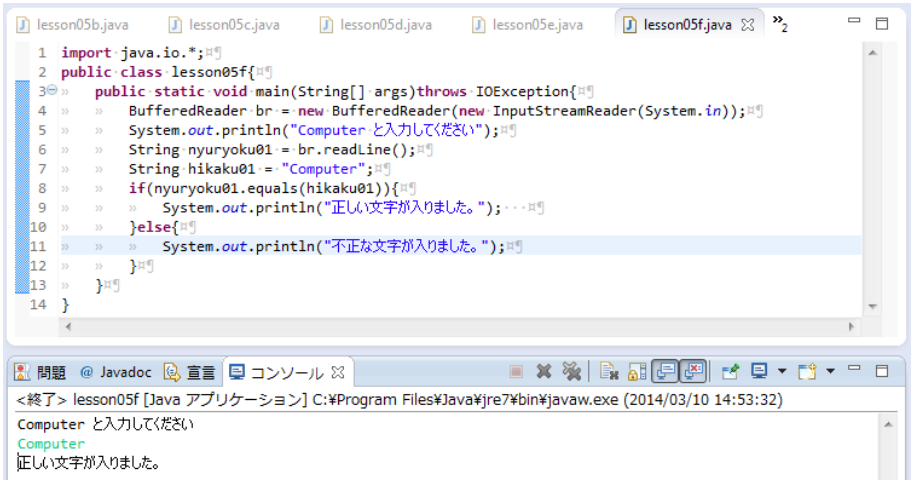
等しいかどうかを調べる	equals,equalsIgnoreCase	if(str1.equals(str2))
文字列長を得る	length	str.length()
文字列の大小を比較する	compareTo, compareToIgnoreCase	str1.compareTo(str2)==0
n 番目の文字を得る	charAt(n)	str.charAt(0)
指定文字が出現する場所を得る	indexOf('指定文字')	str.indexOf('a')

※IgnoreCase は大文字・小文字を無視して比較する場合に用いる

【記述例】

```
import java.io.*;
public class lesson05f{
   public static void main(String[] args)throws IOException{
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Computer と入力してください");
      String nyuryoku01 = br.readLine();
      String hikaku01 = "Computer";
      if(nyuryoku01.equals(hikaku01)){
      System.out.println("正しい文字が入りました。□□");
      }else{
      System.out.println("不正な文字が入りました。□□");
      }
   }
}
```

（実行例）



※漢字を扱う場合には、以下の設定を、ソースを打ち込む前にして下さい。

java ソースファイルを右クリックしてプロパティーウィンドウ開く。

ウィンドウの左側でリソースを選ぶ。

テキストファイルのエンコードで「その他」を選択し、「MS932」を選ぶ。OK ボタンを押して設定を完了する。

メニューのウィンドウ→設定→一般→ワークスペース→MS932 で変更する（コンソールの表示画面で漢字表示）

【練習問題】

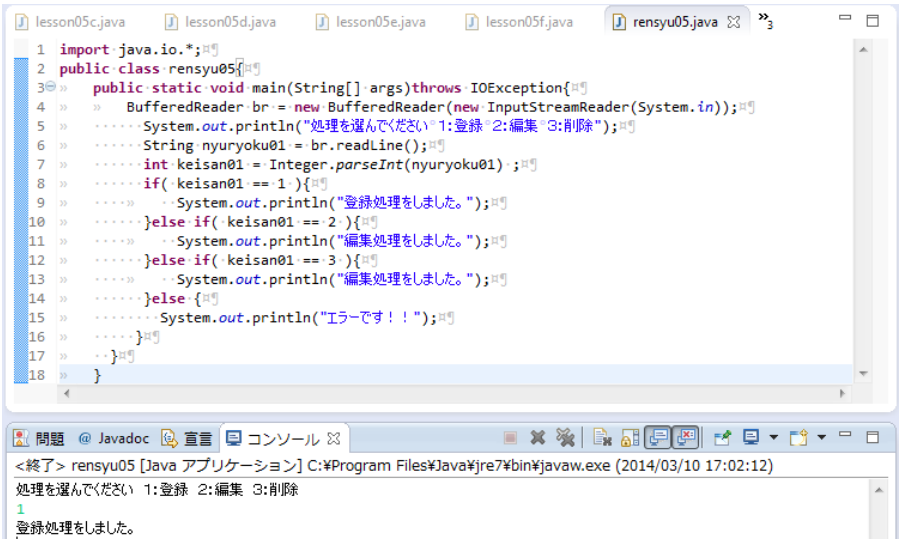
画面に「処理を選んでください　1：登録　2：編集　3：削除」と表示して、キーボードから数字を入力させ、その入力された数字が正しければそれぞれの入力処理をしましたと表示し、それ以外ならエラーを表示してください。

処理手順（1）：int 型の変数をひとつ定義する。
処理手順（2）：画面に以下のメッセージを表示する
「処理を選んでください　1：登録　2：編集　3：削除」
処理手順（3）：キーボードから数字を入力する。
処理手順（4）：入ってきた数字をチェックする。
1の時「登録処理をしました」と表示
2の時「編集処理をしました」と表示
3の時「削除処理をしました」と表示
それ以外の時「エラーです。」と表示

【記述例】

```
import java.io.*;
public class rensyu05{
   public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("処理を選んでください　１：登録　２：編集　３：削除");
        String nyuryoku01 = br.readLine();
        int keisan01 = Integer.parseInt(nyuryoku01) ;
        if( keisan01 == 1 ){
             System.out.println("登録処理をしました。");
        }else if( keisan01 == 2 ){
             System.out.println("編集処理をしました。");
        }else if( keisan01 == 3 ){
             System.out.println("編集処理をしました。");
        }else {
           System.out.println("エラーです！！");
       }
   }
}
```

（実行結果）



Lesson6　複数の条件分岐

複数の条件をチェックするのは if 文でもできます。しかしあまり条件の数が多くなると記述するだけでも大変なものです。そこで switch 文です、これは条件分岐を比較する式が1つだけで、式の値が int 型でなければならないという制約条件があります。break は switch 文を抜けるための命令。default はそこまでの条件に合致しないものについて実行するものです。

条件分岐にはいくつかのケースがあります。単一の条件ではひとつの if 文で完結します。

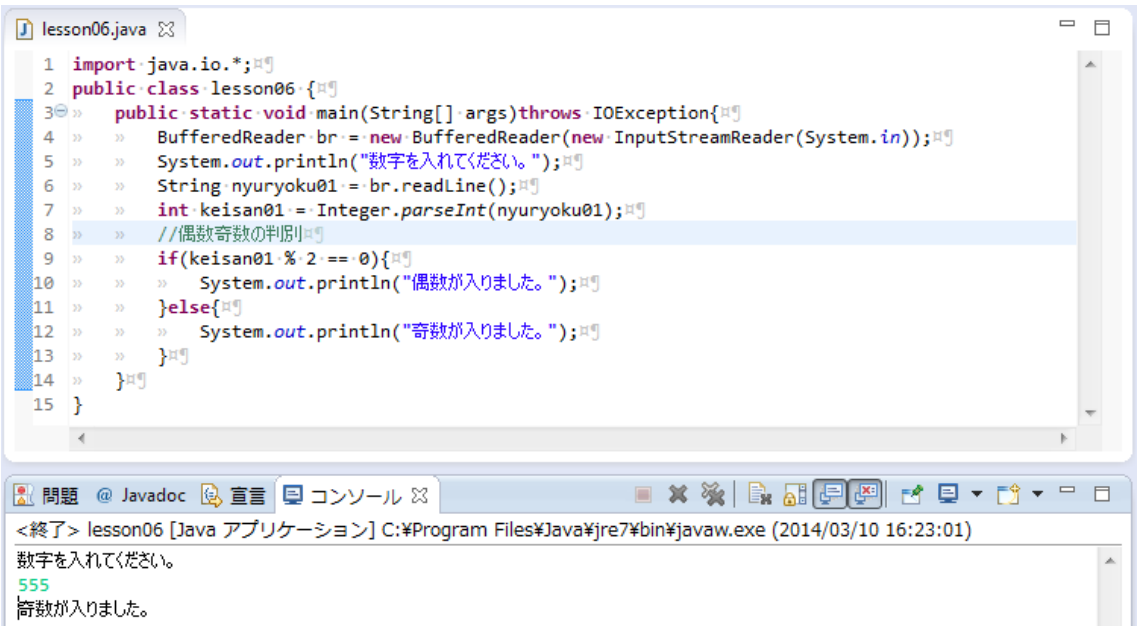
「入ってきた数字は1である（y/n）」のような場合、この条件分岐は単一の条件分岐であるといいます。しかし、こんな場合はどうでしょう？「入ってきた数字は偶数である」これは yes-no の関係ではありませんね。しかも偶数の判断をする必要があるというと、2か？4か？6か？8か？と聞く（もっと簡単な方法が実際にはありますが・・・）必要がありますね。そんなとき、いちいち YES-NO 形式でひとつひとつ聞くのはどうでしょう？大変ですよね 。ですから今回学習する一度に複数の条件を満たすかどうかを判断する命令が必要なのです。

ちなみに、偶数かどうかを判断するには以下のようなプログラムを組めばいいです。やってみましょう。

【記述例】

```
import java.io.*;
public class lesson06{
   public static void main(String[] args) throws IOException{
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("数字を入れてください。");
      String nyuryoku01 = br.readLine();
      int keisan01 = Integer.parseInt(nyuryoku01) ;
      if( keisan01 % 2 == 0 ){
            System.out.println("偶数が入りました。");
      }else {
            System.out.println("奇数が入りました。");
      }
   }
}
```

（実行結果）



■パターン1（それぞれのケースで処理が完結する）

switch(式) {
case 値 1: 式の値が、値1と等しい場合に実行される命令;
　　break;
case 値 2: 式の値が、値1と等しくなく、かつ値2と等しい場合に実行される命令;
　　break;
case 値 3:式の値が、値1と値2に等しくなく、かつ値3と等しい場合に実行される命令;
　　break;
default:式の値が、値1～値3のいずれとも等しくない場合に実行される命令;
}

■パターン2（全てのケースを該当すれば実行する：OR の場合）

switch(式){
case 値 1:式の値が、値1と等しい場合に実行される命令;
case 値 2:式の値が、値1または値2と等しい場合に実行される命令;
case 値 3:式の値が、値1または値2または値3と等しい場合に実行される命令;
　　break;
default:式の値が、値1～値3のいずれとも等しくない場合に実行される命令;
}

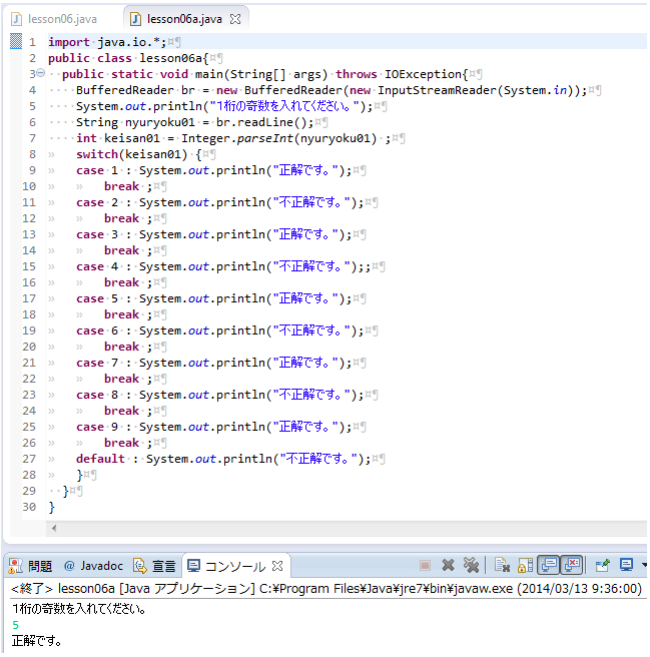
の様に break の有無で意味が異なるので注意が必要です。

具体例を以下に示します。打ち込んで実行してみましょう。1桁の奇数を判断するプログラムです。これは上記パターン1で作りました

【記述例】

```
import java.io.*;
public class lesson06a{
   public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   System.out.println("１桁の奇数を入れてください。");
   String nyuryoku01 = br.readLine();
   int keisan01 = Integer.parseInt(nyuryoku01) ;
      switch(keisan01) {
      case 1 : System.out.println("正解です。");
            break ;
      case 2 : System.out.println("不正解です。");
            break ;
      case 3 : System.out.println("正解です。");
            break ;
      case 4 : System.out.println("不正解です。");;
            break ;
      case 5 : System.out.println("正解です。");
            break ;
      case 6 : System.out.println("不正解です。");
            break ;
      case 7 : System.out.println("正解です。");
            break ;
      case 8 : System.out.println("不正解です。");
            break ;
      case 9 : System.out.println("正解です。");
            break ;
      default : System.out.println("不正解です。");
      }
   }
}
```

（実行結果）

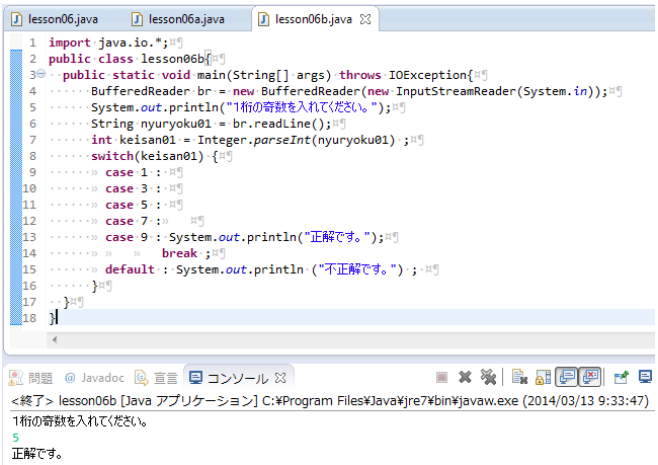

lesson06.java
lesson06a.java
1 import java.io.*;
2 public class lesson06a{
3 public static void main(String[] args) throws IOException{
4 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
5 System.out.println("1桁の奇数を入れてください。");
6 String nyuryoku01 = br.readLine();
7 int keisan01 = Integer.parseInt(nyuryoku01) ;
8 switch(keisan01) {
9 case 1 : System.out.println("正解です。");
10 break ;
11 case 2 : System.out.println("不正解です。");
12 break ;
13 case 3 : System.out.println("正解です。");
14 break ;
15 case 4 : System.out.println("不正解です。");;
16 break ;
17 case 5 : System.out.println("正解です。");
18 break ;
19 case 6 : System.out.println("不正解です。");
20 break ;
21 case 7 : System.out.println("正解です。");
22 break ;
23 case 8 : System.out.println("不正解です。");
24 break ;
25 case 9 : System.out.println("正解です。");
26 break ;
27 default : System.out.println("不正解です。");
28 }
29 }
30 }
問題
Javadoc
宣言
コンソール
<終了> lesson06a [Java アプリケーション] C:¥Program Files¥Java¥jre7¥bin¥javaw.exe (2014/03/13 9:36:00)
1桁の奇数を入れてください。
5
正解です。

【記述例】

次に同じ処理をパターン2で実現したものを示します。違いをしっかり理解してくださいね。処理結果は同じになります。

```
import java.io.*;
public class lesson06{
   public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("１桁の奇数を入れてください。");
        String nyuryoku01 = br.readLine();
        int keisan01 = Integer.parseInt(nyuryoku01) ;
        switch(keisan01) {
           case 1：
           case 3：
           case 5：
           case 7：
           case 9：System.out.println("正解です。");
                    break；
           default：System.out.println ("不正解です。") ;
        }
   }
}
```

（実行結果）



どうですか？同じことをするのにもいろいろな方法があるということを理解できましたか？では次に、絶対パターン1でしか実現できない処理の例を示します。1が入ってきたら登録処理、2が入ってきたら更新処理、3が入ってきたら削除処理という風にそれぞれの入力された値に意味が

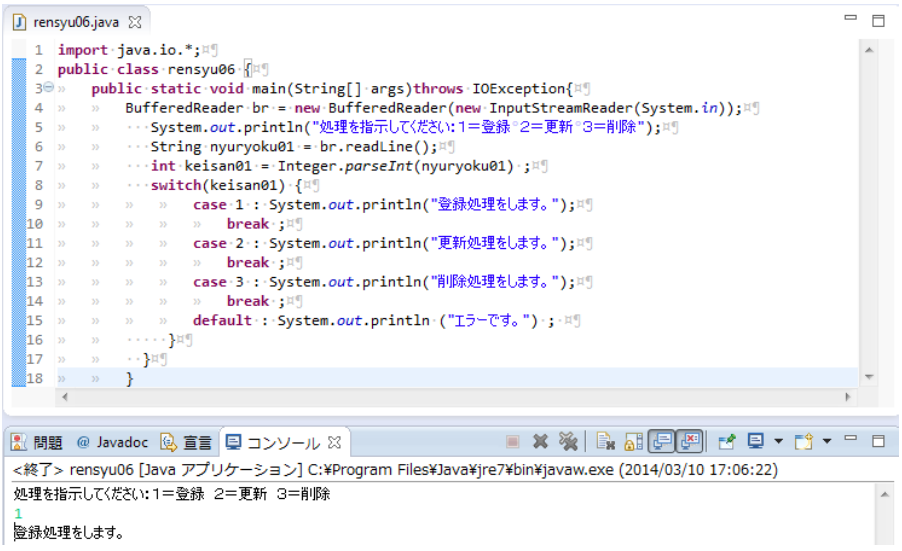
あるというケースです。

【記述例】

```
import java.io.*;
public class lesson06{
   public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("処理を指示してください：１＝登録　２＝更新　３＝削除");
    String nyuryoku01 = br.readLine();
    int keisan01 = Integer.parseInt(nyuryoku01) ;
    switch(keisan01) {
          case 1 : System.out.println("登録処理をします。");
               break ;
          case 2 : System.out.println("更新処理をします。");
               break ;
          case 3 : System.out.println("削除処理をします。");
               break ;
          default : System.out.println ("エラーです。") ;
      }
   }
}
```

これは、パターン2では表現できません。しっかり使い分けをしましょう。

（実行結果）



【練習問題】

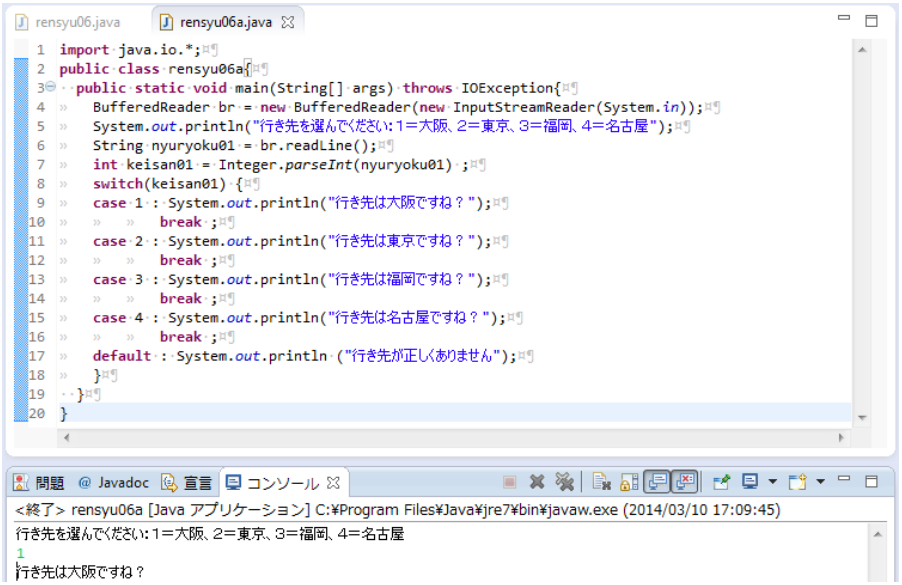
画面に「行き先を選んでください：１＝大阪、２＝東京、３＝福岡、４＝名古屋」と表示させてそれぞれの数字に対応したメッセージを出してください。表示は例えば1が入ったならば「行き先は大阪ですね？」とします。１～４以外が入った場合には「行き先が正しくありません」と表示してください。

処理手順（1）：int 型の変数を定義する。
処理手順（2）：画面に行き先を選んでください・・・・メッセージを表示
処理手順（3）：キーボードから数字を入力させる。
処理手順（4）：条件分岐で判断をする。

【記述例】

```
import java.io.*;
public class rensyu06a{
   public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println
("行き先を選んでください：１＝大阪、２＝東京、３＝福岡、４＝名古屋");
    String nyuryoku01 = br.readLine();
    int keisan01 = Integer.parseInt(nyuryoku01) ;
    switch(keisan01) {
          case 1：System.out.println("行き先は大阪ですね？");
                  break ;
          case 2：System.out.println("行き先は東京ですね？");
                  break ;
          case 3：System.out.println("行き先は福岡ですね？");
                  break ;
          case 4：System.out.println("行き先は名古屋ですね？");
                  break ;

          default：System.out.println ("行き先が正しくありません") ;
      }
  }
}
```

（実行結果）



今回はここまでです。

Lesson7　繰り返し処理（カウンター）

繰り返し処理は条件分岐と共に非常に大切な処理です。何回か同じ事を繰り返す時に使います。繰り返し処理をカウンターにて実現するのが for 文です。これは、まずカウンターとなる変数をスタート値からエンド値まで繰り返しをするもので、これを使う場合にはカウンターの変数を初期化する必要がありませんから便利です。

```
for（式1； 式2； 式3）
{
      繰り返し実行される命令;
}
```

式1：繰り返しのはじめに1度だけ実行される。
式2：繰り返しの継続条件。
式3：1回の繰り返しの終わりに毎回実行される。

例）処理のみ記述（実行するには Class 等の指定が必要）

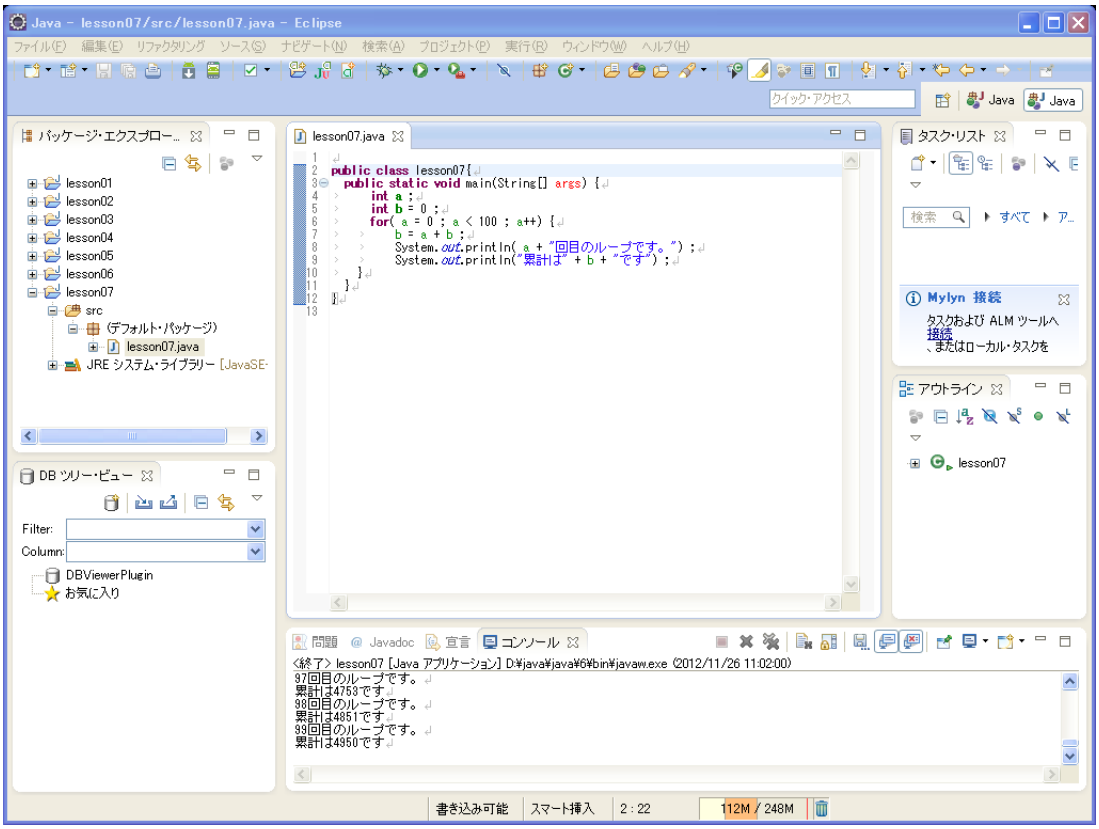
```
int  i;    // 初期化しなくていい。
for( i=0 ; i < 10 ; i++ )
{
      System.out.println ("繰り返し");
}
```

繰り返し処理をループ処理とも言います。式1にはたいていの場合初期値の設定、式2には継続のための演算子、式3にはループ時のカウンターの設定を行うものです。
ループを途中で抜ける場合には break;を、それ以降の行を実行せずに次の繰り返し処理を継続するには continue;を実行します。では、実際の例を挙げてみます。
【記述例】

```
public class lesson07{
   public static void main(String[] args) {
        int a ;
        int b = 0 ;
        for( a = 0 ; a < 100 ; a++) {
             b = a + b ;
             System.out.println( a + "回目のループです。") ;
             System.out.println("累計は" + b + "です") ;
     }
   }
}
```

この例では、0 から 99 までの整数をループ処理で累計していく処理です。

（実行例）



表示（桁数）を整えるには下記の記述方法を試すといいでしょう

System.out.print(“カウンター”);	改行しない表示
System.out.print(String.format(“%4d”,count01));	表示を整える
System.out.print(“¥t”);	¥t はタブ

では、次に練習問題。

【練習問題】

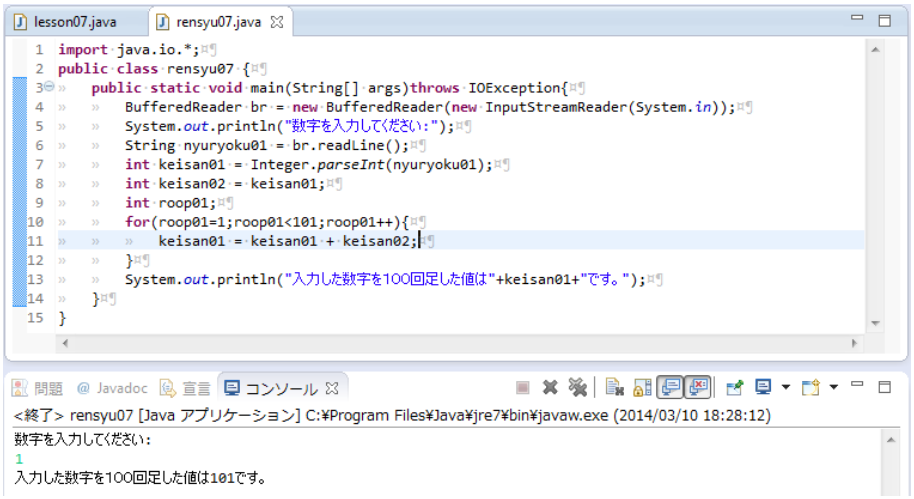
画面に「数字を入力してください：」と表示させ、入力された値を100回足し算した結果を画面に表示しましょう。

処理手順（1）：int 型の変数を定義する。
処理手順（2）：画面に「数字を入力してください：」とメッセージを表示する。
処理手順（3）：キーボードから数字を入力させる。
処理手順（4）：カウンターを用いて1から100までループする。その中で入力された値を足し算する。
処理手順（5）：ループを抜けたら最終的な数値を表示する。

【記述例】

```
import java.io.*;
public class rensyu07 {
	public static void main(String[] args)throws IOException{
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		System.out.println("数字を入力してください:");
		String nyuryoku01 = br.readLine();
		int keisan01 = Integer.parseInt(nyuryoku01);
		int keisan02 = keisan01;
		int roop01;
		for(roop01=1;roop01<101;roop01++){
			keisan01 = keisan01 + keisan02;
		}
		System.out.println("入力した数字を１００回足した値は"+keisan01+"です。");
	}
}
```

（実行例）



Lesson8　カウンター以外の繰り返し処理

さて、前回カウンターによるループを学習しましたが、カウンターでのループには制約事項があります。まず、カウンターの終了条件。a<100 のように定数を用いる場合が多いのです。しかしプログラムでは定数で 10 回とか 100 回という終了条件以外での終わり方も多々あります。そんな時に用いるのが今回学習する while 文などによるカウンターに頼らない終了条件というものなのです。では、具体的な説明をしましょう。

繰り返し処理の場合、始まりはそのループに入れば実行します。でも終わり方は様々です。例えばデータをループ毎に入力させて、何か終了のしるしが入ってくれば終わる。ゲームなどで再チャレンジ？(y/n)と表示して n が入ったらおしまいということを実現するには以下のようにします。

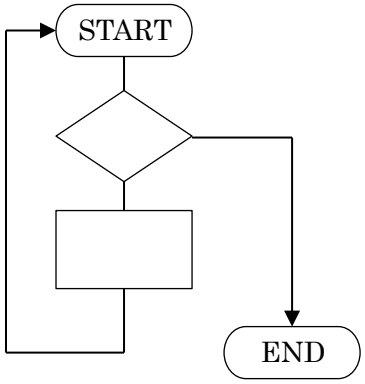
【記述例】

```
char a = 'y' ;
while( a != 'n' ) {
      ゲームの処理 ;
      System.out.println("再チャレンジ？(y/n) : ") ;
      String nyuryoku01 = br.readLine();
            a = nyuryoku01.charAt(0) ;       //nyuryoku01 の 1 桁目を取り出す
}
```

まあ、いきなり while(a!='n') と書かれてもわからないと思います。これは変数 a の値が n 以外の場合処理を繰り返すという意味なのです。ゲームなどによく応用しますが、それ以外でも頻繁に用いますのでこれの使い方をしっかり学習しましょう。では、文法など。

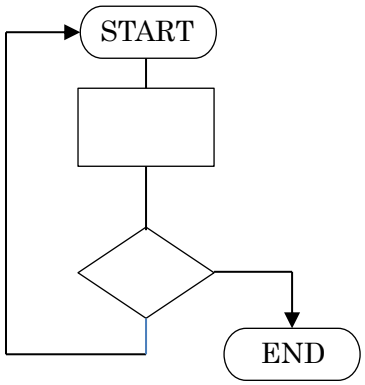
■while 文は式の値が真である間繰り返すものです。

```
while(式){
        繰り返し実行される命令;
}
```



■以下の様にすると最低1回は命令が実行されます（while の式が処理後に評価される）。

```
do{
        繰り返し実行される命令;
} while(式);
```



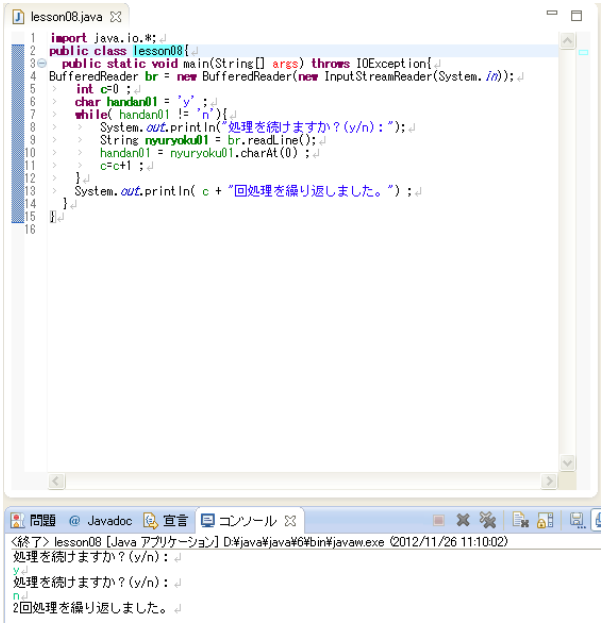
では、実際の利用例を記述してみましょう。

まずは while を使った方法。

【記述例】

```
import java.io.*;
public class lesson08 {
   public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int c=0 ;
      char handan01 = 'y' ;
      while( handan01 != 'n'){
            System.out.println("処理を続けますか？(y/n) : ");
            String nyuryoku01 = br.readLine();
            handan01 = nyuryoku01.charAt(0) ;
            c=c+1 ;
      }
      System.out.println( c + "回処理を繰り返しました。") ;
   }
}
```

（実行結果）

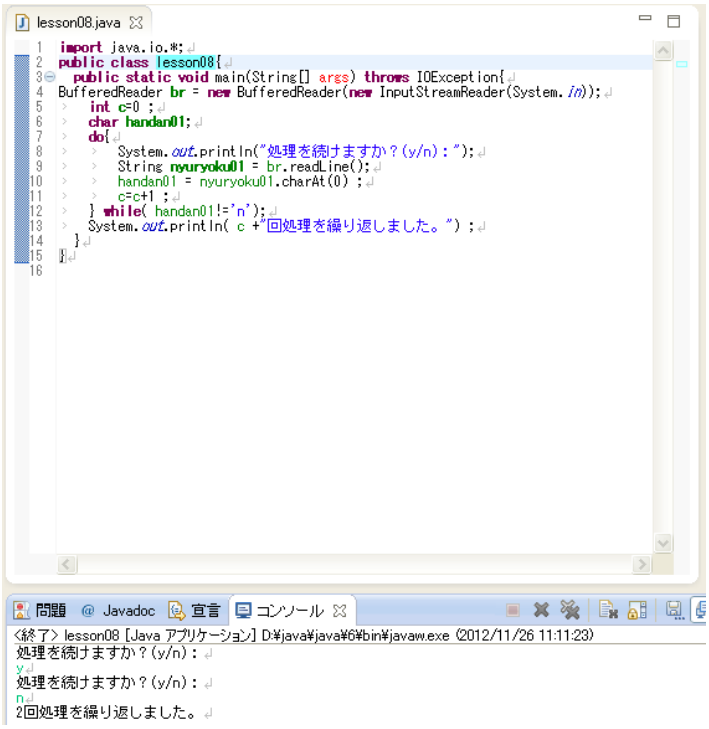


次に do-while を使った方法。全く同じ処理をしてみます。

【記述例】

```
import java.io.*;
public class lesson08{
   public static void main(String[] args) throws IOException{
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int c=0 ;
      char handan01;
      do{
            System.out.println("処理を続けますか？(y/n) : ");
            String nyuryoku01 = br.readLine();
            handan01 = nyuryoku01.charAt(0) ;
            c=c+1 ;
      } while( handan01!='n');
      System.out.println( c +"回処理を繰り返しました。") ;
   }
}
```

（実行結果）



上記2つの処理は見かけ上は同じ結果を表示します。しかし厳密に言えば異なる処理をしているのです。do-while は、一度は必ずループ内の処理をします。while はもしキーボードからの取り込みがループ外にあったら一度も処理されないというケースもあるのです。その違いはしっかり理解しておきましょう。

では練習問題へ進みましょう。

【練習問題】

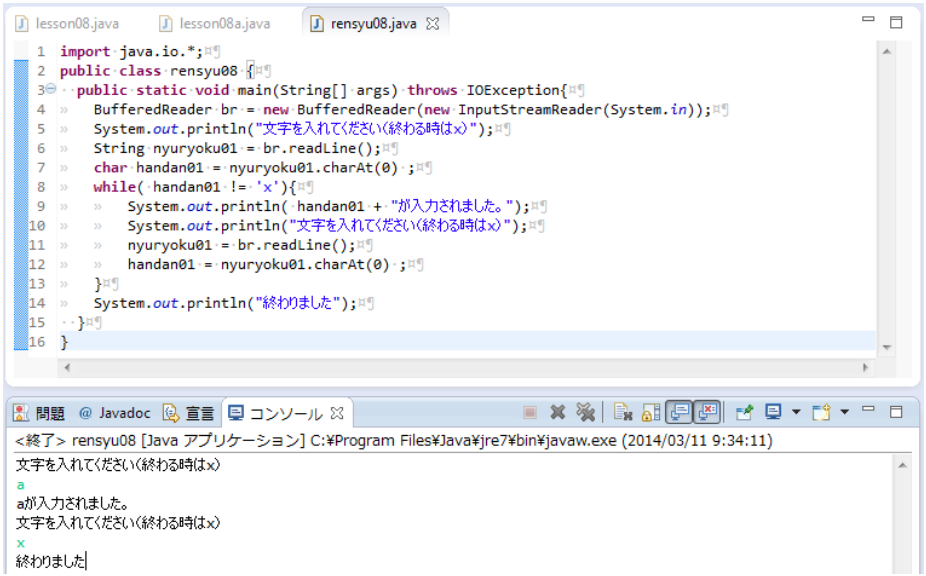
文字をキーボードから一文字ずつ受け取り、画面にその都度表示を行う。キーボードから x が入ってきたら処理を終わらせる。

処理手順 （1）: char 型の変数を定義する。
処理手順 （2）:「文字を入れてください（終わる時は x）」と表示する。
処理手順 （3）: 一文字入力
処理手順 （4）: ループをし、その中で処理手順2と3を繰り返す。終了条件は x が入ってきたとき。

【記述例】

```
import java.io.*;
public class rensyu08{
   public static void main(String[] args) throws IOException{
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   System.out.println("文字を入れてください（終わる時は x）");
   String nyuryoku01 = br.readLine();
   char handan01 = nyuryoku01.charAt(0) ;
   while( handan01 != 'x'){
         System.out.println( handan01 + "が入力されました。");
         System.out.println("文字を入れてください（終わる時は x）");
         nyuryoku01 = br.readLine();
         handan01 = nyuryoku01.charAt(0) ;
    }
   System.out.println("終わりました");
 }
}
```

（実行結果）



Lesson9　配列を理解しよう

配列とは、変数を複数使う場合に便利な機能です。変数が単一の箱であるとすれば、配列はその箱を連結して複数をひとまとめにしたものと考えてください。

変数 a　変数 a の配列（この場合は 5 つ）

上記のように単一の変数の全く同じものを複数並べたものです。例では 5 つの箱がありますね。これを左を先頭に a(0)番目、a(1)番目、a(2)番目、a(3)番目、a(4)番目というように添字をつけて呼びます。始まりは 0 番目ですのでそこを間違えないようにしてください。
上記の例では int a[5]という様に定義します。char a[4];ならば 4 文字ということです。

■配列はインデックスを[ ] で囲んで宣言します。
char a[] = new char[3]; char 型で要素数が 3 個の配列変数 a の宣言
int a[]= new int[8]; int 型で要素数が 8 個の配列変数 a の宣言

■配列の変数は 0 で始まります。
a[90]なら　a[0]~a[89]が使えます。

■2 次元以上の配列は、インデックスを並べて宣言します。
int　a[][] = new int[10][20] ;　　2 次元配列
int a[][][] = new int[5][10][15]:　3 次元配列

(例)2 次元配列　z[3][4]　3 行 4 列　の配列をイメージにすると

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3

となります。Excel などで用いる A1 とか B2 などというのと似ていますね。
配列を利用する場合に若干これまで学んだルールが変わるので注意しましょう。

まず、変数の内容を利用する場合。

int a[] = new int[3] ;

と定義したとします。初期化は自動的に 0 にて行われます。

または定義の際に一緒に初期化するならば、

int a[] = { 1,2,3,} ;

char a[]={‘a’,’b’,’c’};

とすると 0 番目に 1、1 番目に 2、2 番目に 3 が入ります。これも便利な機能なので覚えておいてください。

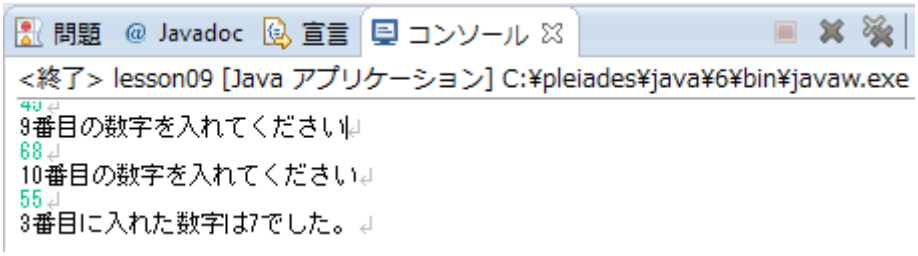
配列の理解のために実例を見てみましょう。

【記述例】

```
import java.io.*;
public class lesson09 {
   public static void main(String[] args) throws IOException{
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int i ;
      int a[] = new int[10] ;
      for( i=0 ; i<10 ; i++ ){
            System.out.println( i +1+ "番目の数字を入れて下さい：");
            String nyuryoku01 = br.readLine();
            int keisan01 = Integer.parseInt(nyuryoku01) ;
            a[i] = keisan01 ;
      }
      System.out.println ("3 番目に入れた数字は" + a[2] + "でした。") ;
   }
}
```

0 番からスタートするため 1 を足しておく

（実行結果）



では、練習問題

【練習問題】

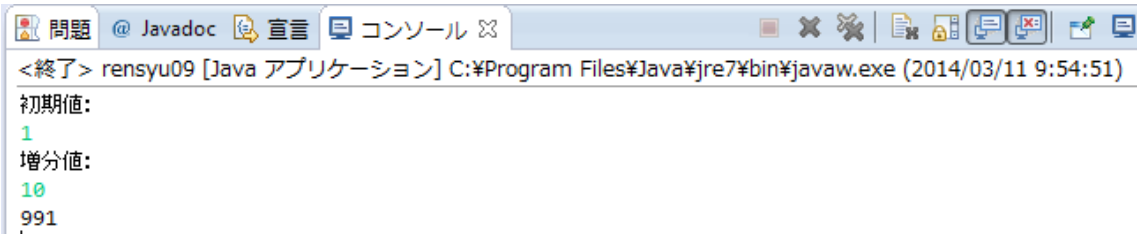
　要素数 100 の配列を作り、そこにキーボードから初期値と増分値をもらって、最初の要素に初期値を、2つめの要素には初期値＋増分値、3つめには2つめ＋増分値、4つめには3つめ＋増分値という風にして全部の要素を埋め、最後の要素に入っている数字を画面に表示せよ。

ヒント：初期値はループ内で入れたらダメですよ。

【記述例】

```
import java.io.*;
public class rensyu09{
public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int a[] = new int[100] ;
      System.out.println( "初期値：");
      String nyuryoku01 = br.readLine();
      int shoki01 = Integer.parseInt(nyuryoku01) ;
      System.out.println( "増分値：");
      nyuryoku01 = br.readLine();
      int zoubun01 = Integer.parseInt(nyuryoku01) ;
      a[0] = shoki01 ;
      int i ;
      for( i=1 ; i<100 ; i++ ){
            a[i] = a[i-1] + zoubun01 ;
      }
      System.out.println( a[99] );
   }
}
```

（実行結果）



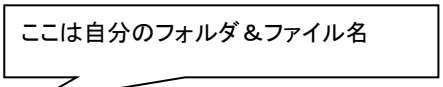
Lesson10 ファイルを扱おう

これまでの学習では実行しても他のファイルを作ったりはしませんでしたね。
今回はファイルを新規作成したり変更したりという処理を行う機能を学習しましょう。

■ファイルの考え方

まず、ファイルとしてテキストファイルを考えます。これはメモ帳などで作成する、ごく単純なものです。メモ帳の操作を思い返してください。まずプログラムを起動したら文章を入れる画面が開きますよね。そこにキーボードで打鍵して文字を打ち込みます。そして終わったら名前を付けて保存をします。今回はそのテキストファイルを入出力する手順を学び、ファイルを入出力できるようにしましょう。

■まず読み込む準備




```
File file = new File("d:¥¥コンピュータ応用演習¥¥test.txt");
FileReader filereader = new FileReader(file);
```

1. ファイルの読み込み

ファイルを読み込むには read メソッドを使います。read メソッドを使うと、ファイルの先頭から文字を 1 文字読み込みます。読み込んだ値は int 型の値として取得します。1文字読み込むと自動的に次の文字へと移りますファイルの最後に達すると「-1」という値を返します。

```
read      //単一の文字を読み込みます
public int read() throws IOException;
```

戻り値は読み込まれた文字で、終わりは -1 になります。

最終的にファイルを CLOSE することが必要です。

★日本語の取り扱い：日本語の文字コードを扱っていると文字化けを起こす場合があります。これを回避するには、Eclipse のウィンドウ→設定→一般→ワークスペース→テキストファイルのエンコードをその他にして UTF-8 を選択すれば解決する場合が多いです。
でも、逆にテキストファイルを使って日本語を読み書きする場合にはこれが仇となる場合もあります。その場合にはその他ではなくデフォルトに変えておきましょう。

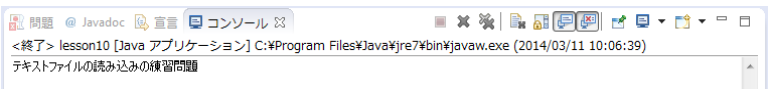
■ファイルを読み込む例

事前にリムーバブルディスク等にメモ帳で test.txt ファイルを新規作成して、その中に文字を入れておくこと、リムーバブルディスクがDドライブならば、f:¥¥test.txt の部分は d:¥¥test.txt とすること　＊ファイル名の最初は英数半角文字

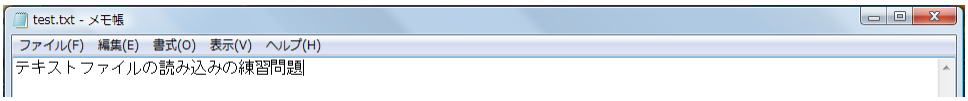
【記述例】

```
import java.io.*;
public class lesson10{
	public static void main(String[] args) {
		//例外をスローする可能性のある処理
		try{
			File file = new File("f:¥¥test.txt");
			FileReader filereader = new FileReader(file);
			//int 型で読み込んだ文字を char 型に変換（キャスト）して表示
			int ch = filereader.read();
			while(ch != -1){
				System.out.print((char)ch);
				ch = filereader.read();
			}
			filereader.close();
		//スローされた例外に対する処理を行う例外ハンドラ
		}catch(FileNotFoundException e){
			System.out.println(e);
		}catch(IOException e){
			System.out.println(e);
		}
	}
}
```

（実行結果）



（プログラムで読み込んだファイルの中身）



例外処理について

ここで出てきた「try」と「catch」は入力の時に説明した例外処理になります。try に書かれた処理を実行し、例外が発生しなければそのまま処理を実行。例外が発生した場合は、例外に合わせて catch へ処理が移行します。

Java では、例外が発生すると例外の種類ごとにクラスが用意されています。対応するクラスのオブジェクトが作成され、そのオブジェクトが catch に渡されます。渡された例外クラスのオブジェクトが catch の引数に書かれた内容と一致すれば、処理を実行します。catch は一つの try に対して、複数記述することができます。

しかし、try と catch に書かれた処理だけでは対応できない場合もあります。try の中で実行したい処理が3つあった場合、1と2の間で例外が発生した場合は、例外が発生した時点で、catch へと移行し try に書かれた3つ目の処理を行いません。この際、3つ目の処理は例外のあるなしにかかわらず必ず実行したい内容であった場合は、「finaly」を利用します。
記述方法は下記のようになります。

```
try{
    例外が発生しているかどうかの確認1;
    例外が発生しているかどうかの確認2;
}catch(例外クラス1 変数名1){
    例外クラス1の例外が発生したときに実行する文;
}catch(例外クラス2 変数名2){
    例外クラス2の例外が発生したときに実行する文;
} finaly{
    例外が発生するしないにかかわらず実行する文;
}
```

例外処理は、メソッドの処理によって記述場所を変える必要があります。finaly は finaly として記述せずに try-catch 文の処理の外に書けば、必ず実行されることになりますが、catch の中で return 文などが実行され、他のメソッドへ移行してしまうと実行されなくなります。例外処理はループ文の中に書くことも可能ですが、処理をよく考え、適切な位置に記述するようにしましょう。

ファイルに関するクラスのオブジェクトを作成する場合には、IOException や FileNotFoundException といった例外が発生する可能性があるので、例外の処理をするようにしましょう。

2. ファイルへの書き込み

ファイルに書き込むには FileWriter クラスを使います。

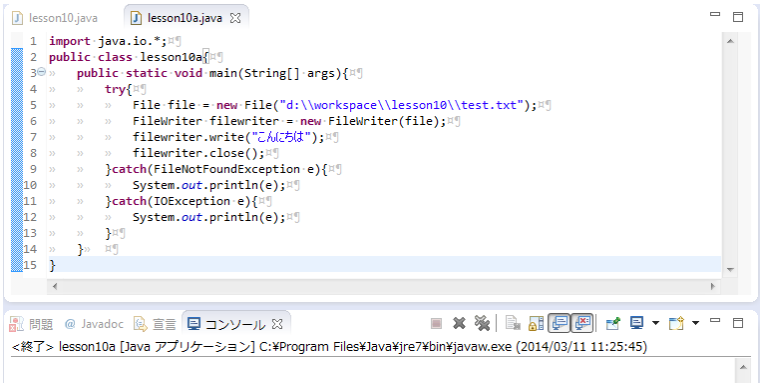
```
File file = new File(":¥¥コンピュータ応用演習¥¥test.txt");
FileWriter filewriter = new FileWriter(file);  //新規書込
FileWriter filewriter = new FileWriter(file,true);  //追加書込
```

【記述例】

```
import java.io.*;
public class lesson10a{
   public static void main(String[] args) {
     try{
       File file = new File("d¥¥test.txt");
       FileWriter filewriter = new FileWriter(file,true);
       filewriter.write("こんにちは");
       filewriter.close();
     }catch(FileNotFoundException e){
       System.out.println(e);
     }catch(IOException e){
       System.out.println(e);
     }
   }
}
```

書き込みたい文字をここに入れる

（実行結果）



（書き出したファイルの中身）

では、練習問題。

【練習問題】

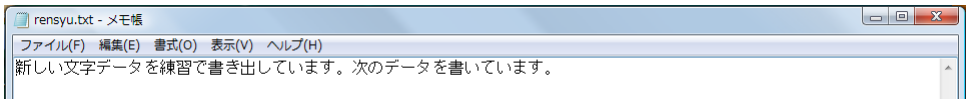
ファイル名 rensyu.txt のテキストファイルに「新しい文字データを練習で書き出しています。」と「次のデータを書いています」を書き出してください。

【記述例】

```
import java.io.*;
public class rensyu10{
	public static void main(String[] args){
		try{
			File file = new File("d:¥¥ rensyu.txt");
			FileWriter filewriter = new FileWriter(file);
			filewriter.write("新しい文字データを練習で書き出しています。");
			filewriter.write("次のデータを書いています。");
			filewriter.close();
		}catch(FileNotFoundException e){
			System.out.println(e);
		}catch(IOException e){
			System.out.println(e);
		}
	}
}
```

（実行結果）

（書き出されたファイルの内容）



Lesson11 乱数

ゲームなどで用いられている乱数。例えばシューティングゲームで敵機の動きが一定だと面白くないですよね。どんな動きをするかわからないからこそ、ゲームとして楽しめると思います。Java で発生させる乱数もゲームの乱数と全く同じ考えで作られています。ですからこの乱数を学べば、ちょっとしたゲームを作ることができるようになるのです。

例えば、敵の攻撃能力を最大30ポイントとした場合、毎回最大ヒットだと面白くないですね、ですからその範囲を0～30ポイントなどとしたほうが、面白くなります。

■乱数の発生方法

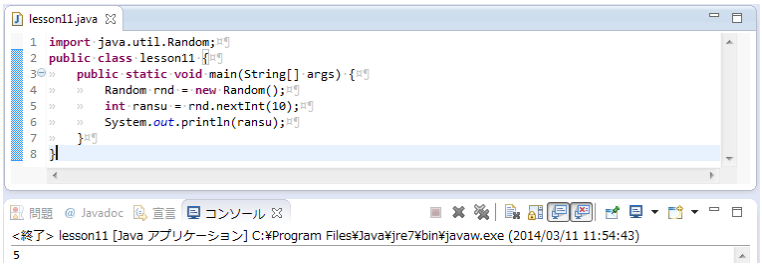
【記述方法】

```
import java.util.Random;
public class lesson11 {
	public static void main(String[] args) {
		Random rnd = new Random();
		int ransu = rnd.nextInt(10);
		System.out.println(ransu);
	}
}
```

※java.util.Random　これは乱数を用いる場合には必ず付けてください。

変数に対して乱数を格納するには対象の変数を左辺に、代入の＝を挟んで　rnd.nextInt(xx)　XX　の数字は発生させたい乱数の数を入れます。但し、スタートが0になるので10ならば0～9の値が出ます。これを1～10にしたければ、rnd.nextInt(10)　+　1　という様に最後に+1　を付けてください。

（実行結果）



【練習問題】

これまでのすべての要素を含んだ形ですごろくゲームを作成しましょう。さいころを1から6までの乱数で発生させ、出た目の数だけ進み、止まるたびにイベントを発生させ、ゴールを目指すものです。表現は自由。

（考え方）

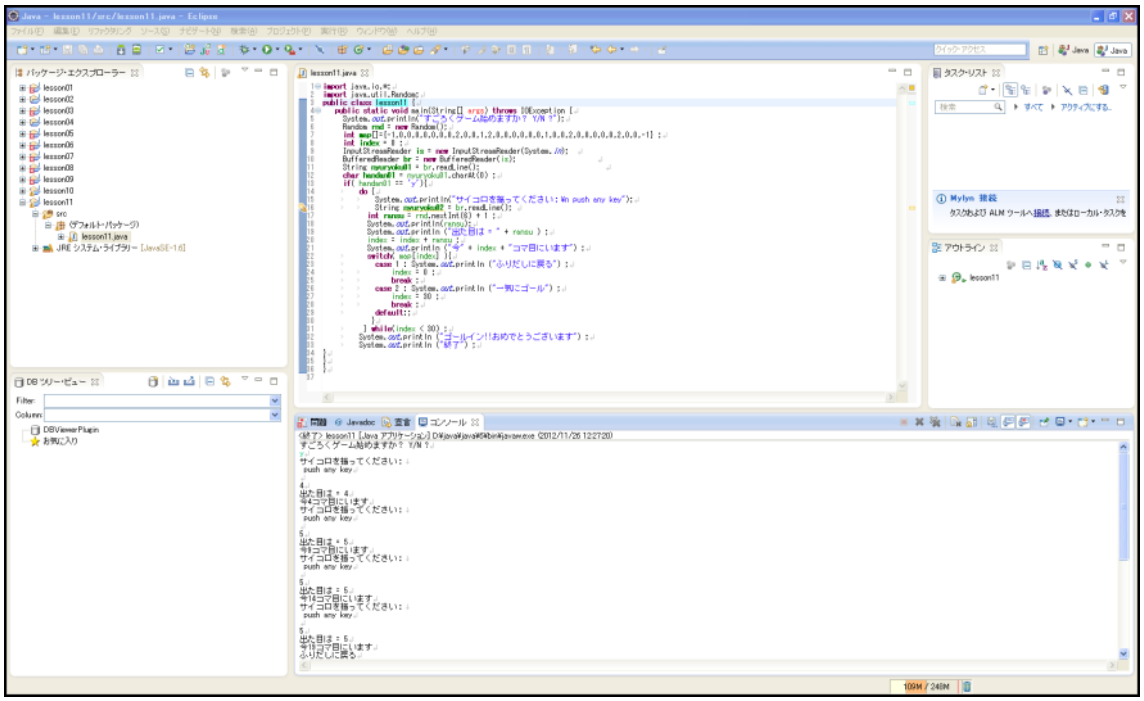
まず、すごろくのマップを考えます。例えばスタートからゴールまで 30 コマあるとしたらその中で何番目にイベントがあるのかを考えて、それを配列にしてみる

例）int map[]={0,1,2,0,0,0,0,0,0,0,0,0,1,2,.....} ;

それを for 文などのループ内でサイコロの数だけ進めていき、止まったところの数字でどんなイベントを発生させるかを決めます。例えば 0 なら何も起こらない、1 は振り出しに戻ります。（この場合、ループの添え字を戻せばよい）など。

```
import java.io.*;
import java.util.Random;
public class rensyu11 {
	public static void main(String[] args) throws IOException {
		System.out.println("すごろくゲーム始めますか？ Y/N ?");
		Random rnd = new Random();
		int map[]={-1,0,0,0,0,0,0,0,2,0,0,1,2,0,0,0,0,0,0,1,0,0,2,0,0,0,0,0,2,0,0,-1} ;
		int index = 0 ;
		InputStreamReader is = new InputStreamReader(System.in);
		BufferedReader br = new BufferedReader(is);
		String nyuryoku01 = br.readLine();
		char handan01 = nyuryoku01.charAt(0) ;
		if( handan01 == 'y'){
			do {
				System.out.println("サイコロを振ってください : ¥n push any key");
				String nyuryoku02 = br.readLine();
				int ransu = rnd.nextInt(6) + 1 ;
				System.out.println(ransu);
				System.out.println ("出た目は = " + ransu ) ;
				index = index + ransu ;
				System.out.println ("今" + index + "コマ目にいます") ;
				switch( map[index] ){
				case 1 : System.out.println ("ふりだしに戻る") ;
					index = 0 ;
					break ;
				case 2 : System.out.println ("一気にゴール") ;
					index = 30 ;
					break ;
				default:;
				}
			} while(index < 30) ;
			System.out.println ("ゴールイン!!おめでとうございます") ;
			System.out.println ("終了") ;
		}
	}
}
```

（実行結果）



Lesson12 バトルシステムの構築

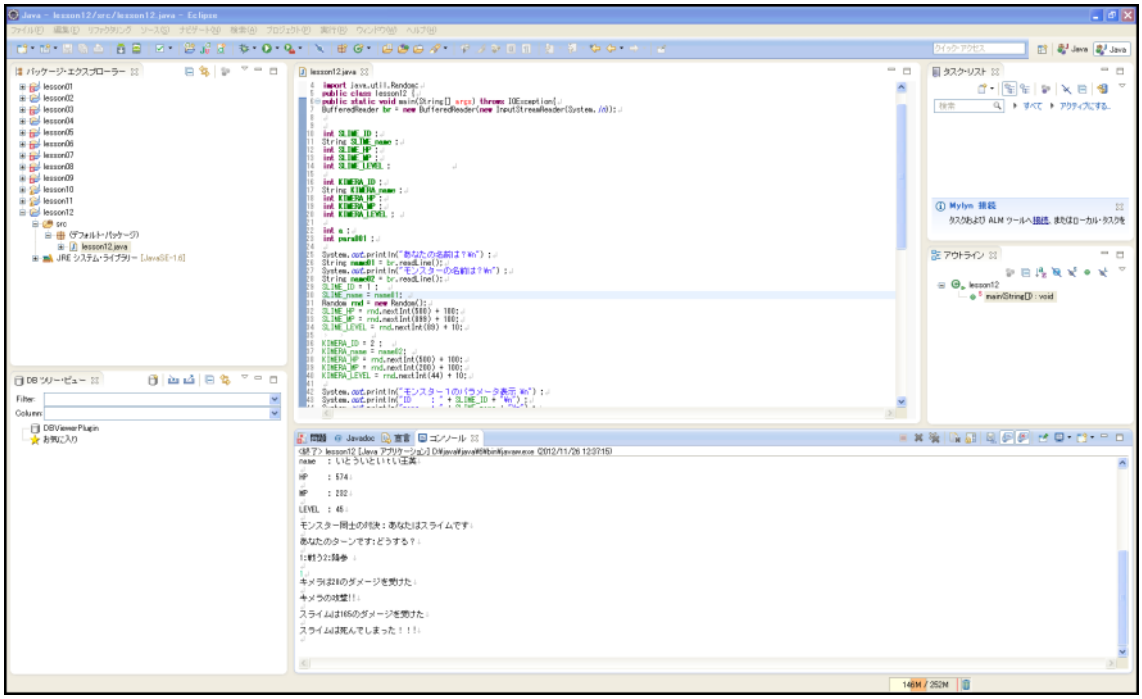
さて、今回は簡単な RPG などで用いられているバトルシステムを作ってみましょう。
ここでは今まで学習してきた全ての技術を用いています。総復習のつもりで確認しながら打ち込んで実行してみましょう。

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Random;
public class lesson12 {
public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int SLIME_ID ;
String SLIME_name ;
int SLIME_HP ;
int SLIME_MP ;
int SLIME_ITEM1 ;
int SLIME_LEVEL ;

int KIMERA_ID ;
String KIMERA_name ;
int KIMERA_HP ;
int KIMERA_MP ;
int KIMERA_ITEM1 ;
int KIMERA_LEVEL ;

int a ;
int para001 ;
String name01;
String name02;

System.out.println("あなたの名前は？¥n") ;
name01 = br.readLine();
System.out.println("モンスターの名前は？¥n") ;
```

```
name02 = br.readLine();
SLIME_ID = 1 ;
SLIME_name = name01;
Random rnd = new Random();
SLIME_HP = rnd.nextInt(500) + 100;
SLIME_MP = rnd.nextInt(899) + 100;
SLIME_LEVEL = rnd.nextInt(89) + 10;

KIMERA_ID = 2 ;
KIMERA_name = name02;
KIMERA_HP = rnd.nextInt(500) + 100;
KIMERA_MP = rnd.nextInt(200) + 100;
KIMERA_LEVEL = rnd.nextInt(44) + 10;

System.out.println("モンスター1のパラメータ表示 ¥n") ;
System.out.println("ID        : " + SLIME_ID + "¥n") ;
System.out.println("name     : " + SLIME_name + "¥n") ;
System.out.println("HP        : " + SLIME_HP + "¥n") ;
System.out.println("MP        : " + SLIME_MP + "¥n") ;
System.out.println("LEVEL   : " + SLIME_LEVEL + "¥n") ;

System.out.println("モンスター2のパラメータ表示 ¥n") ;
System.out.println("ID        : " + KIMERA_ID + "¥n") ;
System.out.println("name     : " + KIMERA_name + "¥n") ;
System.out.println("HP        : " + KIMERA_HP + "¥n") ;
System.out.println("MP        : " + KIMERA_MP + "¥n") ;
System.out.println("LEVEL   : " + KIMERA_LEVEL + "¥n") ;

System.out.println("モンスター同士の対決:あなたはスライムです¥n") ;
do {
    System.out.println("あなたのターンです:どうする？¥n");
    System.out.println("1:戦う 2:降参 ¥n");
    String nyuryoku01 = br.readLine();
    a = Integer.parseInt(nyuryoku01) ;
    if( a == 2 ){
        SLIME_HP = 0 ;
```

```
        }
        else
        {
        para001 = rnd.nextInt(200) ;
        System.out.println("キメラは" + para001 + "のダメージを受けた¥n") ;
        KIMERA_HP = KIMERA_HP - para001 ;
        System.out.println("キメラの攻撃!!¥n") ;
        para001 = rnd.nextInt(200) ;
        System.out.println("スライムは" + para001 + "のダメージを受けた¥n") ;
            SLIME_HP = SLIME_HP - para001 ;
            }
} while(KIMERA_HP > 0 && SLIME_HP > 0) ;

        if( KIMERA_HP > 0 ){
            System.out.println("スライムは死んでしまった！！!¥n") ;
        } else {
            System.out.println("キメラは息絶えた。スライムの勝利！！¥n") ;
        }
    }
}
```

（実行結果）



なお、名前等を漢字で扱いたい場合には以下の設定をして下さい。
java ソースファイルを右クリックしてプロパティーウィンドウ開く。
ウィンドウの左側でリソースを選ぶ。
テキストファイルのエンコードで「その他」を選択し、「MS932」を選ぶ。OK ボタンを押して設定を完了する。

以上です、ここまでの知識をフルに活用して、自分のオリジナル作品を完成させてみましょう。高度なテクニックを使わなくてもいいので、個性豊かな作品を作ってください。

2018 年 12 月 1 日　初版発行

著者：伊藤雅彦、松本清一
発行：大阪アクティブラーニングスクール