Ｊａｖａ入門

（№.１）

科　　名		氏　　名	
Tutorial group			

2004.9

1 Ｊａｖａ

Ｊａｖａは、Sun Microsystems 社によって開発され、1995 年に発表されたオブジェクト指向プログラミング言語である。Ｊａｖａの仕様は、1991 年に開発された家電製品用ソフトウェア開発言語Ｏａｋを基本としている。

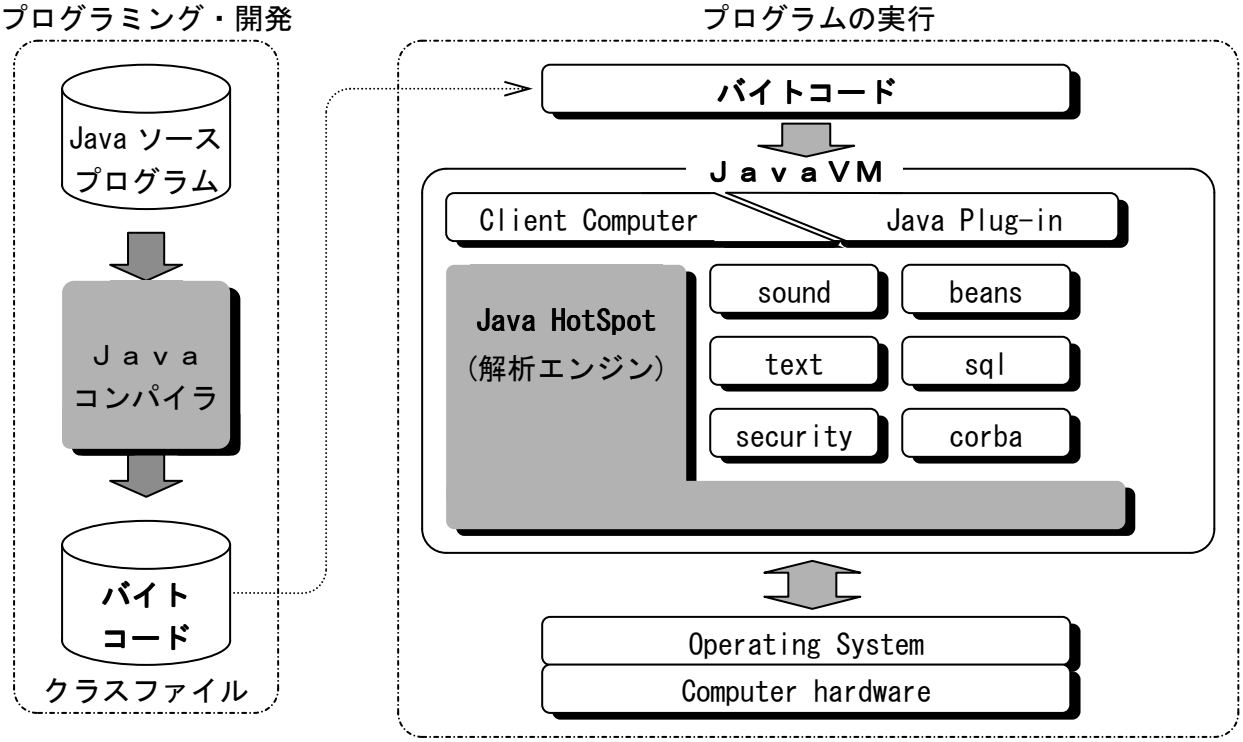
Ｊａｖａの最大の特長は、"Write Once, run anywhere"と言う標語のとおりプラットフォームに依存しないことにある。

2 バイトコード(Byte code)とＪａｖａ ＶＭ(Virtual Machine)

ＣやＣ＋＋などのプログラミング言語のコンパイラは、マシーンコード(machine code)と呼ばれるコンピュータが認識可能な命令群に翻訳したバイナリーコード(binary code)のファイルを生成するが、これらのバイナリーコードには異なるＯＳ間での互換性は無い。

Ｊａｖａコンパイラはソースプログラムを翻訳する場合、コンピュータが直ちに実行可能なバイナリーコードのファイルを生成せず、**バイトコード**(byte code)と呼ばれる仮想的なマシーンコードである中間コード(pseudo code)のファイルを生成する。

Ｊａｖａプログラムの実行は、この**バイトコード**をＪａｖａ ＶＭ(Virtual Machine)が解釈しマシーンコードに変換して実行される。



すなわち、プラットフォームごとにＪａｖａ ＶＭを作製すればソースコードの修正や再コンパイルの作業は全く必要無くなる。

3 Ｊａｖａプログラム

Ｊａｖａプログラムは大別すると

(a) **アプレット**(applet)

(b) **アプリケーション**(application)

に分類することができる。

アプレットとは、Web ブラウザやアプレットビューア上で実行されるＪａｖａプログラムのことである。**アプレット**は、一般的にインターネット上からダウンロードされ、Web ブラウザ上で実行される。

アプリケーションとは、Java コマンド(java.exe)によってＪａｖａ ＶＭを起動して直接実行されるＪａｖａプログラムのことである。

Ｊａｖａアプリケーションには、コンソールアプリケーション、サーブレット及び Java Beans コンポーネントなどがある。

4 Ｊａｖａプログラミング環境

前述のとおり、Ｊａｖａアプリケーションは**Ｃ/Ｃ＋＋**などのコンパイラ型言語のように、

ソースプログラムの作成 ⇒ ソースプログラムのコンパイル ⇒ Ｊａｖａプログラムの実行

の手順で作成・実行される。

次に、Ｊａｖａアプリケーションを作成するために必要なツールを示す。

(a) テキストエディタ ⇒ CPad for Java2 SDK

(b) Java 2 SDK(Software Development Kit) ⇒ Java 2 SDK 1.3.1

} *down load* 可能

http://java.sun.com/から無料で入手可能。

Java 2 SDK は、当初 JDK(Java Development Kit)と呼ばれていた。最初に発表された開発キットは JDK 1.02 であり、その後機能拡張が行われ JDK 1.1 となり、更に JDK の機能は拡張され Java 1.2 となった。

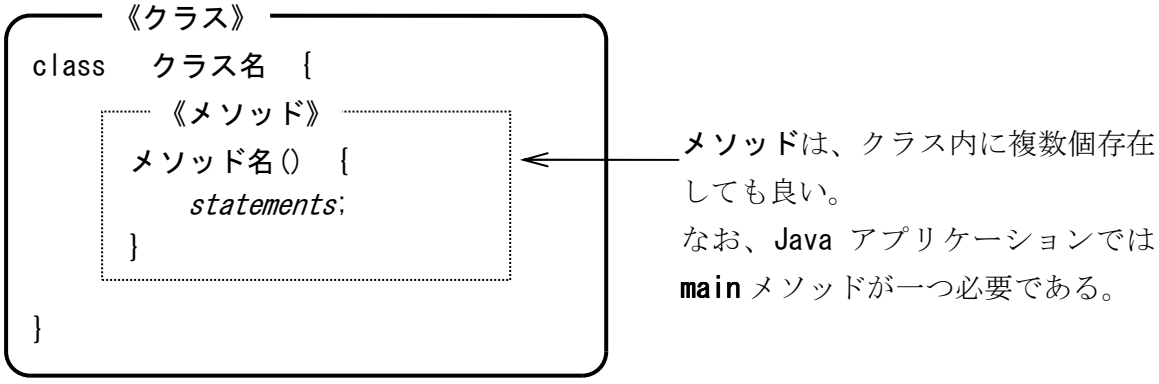
Java 1.2 の機能拡張がかなり大幅であったため、Java 1.2 から Java 2 と呼ばれるようになり、開発キットも Java 2 SDK となった。

5 Java プログラムのコーディング

Java プログラムの構成・特徴は、次のとおりである。

(1) Java プログラムは、class の定義から始まる。

(2) Java VM は、**アプリケーション**の起動時に main メソッド(method)を探し、このメソッドに制御を渡す。ただし、**アプレット**には main メソッドは無い。



[プログラム]

```
import  java.lang.*;  ←――――①
public  class   app   ←――――②
{   //  "Hello from Java!"を表示するプログラム  ←――――③
    public  static  void    main(String args[]) ←――――④
    {
        System.out.println("Hello from Java!");←――――⑤
    }
}
```

【解説】

① パッケージ"java.lang"を、この Java アプリケーションに組み込む。

② public 属性を持つクラス"app"を定義する。このクラス名は、この Java アプリケーションを格納しているファイルのベース名と同一でなければならない。なお、英大文字/英小文字は区別される。 ⇒ アクセス指定子

③ //から行末までは、コメント(comment)として処理される。
また、Java プログラムではブロックは"{"と"}"で囲む。

④ public 属性を持つメソッド"main"を定義する。main メソッドは、JavaVM によって制御が渡され、Java アプリケーションのエントリポイントとなる。
また、メソッドが受け取る引数は( )内に記述する。

⑤ "System"クラス内の"out"クラス変数の"println"メソッドを使用して、引数"Hello from Java!"を標準出力に表示する。println メソッドは、文字列を表示後カーソルを改行する。
なお、Java プログラムでは、";"で文は完結する。

6 Java アプリケーションのコンパイル

Java アプリケーションを、javac.exe コマンドを使用してコンパイルする。

【使用方法】

```
prompt> javac app.java Enter
```

【構文】

```
javac  [options] [source-files] [@files]
  options      … javac で使用できるオプションを指定する。
  source-files … javac がコンパイルする一つまたは複数のソースファイルを指定する。
  @files       … ソースファイルの一覧を含む一つまたは複数のファイルを指定する。
```

本校で使用している"CPad for Java2 SDK"では、Ctrl+F8 を押下する。

7 Java アプリケーションの実行

Java アプリケーションを、java.exe コマンドを使用して実行する。

【使用方法】

```
prompt> java app.class Enter
```

【構文】

```
java   [options]   class   [argument ・・・]
```

```
java   [options]   -jar    file.jar    class   [argument ･･･]
  options      … java で使用できるオプションを指定する。
  class        … java が実行するクラスファイルを指定する。
  file.jar     … java が呼び出す JAR(Java Archive)ファイルを指定する。
  argument     … main()メソッドに引き渡すコマンドライン引数を指定する。
```

本校で使用している"CPad for Java2 SDK"では、Shift+F8を押下する。

なお、本校で使用している"CPad for Java2 SDK"では、F8を押下するとコンパイルと実行を一連の操作として行う。

【例題 1】

［ファイル名］

③ → Hello.java　　(D:¥Home¥User1XX¥Java に保存する。)

［プログラムの説明］

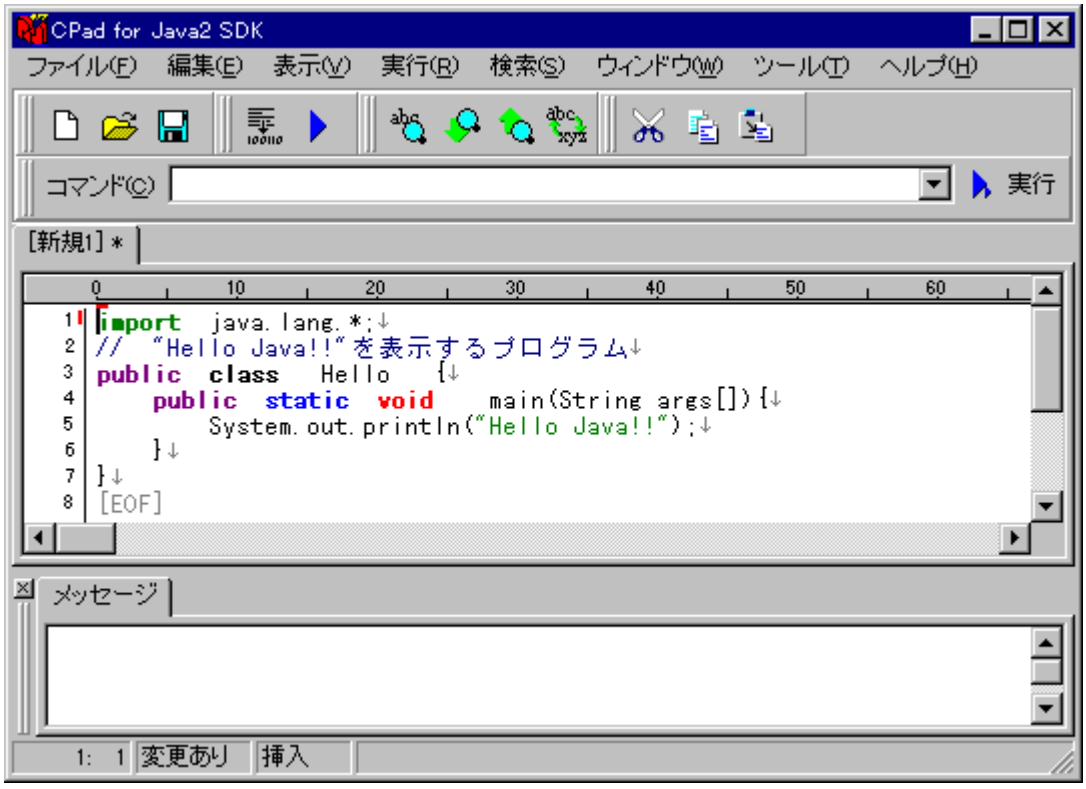
標準出力に、"Hello Java!!"の文字列を表示するプログラムを作成しなさい。

［プログラム］

```
import  java.lang.*;
//  "Hello Java!!"を表示するプログラム
public  class   Hello   {
    public  static  void    main(String args[]){
        System.out.println("Hello Java!!");
    }
}
```

【解説】

次の"CPad for Java2 SDK"を使用して、Java アプリケーションを作成・実行する。



① スタート ボタン ⇒ [ﾌﾟﾛｸﾞﾗﾑ(P)] ⇒ [CPad for Java2]と操作し、"CPad for Java2 SDK"を

起動する。

② コードウィンドウ[**新規 1**]に、Java アプリケーションのソースコードを記述する。

③ メニューバーから、[**ファイル(F)**] ⇒ [**名前を付けて保存(A)...**]と操作し、[**ファイル名を付けて保存**]ダイアログの[**ファイル名(N):**]欄に"Hello.java"と入力し、**保存 (S)** ボタンをクリックする。

Java アプリケーションのソースファイルのベース名は、クラス名と同一でなければならない。また、英大文字/英小文字は区別される。

④ **F8** を押下して、コンパイルと実行を行う。

⇒ コンパイルエラーなどがあると、[**メッセージウィンドウ**]にエラーが表示される。

⑤ 標準出力画面に、Java アプリケーションの実行結果が表示される。

【構文①】- comment

Java は、ソースコード内に説明のためのコメントを記述することができる。Java コンパイラは、コメントを無視しコンパイルの対象とはしない。

Java のコメントの記述形式は、次の 3 種類である。

- //　　… //から行末までの 1 行を、コメントとして扱う。
- /*～*/　… /*から*/までの間のテキストを、コメントとして扱う。
- /**～*/ … /**から*/までの間のテキストを、ドキュメントコメント(document comment)として扱う。

【構文②】- import

Java プログラムに、パッケージの**名前空間**(Name Space)を導入する。

- **名前空間の導入**

  import　パッケージ階層名;

Java の主なパッケージを、次に示す。

パッケージ名	パッケージの内容
java.applet	アプレット関連クラス群
java.awt	AWT(Abstract Window Toolkit)の GUI インターフェース関連クラス群
java.awt.event	AWT のイベント関連クラス群
java.beans	JavaBeans コンポーネント関連クラス群
java.io	入出力ストリーム関連クラス群
java.lang	Java の核となるクラス群(ディフォルトでインポートされる)
java.net	ネットワーク関連クラス群
java.security	セキュリティ関連クラス群
java.text	国際化対応書式化関連クラス群
java.util	コレクション関連クラス群
java.swing	Swing の GUI クラス群

【構文③】- class

Java アプリケーションのクラスの定義を開始する。

- **クラスの定義**

  [Public]　**class　クラス名**

- クラス名は、ソースファイルのベース名と同一でなければならない。

・ Java アプリケーションでは、ソースファイル中には public 属性のクラスは、たった一つだけ定義できる。
・ public 属性のクラスは公開される。

【構文④】- main()メソッド

Java アプリケーションのエントリポイントを定義する。

・ **main()メソッドの定義**
  public static void **main**(String args[])
・ Java アプリケーションクラスには、main()メソッドを必ず一つ定義する必要がある。
・ main()メソッドは、クラスの外部からアクセスされる可能性があるので、Public 属性を与えておく。
・ main()メソッドは、戻り値を持たないので、void を指定しておく。
・ main()メソッドは、コマンドライン引数を受け取ることができる。

【構文⑤】- println()メソッド

System クラスの println()メソッドは、引数に与えられた文字列を**標準出力**に出力する。

・ メソッドの呼び出し
  System.out.**println**(引数);

【例題 2】

[ファイル名]

MyNameIs.java　　(D:¥Home¥User1XX¥Java に保存する。)

[プログラム]

```
// 名前を表示するプログラム
public class MyNameIs {
    public static void main(String args[]){
        System.out.print("My name is ");
        System.out.println("XXXXXX XXXXXX.¥n");
    }
}
```

※ XXXXXX XXXXXX は、各自の名前を記述する。

【考察】

【問題 1】

[ファイル名]

MyName.java　　(D:¥Home¥User1XX¥Java に保存する。)

[プログラムの説明]

一つの println()**メソッド**によって、[出力例]のように標準出力に表示するプログラムを作成しなさい。

[出力例]

Hello!!

My name is XXXXXX XXXXXX.

※ XXXXXX XXXXXX は、各自の名前を記述する。

【考察】

【例題 3】

[ファイル名]

Hizuke.java

[プログラムの説明]

テキスト・P38 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Hizuke"とすること(以降、クラス名"app"は[ファイル名]に指示したクラス名とする)。

なお、パッケージのインポートは、"java.util.Date"とすること。

【考察】

【例題 4】

[ファイル名]

printer.java,　prnapp.java

[プログラムの説明]

テキスト・P39 printer クラスと app クラスのプログラムを作成しなさい。

【手順】

① コードウィンドウ[**新規 1**]に、テキスト・P39 printer クラスをコーディングする。

② メニューバーから、[**ファイル(F)**] ⇒ [**名前を付けて保存(A)...**]と操作し、[**ファイル名を付けて保存**]ダイアログの[**ファイル名(N):**]欄に"printer.java"と入力し、**保存(S)**ボタンをクリックする。

③ **Ctrl**+**F8**を押下して、コンパイルを行う。

⇒ コンパイルエラーが無く、"printer.class"が作成されることを確認する。

④ メニューバーから、[**ファイル(F)**] ⇒ [**新規作成(N)**]と操作し、コードウィンドウ[**新規 2**]を開く。

⑤ コードウィンドウ[**新規 2**]に、テキスト・P39 app クラスをコーディングする。ただし、クラス名"app"はクラス名"prnapp"とすること。

⑥ メニューバーから、[**ファイル(F)**] ⇒ [**名前を付けて保存(A)...**]と操作し、[**ファイル名を**

付けて保存]ダイアログの[ファイル名(N):]欄に"prnapp.java"と入力し、 保存(S) ボタンをクリックする。

⑦ F8 を押下して、コンパイルと実行を行う。

【考察】

【実験 1】

[ファイル名]

printapp.java

[プログラム]

```
// printer クラスと app クラスのプログラム(P39)
class println { ←――①  自作の println クラス
    public void print() { ←――②  自作の print メソッド
        System.out.println("Hello from Java!");
    }
}

public class printapp { ←――③  自作の printapp クラス
    public static void main(String args[]) {
        (new println()).print();  ←――④
    }
}  // End of printapp.java
```

【考察】

8 データ型と変数

Ｊａｖａには、次の 8 種類の**基本データ**(primitive)**型**がある。

	型 名	記憶サイズ	範 囲	デフォルト値
整数型	byte	1byte	-128 ～ 127	0
	short	2byte	-32,768 ～ 32,767	0
	int	4byte	-2,147,483,648 ～ 2,147,483,647	0
	long	8byte	-9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807	0
*1	float	4byte	±1.401298E-45 ～ ±3.402823E38	0.0
	double	8byte	±4.94065645841247E-324 ～ ±1.79739313486232E308	0.0
*2	char	2byte	'¥u0000' ～ '¥uffff'(0 ～ 65,535)	'¥u0000'
*3	boolean	1byte	true/false	false

*1 浮動小数点数型

*2 文字型 … Unicode の 1 文字を表す。

*3 論理型

※ デフォルト値は、基本データ型のインスタンス変数とクラス変数に自動的に設定される。

・ Ｊａｖａは、データ型が厳密なプログラミング言語である。しかし、基本データ型はboolean型を除いて相互に変換できる。

(a) **ナローイング変換**(narrowing conversion) … ある型からより少ないbit数への型変換

(b) **ワイドニング変換**(widening conversion) … ある型からより多くのbit数への型変換

・ **ナローイング変換**は、キャストによって型変換しなければならない。⇒ 縮小変換

・ **ワイドニング変換**は、自動的に行われる。⇒ 拡張変換

◆ **変数**(variable)

変数は、データが格納されている主記憶装置上の位置を表す識別子である。

Ｊａｖａの変数の種類は、次のとおりである。

(1) **インスタンス変数** … オブジェクトの属性を表す変数。

(2) **クラス変数** … あるクラスのオブジェクト全体で共有される変数。

(3) **ローカル変数** … メソッド内で一時的に使用する変数。

(a) **変数の宣言**

型名　変数名;

(b) **変数の宣言と初期化**

型名　変数名 = 初期値;

・ 変数の宣言時に、","で区切って同時に複数の変数を宣言できる。

・ Ｊａｖａの変数の命名規約は、次のとおりである。

① 先頭は英字(A～Z, a～z)、下線(_)あるいはドルマーク($)である。

② 2文字目以降は、①に加えて数字(0～9)である。

③ Ｊａｖａの予約語、true、false及びnullは使用できない。

④ 英大文字と英小文字は区別される。

・ 変数名の先頭文字には、下線(_)とドルマーク($)は使用すべきではない。

◆ **リテラル**(literal)

リテラルは、Ｊａｖａプログラムのソースコード中に記述された定数である。

・ Ｊａｖａのリテラルの命名規約は、変数名の命名規約に従う。

・ **整数リテラル**のデフォルトは、int型である。

・ **整数リテラル**は、先頭に0を付けると8進数形式として、先頭に0xあるいは0Xを付けると16進数形式として扱われる。

・ **整数リテラル**で、long **リテラル**を明示的に表す場合、リテラルの末尾にLを付ける。

・ **浮動小数点数リテラル**のデフォルトは、double型である。

・ **浮動小数点数リテラル**で、float **リテラル**を明示的に表す場合、リテラルの末尾にfを付ける。

◆ **配列**(array)

配列は、同じ型(基本データ型、参照型)のデータを複数格納できるオブジェクトである。

・ **配列の宣言と生成**

(a) **型名　配列名[] = new　型名[要素数]**;

(b) **型名[]　配列名 = new　型名[要素数]**;

・ 配列の宣言は、**配列名**の定義を行うだけで実際に**配列**を主記憶装置には割り当てない。
・ 配列の生成は、実際に**配列**を主記憶装置に割当て、**配列名**から参照できるように関連付ける。
・ **配列**の要素数は、配列の生成時に決定され変更することはできない。
・ **配列**は、配列の生成時に、初期値を明示的に設定することができる。

【例題 5】 ― 整数型

［ファイル名］

Intapp.java

［プログラムの説明］

テキスト・P56 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Intapp"とすること。

【考察】

【例題 6】

［ファイル名］

Longapp.java

［プログラムの説明］

テキスト・P64 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Longapp"とすること。

【考察】

【例題 7】

［ファイル名］

Intapp2.java

［プログラムの説明］

テキスト・P65 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Intapp2"とすること。

【考察】

【例題 8】

［ファイル名］

Intapp3.java

［プログラムの説明］

テキスト・P71 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Intapp3"とすること。

【考察】

【例題 9】－ 浮動小数点数型

［ファイル名］

Dblapp.java

［プログラムの説明］

テキスト・P66 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Dblapp"とすること。

【考察】

コンパイルできるように、変数 value を適切な型に変更しなさい。

【例題 10】

［ファイル名］

Fltapp.java

［プログラムの説明］

テキスト・P67 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Fltapp"とすること。

【考察】

【例題 11】

［ファイル名］

Dblapp2.java

［プログラムの説明］

テキスト・P72 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Dblapp2"とすること。

【考察】

【例題 12】－ 文字型

［ファイル名］

Chrapp1.java, Chrapp2.java

［プログラムの説明］

テキスト・P68 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Chrapp1"とクラス名"Chrapp2"とすること。

【考察】

次の ASCII コード(抜粋)をもとにして、$(67)_{10}$が'C'を表すことを説明しなさい。

ASCII コード(抜粋)								
＼	0	1	2	3	4	5	6	7
0	NUL	TC_7	SP	0	@	P		p
1	TC_1	DC_1	!	1	A	Q	a	q
2	TC_2	DC_2	″	2	B	R	b	r
3	TC_3	DC_3	#	3	C	S	c	s
4	TC_4	DC_4	$	4	D	T	d	t

【例題 13】

［ファイル名］

Chrapp3.java

［プログラムの説明］

テキスト・P73 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Chrapp3"とすること。

【考察】

【例題 14】

［ファイル名］

Chrapp4.java

［プログラムの説明］

テキスト・P74 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Chrapp4"とすること。

【考察】

算術演算子		
演算子	用法	説　　明
++	++op	op を+1 して、その値を返す。
	op++	op の値を返して、op を+1 する。
--	--op	op を-1 して、その値を返す。
	op--	op の値を返して、op を-1 する。
+	+op	op に「単項の数値格上げ変換」を適用する。
-	-op	op の符号を反転する。

【例題 15】

［ファイル名］

Chrapp5.java

［プログラム］

```
//  +演算子を確認するプログラム
public  class   Chrapp5 {
    public  static  void    main(String args[]) {
        char    char1 = 'A';
        char    char2 = (char)(char1 + 1);

        System.out.println("char1 = " + char1);   ←――①
        System.out.println("char2 = " + char2);
        System.out.print("char1 + char2 = ");
        System.out.println(char1 + char2);        ←――②
    }
}   //  End of Chrapp5.java
```

【考察】

+演算子の動作(①と②)について、考察しなさい。

【例題 16】－ 論理型

［ファイル名］

Boolapp1.java

［プログラムの説明］

テキスト・P67 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Boolapp1"とすること。

【考察】

【例題 17】

［ファイル名］

Boolapp2.java

［プログラムの説明］

テキスト・P74 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Boolapp2"とすること。

【考察】

【例題 18】

［ファイル名］

Boolapp3.java

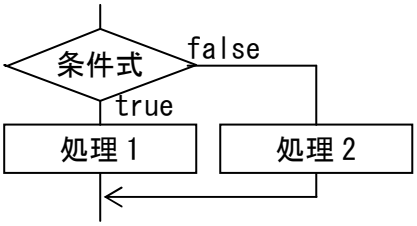
［プログラムの説明］

テキスト・P75 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Boolapp3"とすること。

【構文⑥】－ if 文

条件式が true(**真**)の場合に"**処理 1**"を、false(**偽**)の場合に"**処理 2**"を実行する制御文。

```
if(条件式){
    statements; ← 処理 1
}else{
    statements; ← 処理 2
}
```



※ 実行する *statements* が 1 文の場合、**{}**は記述する必要はない。

【考察】

【例題 19】－ 配列

［ファイル名］

array1.java

［プログラムの説明］

テキスト・P58 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"array1"とすること。

【考察】

【例題 20】

［ファイル名］

array2.java

［プログラムの説明］

テキスト・P59 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"array2"とすること。

【考察】

【例題 21】

［ファイル名］

array3.java

［プログラムの説明］

テキスト・P60 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"array3"とすること。

【考察】

【例題 22】

［ファイル名］

array4.java

［プログラムの説明］

テキスト・P87 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"array4"とすること。

【構文⑦】－ for 文

条件式が true(**真**)の場合に、指定された処理を繰返し実行する制御文。

for(初期化; 条件式; 更新){

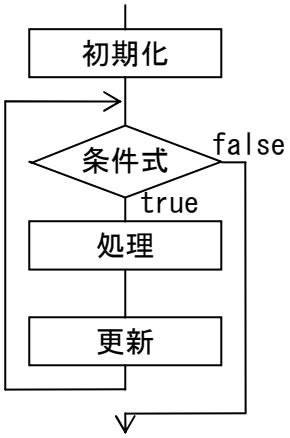
　　statements; ← **処理**

}

(a) **初期化** … for 文の開始前に 1 回だけ実行される。

(b) **条件式** … 繰返しごとに評価される。

(c) **更新**　 … 処理終了後に毎回実行される。



※ 実行する *statements* が 1 文の場合、**{}**は記述する必要はない。

◆ length **インスタンス変数**

length **インスタンス変数**は、配列の要素数を格納している。

インスタンス変数とは、インスタンスごとに異なる値(属性)を保持するための変数である。

⇒ データメンバ

・ インスタンス変数の参照(インスタンスの外部)

インスタンス名.インスタンス変数名

【考察】

【例題 23】

［ファイル名］

array5.java

［プログラムの説明］

テキスト・P88 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"array5"とすること。

【考察】

【例題 24】

［ファイル名］

array6.java

［プログラムの説明］

テキスト・P89 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"array6"とすること。

【考察】

【例題 25】

［ファイル名］

array7.java

［プログラムの説明］

テキスト・P90 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"array7"とすること。

【考察】

9 String クラス

String **クラス**は、初期化の除いて変更のできない文字列を格納するオブジェクトである。

また、文字列リテラルも、暗黙に String **クラス**の一時オブジェクトが自動的に生成され、String クラスオブジェクトとして処理される。

(例) String s1 = "Hello from Java!";

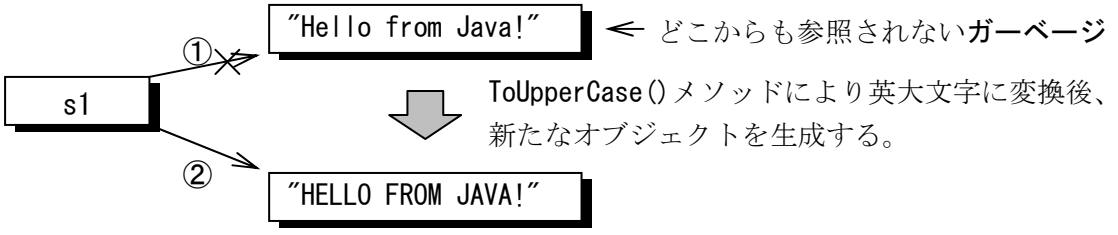
↓ 次のように解釈される。

String s1 = new String("Hello from Java!");

したがって、String **クラス**のメソッドの多くは別の新しい String クラスオブジェクトを生成して返している。⇒ 参照を代入

(例) String s1 = "Hello from Java!"; ← ①

s1 = s1.toUpperCase(); ← ②



【例題 26】－ String クラス

［ファイル名］

Strapp1.java

［プログラムの説明］

テキスト・P91 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Strapp1"とすること。

【考察】

【実験 2】

［ファイル名］

Strapp1a.java

［プログラム］

```
//  String クラスオブジェクト(P91)改
public  class   Strapp1a{
    public  static  void  main(String args[]) {
        String  s1;
        s1 = new String("Hello from Java!!");

        System.out.println(s1);

        s1 = s1.toUpperCase();              //  見かけ上編集されたように見える
        System.out.println(s1);

        s1 = "My name is " + "XXXXXXX XXXXX.";
        System.out.println(s1);
    }
}   //  End of Strapp1a.java
```

【考察】

【例題 27】

［ファイル名］

Strapp2.java

［プログラムの説明］

テキスト・P97 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Strapp2"とすること。

◆ メソッド(method)－(1)

static キーワードのついたメソッドは**クラスメソッド**(class method)と呼ばれ、クラス定義だけに関連付けられたメソッドである。

・ **クラスメソッドの呼び出し**

クラス名.クラスメソッド名()

(例)　String.valueOf(double1);

・ **クラスメソッド**は、オーバーロードできる。

・ valueof()メソッドは、『与えられた引数の値の文字列表現を返す』メソッドである。

【考察】

【例題 28】

[ファイル名]

Strapp3.java

[プログラムの説明]

テキスト・P98 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Strapp3"とすること。

◆ メソッド(method)－(2)

オブジェクトの中に含められる種類のメソッドを、**インスタンスメソッド**(instance method)と言う。⇒ メンバ関数

・ **インスタンスメソッドの呼び出し**

インスタンス名.インスタンスメソッド名()

(例)　s1.length()

・ **インスタンスメソッド**は、オーバーロードできる。

・ length()メソッドは、『インスタンスの文字数を返す』メソッドである。

【考察】

【例題 29】

[ファイル名]

Strapp4.java

[プログラムの説明]

テキスト・P99 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Strapp4"とすること。

◆ 文字列の連結

Ｊａｖａでは、文字列の連結は、+演算子と concat()メソッドによって行うことができる。

【考察】

【例題 30】

［ファイル名］

Strapp5.java

［プログラムの説明］

テキスト・P101 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Strapp5"とすること。

【考察】

【例題 31】

［ファイル名］

Strapp6.java

［プログラムの説明］

テキスト・P102 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Strapp6"とすること。

◆ replace()メソッド

replace()メソッドは、『インスタンス内の文字列の中の、引数 1 で示された文字を、引数 2 で示された文字にすべて置換え、新しいオブジェクトの参照を返す』メソッドである。

（例） String　s2 = "Edna, you¥'re hired!";　← ①
System.out.println(s2.replace('h', 'f'));　← ②

- ①の s2 は、"Edna, you¥'re hired!"の参照を格納している。
- ②の s2.replace('h', 'f')は、新しい String クラスのオブジェクト"Edna, you¥'re fired!"を生成して、その参照を返している。
- 新しい String クラスオブジェクト"Edna, you¥'re fired!"は、println()メソッドの終了後、どこからも参照されない**ガーベージ**(garbage)となるが、Ｊａｖａでは**Ｊａｖａインタープリタ**の**ガーベージコレクタ**(garbage collector)が自動的にメモリ領域を解放する。⇒ **ガーベージコレクション**(garbage collection)
- このため、Ｊａｖａのクラスにはデストラクタはない。

【考察】

【例題 32】

［ファイル名］

Strapp7.java

［プログラムの説明］

テキスト・P103 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Strapp7"とすること。

【考察】

【例題 33】

［ファイル名］

Strapp8.java

［プログラムの説明］

テキスト・P104 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"Strapp8"とすること。

【考察】

10 StringBuffer クラス

StringBuffer クラスは、初期化後もオブジェクトの内容を変更することができるクラスである。すなわち、StringBuffer クラスのメソッドでは、自分自身を変更することが可能である。

◆ StringBuffer クラスと String クラス間の相互変換

(1) String クラスオブジェクト ⇒ StringBuffer クラスオブジェクト

StringBuffer クラスのコンストラクタを使用して、StringBuffer クラスオブジェクトを生成する。

(例) String str = new String("Hello from Java!");

StringBuffer strbuff = new **StringBuffer**(str);

(2) StringBuffer **クラス**オブジェクト ⇒ String **クラス**オブジェクト
StringBuffer **クラス**の substring()メソッドを使用して、String **クラス**オブジェクトを生成する。
(例) str = strbuff.**substring**(0);

(3) StringBuffer **クラス**オブジェクト ⇒ String **クラス**オブジェクト
StringBuffer **クラス**の toString()メソッドを使用して、String **クラス**オブジェクトを生成する。
(例) str = strbuff.**toString**();

【例題 34】

[ファイル名]

StrBapp1.java

[プログラム]

```
public  class   StrBapp1{
    public  static   void    main(String args[]) {
        StringBuffer    s1 = new StringBuffer("Hello from Java!");
                                                  ↑          ↑
                                                  │          └──String クラスオブジェクト
                                                  └──StringBuffer クラスのコンストラク

        s1.replace(6, 10, "to");

        System.out.println(s1);
    }
}   // End of StrBapp1.java
```

【考察】

【例題 35】－ insert()メソッド

[ファイル名]

StrBapp2.java

[プログラムの説明]

テキスト・P108 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"StrBapp2"とすること。

【考察】

自身のインスタンスに作用していることに注意すること。

【例題 36】－ length()メソッド, capacity()メソッド, setLength()メソッド

［ファイル名］

StrBapp3.java

［プログラムの説明］

テキスト・P109 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"StrBapp3"とすること。

【考察】

【例題 37】

［ファイル名］

StrBapp3a.java

［プログラム］

```
public  class   StrBapp3a   {
    public  static  void    main(String args[]) {
        StringBuffer    str = new StringBuffer();

        System.out.println("str-capaciry = " + str.capacity());
        System.out.println("str-length   = " + str.length());
    }
}   //  End of StrBapp3a.java
```

【考察】

テキスト・P107 の StringBuffer クラスオブジェクトのデフォルトの容量とサイズを確認しなさい。

【例題 38】－ setCharAt()メソッド

［ファイル名］

StrBapp4.java

［プログラムの説明］

テキスト・P110 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"StrBapp4"とすること。

【考察】

【例題 39】－ append()メソッド

［ファイル名］

StrBapp5.java

［プログラムの説明］

テキスト・P111 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"StrBapp5"とすること。

【考察】

【例題 40】－ deleteCharAt()メソッド

［ファイル名］

StrBapp6.java

［プログラムの説明］

テキスト・P111 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"StrBapp6"とすること。

【考察】

【例題 41】－ replace()メソッド

［ファイル名］

StrBapp7.java

［プログラムの説明］

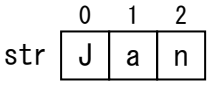
テキスト・P112 app クラスのプログラムを作成しなさい。ただし、クラス名"app"はクラス名"StrBapp7"とすること。

【考察】

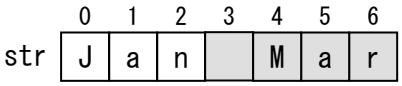
◎ StringBuffer クラスのまとめ

(例)

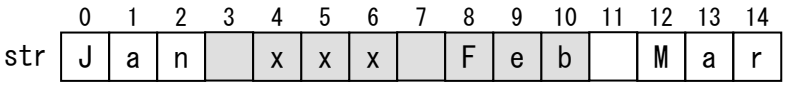
① StringBuffer　str = new StringBuffer("Jan");



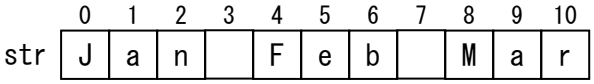
② str.append(" Mar");



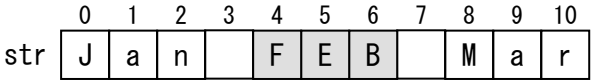
③ str.insert(3, " xxx Feb");



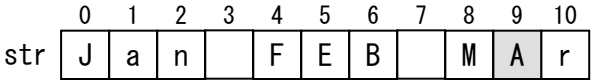
④ str.delete(4, 8);



⑤ str.replace(4, 7, "FEB");



⑥ str.setCharAt(9, 'A');



【考察】