第一週 Java の概要、オーバーロード

Java について

Java は 1995 年に Sun Microsystems 社で開発されたプログラミング言語です. 現在は Oracle 社によって提供されています. スローガンは「Write once, run anywhere」であり、これは一度コードを書けばどの環境で実行できることを意味しています. Java はコンパイルにより生成される中間コード（バイトコード）を JVM（Java Virtual Machine：Java 仮想マシン）に読み込ませることで実行するという仕組みになっています. これにより OS など環境に依存しないようになっています. また、JavaScript という言語とは別物です.

主な特徴

- オブジェクト指向言語である.
- プラットフォームに依存しない.
- C や C++から多くの構文を引き継いでいる.
- 高いセキュリティ機能・ネットワーク機能を標準搭載している.

開発環境

Java の IDE（統合開発環境）としては Eclipse や NetBeans が有名です. エディタを使って開発することも可能です. この講座では演習室の PC にもインストールされている Eclipse を使って開発を行います.

プログラムの実行

それでは Eclipse を使って「Hello MPC !」と表示するプログラムを書いてみましょう. 以下にプログラムを実行するまでの手順を示します.

1. Eclipse の起動

まずは Eclipse を起動してください.

2. ワークスペースの切り替え

ソースファイルなどを作成し保管する「ワークスペース」を用意しましょう. デフォルトではマイドキュメント直下が指定されているのでこれを変更します.

2.1. ワークスペースとするディレクトリを用意する.
2.2. [ファイル] - [ワークスペースの切り替え] - [その他] を選択する.
2.3. [参照] を選択し、2.1 で用意したディレクトリを指定する.

3. プロジェクトの作成

Eclipse は Visual C++と同様プロジェクトで開発を行います.

3.1. [ファイル] - [新規] - [プロジェクト] を選択する.
3.2. [Java プロジェクト] を選択し、[次へ] をクリックする.
3.3. プロジェクト名に「Week1」と入力し、[完了] パッケージエクスプローラに「Week1」ができあれば OK です.

4. クラスの作成

次にソースファイルを作成します。

4.1. パッケージエクスプローラ内の「Week1」を右クリック - [新規] - [クラス] を選択する.
4.2. 名前に「Sample1」と入力し、[完了] をクリックする.

Week1 プロジェクト内に Sample1.java が作成され、エディタが表示されれば、ソースを書く準備は整いました。

5. ソースの入力

それでは記念すべき第一回目のプログラムを書きましょう. エディタに以下のソースを入力してください.

```
public class Sample1{
        public static void main(String[] args) {
                System.out.println("Hello Java World!");
        }
}
```

6. プログラムの実行

実行するソースが書けたら実行してみましょう.

6.1. メニューバーの [実行] - [実行]（[Ctrl + F11]、またはツールバーのボタンでも可）を押す.

6.2. 「常に起動前にリソースを保管する」にチェックを入れ、[OK]を押す. 実行結果は下の「コンソール」ウィンドウに表示されます.

```
Hello Java World!
```

7. エラーが出た場合

7.1. [継続]を選択する.

7.2. 「コンソール」ウィンドウを参考にコードを修正する 実行前でもプログラムの記述にミスがある場合は、その箇所に赤波線が出たり、行番号の左にアイコンが出たりします.

Sample1.java の解説

まず 1 行目の「public class Sample1 {」では、Sample1 という名前の public なクラスを作成しています. public やクラスの詳細については今後の講座で紹介します. ただし、

- ファイル名とクラス名は同じ名前にする.
- 一つの java ファイルには最低一つのクラスが必要である.

という 2 点に注意してください.

2 行目の「public static void main(String[] args) {」は main メソッドといいます. C 言語でいうところの main 関数に当たるものですが、C 言語で「関数」と呼んでいたものは Java において「メソッド」といいます. Java はこの main メソッドから処理を開始します. 頭に付いている public や static についてはこれまた今後の週で説明します.

3 行目の「System.out.println("Hello MPC !");」は標準出力です. これは C 言語でいう「printf()」と同様、( )内の文字を画面に出力します. ただし、「System.out.println()」は文末で必ず改行をするという違いに注意です. 自動で改行をしたくない場合は「System.out.print()」を使います. また、「\n」を使って強制改行させることも可能です.

データ型

Java の変数は C 言語とほとんど変わりません. 以下は Java で使える型の一覧です.

データ型	値
boolean	true または false の二値
char	2 バイト文字(\u0000～\uffff)
byte	1 バイト整数(-128～127)
short	2 バイト整数(-32768～32767)
int	4 バイト整数(-2147483648～2147483647)
long	8 バイト整数(-9223372036854775808～9223372036854775807)
float	4 バイト単精度浮動小数点数
double	8 バイト倍精度浮動小数点数

boolean 型と byte 型以外は C 言語にもあった型です. long 型は C 言語にもありましたが仕様が少し異なります. long 型に値を代入する場合は値の後ろに「L」または「l」という文字をつけなければ代入することができません.

```
long a = 123456789L;
```

boolean 型は値ではなく true か false かの真偽を返す型です. 例えば今まで C 言語でゲーム制作を行ってきた時には flag の管理は int 型で行ってきましたが Java では boolean 型を使って flag の管理をすることができます.

System.out.println()メソッド

System.out.println();は先ほども言いましたが Java の標準出力です. しかし、出力の方法は C とは少し異なります. C では%d や%f といった変換指定文字というのを使って文字列の中に変数や定数などを出力しましたが Java では次のような方法で出力を行います.

```
System.out.println("文字列"+変数名+…");
```

%d や%f を使わずに文字列と変数を+で繋ぎます. 文字列部分は C 言語と同様に" "を使います. エスケープシーケンスは C 言語と同じように使うことが可能です.

四則演算、制御文

Java にも制御文（if(), for(), while(), switch()など）があります. 以下のサンプルコードを打ち込んでください.

```
public class Sample2 {

	public static void main(String[] args) {

		int a = 1 + 1;

		int b = 10 - 8;

		int c = 20 * 8;

		int d = 100 / 20;

		int e = 3 % 2;


		System.out.println("1 + 1 = " + a);

		System.out.println("10 - 8 = " + b);

		System.out.println("20 * 8 = " + c);

		System.out.println("100 / 20 = " + d);

		System.out.println("3 % 2 = " + e);

	}

}
```

```
1 + 1 = 2

10 – 8 = 2

20 * 5 = 160

100 / 20 = 5

3 % 2 = 1
```

```
public class Sample3 {
	public static void main(String[] args) {
		int a=0;

		for(int i=0;i<10;++i){
			System.out.println("i="+i);
		}

		while(true){
			switch(a){
			case 0:
				System.out.println("始まり"); break;
			case 10:
				System.out.println("終わり"); break;
			default:
				System.out.println("その他"); break;
			}
			System.out.println("こんにちは");
			a++;
			if(a > 10){
				break;
			}
		}
	}
}
```

i=0

i=1

i=2

i=3

i=4

i=5

i=6

i=7

i=8

i=9

始まり

こんにちは

その他

こんにちは

その他

こんにちは

その他

こんにちは

・

・

・

終わり

こんにちは

それではC言語との違いを見ていきましょう. まず最初にC言語との違いはfor()の中にあるint i = 0;という文です. Javaは C言語と違い変数をどこでも宣言することができます. なのでfor文の中で宣言することができます. ちなみにこのfor文の中で宣言された変数iはこのfor文の中でしか使えません. C言語の時にやったグローバル変数とローカル変数と同じような考え方でこのfor文の中で宣言された変数iはfor文の中だけで使えるローカル変数ということになります. 次にC言語と違うところはwhile(true)という部分です. C言語ではwhile文で無限ループを作るときはwhile(1)というのを使いましたがJavaではwhile()の引数がboolean型のため、while(true)と書いて無限ループを実装します. そのほかのif文やswitch case文に違いは特にありません. ここでは紹介してませんがdo while文も使えます.

キャスト

Javaにはキャスト(型変換)と呼ばれる機能があります. これはC言語と同じで変数名の前に(型名)を入れて行います.

```
(型名)変数名        // キャスト演算子を用いた明示的な型変換
```

Javaでは暗黙の型変換と明示的な型変換と二つの型変換があります. 一つ目の暗黙の型変換はサイズが小さい型から大きい型への変換でキャスト演算子をつけなくても値を代入したりすることが可能です.

例)

int型から long型

float型から double型

次に明示的な型変換についてです. こちらは暗黙の型変換を使用することができません. サイズが大きい型から小さい型への変換でキャスト演算子を使用する必要があります.

例)

long型から int型

double型から float型

なぜ明示的な型変換が必要なのかというと long型の格納されていた値が int型のサイズを超えていた場合にオーバーフローを起こすので暗黙の型変換を使うことができません. 明示的な型変換を行うと、int型に格納される値は int型に格納できる最大の値となります. そして型変換における注意点が浮動小数点型から整数型への変換です. この変換は明示的な型変換でのみ変換が可能です. そして浮動小数点型から整数型への変換を行った場合値は四捨五入で丸められて格納されます.

配列、String 型

Java にも配列がありますがその実装方法が異なります.

```
int[] a = new int[5];
```

C とは全然書き方が違いますが詳細は今度やるので今回はこういうものだと思ってやってください. Java には String 型と呼ばれる文字列を扱う型があります. こちらは int 型や char 型などの型とは種類が違うので別に書きました. しかし今回は普通の型と同様の扱いで使っていきます. C 言語では文字列を扱う際には char 型の配列を用意して扱っていましたが Java には String 型と呼ばれる型がありこれは以下のように文字列を扱うことができます.

```
String str = “Hello World”;
```

とても便利な型となっています. ちなみにマルチバイト文字にも対応しています.

```
String str = “こんにちは”;
```

それでは以下のサンプルを打ち込んでみてください.

```
public class Sample4 {
	public static void main(String[] args){
		int a[] = {0,10,20,30,40,50,60,70,80,90};
		String str= "生麦生米生卵";

		for(int j=0;j<10;++j){
			System.out.println("a[" + j + "]="+a[j]);
			System.out.println("str="+str);
		}
	}
}
```

a[0]=0

str=生麦生米生卵
a[1]=10

str=生麦生米生卵

a[2]=20

str=生麦生米生卵
a[3]=30

str=生麦生米生卵

a[4]=40

str=生麦生米生卵
a[5]=50

str=生麦生米生卵

a[6]=60

str=生麦生米生卵
a[7]=70

str=生麦生米生卵

a[8]=80

str=生麦生米生卵
a[9]=90

str=生麦生米生卵

特に難しいこともありません. しかし、この二つはさっきも言ったように厳密には普通の型とは違います. どう違うのかは来週説明します.

オーバーロード(overload)

C では同じ名前の関数を複数作る事ができませんでした. Java ではメソッド名が同じでも引数の型や数が異なれば違うメソッドとして扱われます. つまり 1 つのクラスの中に名前が同じで引数の型や数が異なるメソッドを複数作る事ができます.

このように同名のメソッドを複数作る事をメソッド(関数)のオーバーロード(多重定義)といいます.

```
public class Sample5{
        void method(int a){};

        /*char method(int a){}*/   //返り値の型だけが違うのは NG
        char method(float a){}      //引数が違うとき返り値が違っても OK
        void method(double a){}   //引数の型が違うので OK
        /*void method(int b){}*/   //仮引数名だけが違のは NG
        void method(int a, int b){} //引数の数が違うので OK
        //次の 2 つは引数の順番が違うので OK
        void method(String s, char c){}
        void method(char c, String s){}
}
```

演習問題

1. 九九表を表示するプログラムを作成してください.

2. 整数の足し算をした結果を返す関数 static int sum(int a, int b)を作成しなさい. また double 型や String 型の足し算をした結果を返す関数を用意しなさい. ただし、関数名は int 型と同様 sum にすること. ※ static については今後の講座で解説します. とりあえずつけるようにしてください.

3. 10 個の整数値を昇順にバブルソートを用いて並び替えなさい. また、同様に 10 個の文字を昇順に並び替えなさい.

    - static void bubblesort(int[] array){ … }
    - static void bubblesort(char[] array){ … }

時間が余ったら・・・

演習問題の 2 をキーボードから値を入力できるようにしてみてください.

Java ではパッケージ、クラス、メソッド（関数）がたくさん用意されているので自分で調べてどんどん利用しましょう！