7-2.mainメソッドとJavaコマンドの仕様

mainメソッド

JavaSilverでは、mainメソッドの仕様についての問題も頻出です。
mainメソッドはこのような構造になっていました。

```
public static void main(String[] args) {
    //any code
}
```

javaを実行した時、PCはまずmainメソッドの場所を探し、そこから処理を開始します。
mainメソッドのこのような役割のことを、 **エントリーポイント** と呼びます。

mainメソッドが見つからない場合、コンパイルエラーとなります。
なので、プロジェクトに必ず1つ必要になります。

mainメソッドの構造

mainメソッドはいつも決まりごとのように上記の形で記述していますが、良く見てみると、構造は他のメソッドと同じです。

publicでstaticで戻り値はvoid(引数無し)のメソッド。
そして仮引数として、String配列型の変数argsを設定しています。

このmainメソッドの中で、自由に変更できるのは`args`の部分だけです。
メソッドの仮引数名はただの変数の名前であり、名前は何でもOKでしたね。
この変数名として、慣習的に`args`としているだけであり、実は自由で構いません。(argsはargumentsの略で、「引数」という意味)
なので、`(String[] aaa)`でも`(String[] bbb)`でも動作的には問題ありません。

可変長引数

また、`String[]`の表記についても1点注意があります。
String配列型であることは必須なのですが、以下のように、カッコの代わりにドットが3つ書かれている場合があります。

```
(String... args)
```

このような引数を、 **可変長引数** と呼びます。

可変長引数は、 **引数をいくつも受け取ることができる** 、という機能を持っています。
しかし、この機能は、すなわちほぼ普通の配列と違いはありません。
厳密には少し違いはありますが、JavaSilver試験では不要な知識なので、興味がある方は調べてみて下さい。
試験の中で見かけたら、表記がいつもと違うだけと認識すればOKです。
混乱しないようにしておきましょう！

可変長引数は、mainメソッド以外の通常のメソッドでも使用可能です。
ただし、そのメソッドの最後の引数に1つのみ設定出来ます

```
//例
static void Method(String a , Int... b) {}

//これは不可
static void Method(String... a , Int... b) {}
```

Javaコマンド

?

コマンドライン引数(起動パラメータ)

mainメソッドにも引数設定があることはわかりましたが、普段、mainメソッドに引数を渡しているコードは見当たらないですね。

mainメソッドに渡す引数は、java実行コマンドと一緒に入力します。
この値のことを **コマンドライン引数(または起動パラメータ)** と呼びます。

`java クラス名` でjavaのクラスファイルを実行できましたが、コマンドライン引数はその後ろにスペースで区切って指定します。

```
//コマンド
> java Sample a b c

//サンプルコード
public class Sample {
    public static void main(String[] args) {
        for (String arg : args) {
            System.out.println(arg);
        }
    }
}

//出力結果
a
b
c
```

java実行時の3つのコマンドライン引数a,b,cが、mainメソッドの引数argsに代入され、処理が行われます。

コマンドライン引数はeclipseでも設定できますが、試験の範囲とは外れるため、知りたい方は以下のリンクなど参照して下さい。

【eclipse】コマンドライン引数に値を設定するやり方を解説します【Java】

コマンドライン引数の注意点

試験問題には、少しややこしいコマンドライン引数の問題が出されます。

コマンドライン引数はスペースで区切りますが、 **値をダブルクォーテーションで囲むことで、スペースを含む1つの値として引数を渡すことができます。**
ダブルクォーテーション自体は文字として認識されません。
先程のサンプルコードを実行するコマンドを考えます。

```
> java Sample Hello World "Hello World"
Hello
World
Hello World
```

```
> java Sample "a"bc
abc
```

このように、ダブルクォーテーション自体は消え、1つの値が入ります。

ただし、ダブルクォーテーション記号の前に円マーク(¥)をつけることで、記号そのものを文字列として認識して渡すことができます。

```
> java Sample Hello World ¥"Hello World¥"
Hello
World
"Hello
World"
```

引数の内容や個数がまるで違ってきてしまいますので、よく見ながら頭の中で整理しましょう！

ソースファイルモード

Javaコマンドに関するその他の機能について解説します。

Javaのコードをコマンドラインから実行するには、javacコマンドでコンパイルした後、javaコマンドで実行を行っていました。

(例)

```
> java Sample.java
> javac Sample
```

しかしJava11から、javacコマンドでコンパイルせずにソースファイルを直接実行できる **ソースファイルモード** が追加されました。

(例)

```
> java Sample.java
```

javaコマンドでファイルを直接指定することで実行できます。
学習時など、とりあえずプログラムを動かす時に使用されるもので、本番の開発ではあまり用いられません。

コンパイルはしていないわけではなく、ディスク上ではなくメモリ上に暮らすファイルに相当するデータを出力しています。

また、javaにはpublicクラス名とファイル名は一致していなければならないルールがありました。
(Sample.javaファイルだったら、Sampleクラスでなければならない)

しかし、ソースファイルモードでは、それらが一致していなくても実行出来るという特徴がありますので覚えておきましょう。

その他のJavaコマンドのオプション等

試験問題のJavaコマンドの中で、時々オプションが記載されていることがあるので、主要なオプションについては少し目を通しておきましょう。

javacコマンドの主なオプション

オプション	概要
-sourcepath	ソースファイルの検索先を指定
-d	クラスファイルの出力先を指定
--module-path	モジュールの検索先を指定

javaコマンドの主なオプション

オプション	概要
-classpath -cp	クラスファイルの検索先を指定
--module-path	モジュールの検索先を指定
-jar	.jarファイルを実行

課題

満点を取れるまで小テストを受験して下さい。

制限時間: 30 分

評定方法: 最高評点

合格点: 100 / 100

受験件数: 136