Java Bronze-knowledge-01

Java Bronzeでの頻出知識（基本編）

前置き

Java Bronze試験でよく出てくる単語、問題、考え方などを書き残しています。（**業務で使うわけではない**）

目次

1. Javaエディション（Java SE、Java EE、Java ME）

他にも色々ありますが、とりあえずこれらを抑えておけば問題ないです。

Java SE(Java Standard Edition)

Javaを動かすための基本的な機能をまとめたもの。一番スタンダードなエディション。
Java SEでは以下の機能を提供している。

- JREの提供
  ⇒ Javaプログラムを実行するためのライブラリ、JVM※1 をまとめたもの。
- JDKの提供
  ⇒ 開発に必要なコンパイラやデバッガなど各種開発に必要なツールをまとめたもの。
- 標準クラスライブラリの提供
  ⇒ java.langやjava.utilなどの基本的なパッケージのこと。
  - java.lang
    ⇒ Boolean,Byteなどのクラス。
  - java.util
    ⇒ Collection,List,Mapなどのクラス。

※1 JVMとは
コンパイルしたclassファイルを実行するソフトウェアのこと。

Java EE(Java Enterprise Edition)

エンタープライズシステム※2 の開発に特化したエディション。エンタープライズアプリケーションで使用されているソフトウェアが相互に連携するための**基盤機能**の仕様やAPIを定めたものを指す。

基盤機能の一例

- 分散処理
  ⇒ 物理的に離れた場所にあるソフトウェアの機能をほかのソフトウェアが利用する機能。
- トランザクション管理
  ⇒ 一連のデータの処理が確実に行えることを保証する機能。

※2 エンタープライズシステムとは
数多くのソフトウェアが連携している大規模システムにおけるソフトウェア群のこと。

Java ME(Java Micro Edition)

携帯電話やPDAなどの携帯端末、工業用ロボット、プリンタなど（組み込み系）のハードウェアを制御するソフトウェアを作るためのエディション。
通常のアプリケーションと異なり、ハードウェア要件に制約があるため、それに対応するための**KVM**と呼ばれる仮想マシンも用意されている。

2. データ型（プリミティブ型）

データ型は大きく分けて二つの分類があり、そのうちの**プリミティブ型**は特に覚えることが多いので、以下の一覧表を覚えておくことをお勧めします。（指定されたデータ型の変数に代入できるかどうか、よく試験に出てきます）

分類	データ型	保持できる値
整数型	byte	8ビット整数 -128～127
	short	16ビット整数 -32,768～32,767
	int	32ビット整数 -2,147,483,648～2,147,483,647

分類	データ型	保持できる値
	long	64ビット整数 -9,223,372,036,854,775,808～9,223,372,036,854,775,807
浮動小数点数型	float	32ビット符号付き浮動小数点数
	double	32ビット符号付き浮動小数点数
文字型	char	16ビットUnicode文字 ¥u0000～¥uFFFF
真偽値	boolean	true,false

また、charについては値の代入方法や代入可能な値が特殊であるため、この表のデータ型の中でも特に試験に出てきます。

出題例：変数の初期化方法のうち正しいものを選択する。

```
char char01 = 'a';          // 代入する際はシングルクォーテーションを使用すること。
char char02 = 'あ';         // シングルクォーテーションを使用しているため代入可能。
char char03 = '¥u0000';     // Unicode表記のため代入可能。
char char04 = 97;           // Unicode変換されて代入される。9の場合、「a」が割り当てられている。

char char05 = "a";          // コンパイルエラー（ダブルクォーテーションを使用しているため）
char char06 = 'ab';         // コンパイルエラー（1文字でないため）
```

3. 演算子（++, --）

インクリメント演算子（++）とデクリメント演算子（--）は「1加算する。または1減算する」演算子になりますが、前置と後置で意味が変わってくるので要注意です。

```
int num01 = 0;
int num02 = 0;

// インクリメント演算子（前置）
System.out.println(num01);    //「0」
System.out.println(++num01);  //「1」加算処理 ⇒ 出力処理
System.out.println(num01);    //「1」

// インクリメント演算子（後置）
System.out.println(num02);    //「0」
System.out.println(num02++);  //「0」出力処理 ⇒ 加算処理
System.out.println(num02);    //「1」
```

4. 分岐文（if）

if文を記述する際に記述する波括弧は、実は省略することができます。以下の二つの分岐文はどちらも同じ意味になります。

```
int num01 = 0;

// if文（括弧あり）
if(num01 == 0){
    System.out.println("AAA");  // 実行される
}

// if文（括弧なし）
if(num01 == 0)
    System.out.println("AAA");  // 実行される
```

これは、「括弧を省略すると、条件式の結果がtrueの場合に次の1文のみが実行される」というJavaの仕様のためです。
つまり、以下のような場合出力結果は「BBB」のみとなります。

```
// if文（括弧なし）
if(false)  // false
    System.out.println("AAA");  // 実行されない
    System.out.println("BBB");  // 実行される
```

5. ショートサーキット演算子

演算子の左オペランドの結果によって、右オペランドを評価しない演算子のことを**ショートサーキット演算子**といいます。以下の例をもとに解説します。

```
int num01 = -1;
int num02 = 5;

// 右オペランド未評価
if(num01 >= 0 && num02 >= 0){   // 左オペランド：false、右オペランド：未評価
    System.out.println("AAA");
}
```

後ろの条件を評価してもしなくても結果が変わらない場合は実行されないというだけの話になります。
が、試験では、「3. 演算子」で書いたインクリメント演算子と融合した問題が出題されるケースがあります。（間違いやすいです）

```
int num01 = 0;
int num02 = 0;
int num03 = 0;
int num04 = 0;

// 右オペランド：未評価
if(num01++ > 0 && num02++ > 0){    // 左オペランド：false、右オペランド：未評価
    System.out.println("AAA");  // 実行されない
}
System.out.println("num01: " + num01);  // 「num01: 1」
System.out.println("num02: " + num02);  // 「num02: 0」 ※未評価のため

// 右オペランド：評価
if(++num03 >= 0 && ++num04 >= 0){    // 左オペランド：true、右オペランド：true
    System.out.println("BBB");
}
System.out.println("num03: " + num03);  // 「num03: 1」
System.out.println("num04: " + num04);  // 「num04: 1」
```

ループ文（for）

以下のfor文を実装した場合、コンパイルエラーにはならず正常に実行されます。
ただし、条件式の定義がないため、for文から抜け出すことができません。

```
// 無限ループ
for(int i = 0; ; i++){  // 条件式：定義なし
    System.out.println("LOOP");  // 実行される
}
```