オブジェクト指向と
ゲームプログラミング

○　Java 入門編 － 第４回　コメント、変数、演算子　○

コメント

Javaのコメントは、C++のように、複数行を「/*」と「*/」で囲む複数行コメントと、ダブルスラッシュの右側すべてをコメントとして扱う１行コメントがあります。また、Java独自のコメントとして、ドキュメントコメントがあります。

ドキュメントコメントは、「/**」で始まり「*/」で終わる特別なコメントです。この方法で記述されたコメントは、javadocというユーティリティで自動的にドキュメントを作成することができます。javadocでは、HTMLフォーマットでドキュメントを出力することもできます。

変数

Javaの変数には、以下の８つの型があります。C／C++のようなunsignedによる符号無しへの拡張はできません。また、実行環境がJavaVMに統一されているため、環境によって型の範囲やサイズが異なるということはありません。

型名	説明	値の範囲
boolean	真偽値を格納	true, false
char	unicode文字を格納	\u0000 ～ \uffff
byte	１バイト整数	-128 ～ 127
short	２バイト整数	-32,768 ～ 32,767
int	４バイト整数	-2,147,483,648 ～ 2,147,483,647
long	８バイト整数	-92,233,372,036,854,775,808 ～ 92,233,372,036,854,775,807
float	単精度浮動小数点(４バイト)	±10の±38乗(有効桁数9桁)
double	倍精度浮動小数点(８バイト)	±10の±308乗(有効桁数16桁)

名前

変数およびクラス、メソッドの名前に使用できる文字は、アルファベット、0～9、$, _です。その他の文字、記号、スペースやタブなどは使用できません。また、名前の先頭が0～9だったり、予約語とまったく同じ名前もエラーになります。アルファベットの大文字と小文字は区別され、別の文字として扱われます。

- 使用できる文字は、英字A～Zとa～z、数字0～9、記号"$"と"_"
- 変数名の先頭が数字から始まってはいけない
- 予約語とまったく同じ綴りは使用できない
- 英字の大文字と小文字は別の文字として扱われる

予約語

プログラム中で、特別な意味を持つキーワードがあり、これらを名前として使うことはできません。これらの特別なキーワードは、あらかじめ「予約」されているため、予約語と呼ばれます。

Javaの予約語は、以下のようになっています。

abstract	boolean	break	byte	cast	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	goto	if	implements	import	instanceof
int	instance	long	native	new	null
package	private	protected	public	return	short
static	super	switch	syncronized	this	throw
throws	transient	try	void	volatile	while

予約語の一部を大文字に変えれば、予約語と同じ綴りでも名前として使用できます。しかし、たいへん紛らわしいので、やめたほうが無難です。なお、名前の一部に予約語が入るのは問題がありません。

演算子

Javaの演算子は、C／C＋＋とほとんど同じです。Java独自の演算子としては、「>>>=」と「instanceof」演算子があります。

>>>=演算子は、論理右シフトを行います。論理右シフトは、符号を無視したシフトで、桁を右にずらしたときに空いた上位桁に0を入れます。この演算子が使用できるのは、intとlongだけです。

instanceof演算子は、以下のような動作をするもので、ポリモーフィズムを用いる場合に使用します。

- 左辺で指定されたオブジェクト変数が右辺で指定されたクラスのインスタンスであればtrueを返す
- 右辺の指定がインターフェイスだった場合は、オブジェクトが指定インターフェイスを実装しているかを確認し、オブジェクトが指定インターフェイスをインプリメントしていればtrueを返す
- いずれにも該当しない場合は、falseを返す

演算子名	記号	名　前	記述例	機　能
算術演算子	++	インクリメント	a++	値に1を加える
	--	デクリメント	a--	値から1を引く
	+	正符号	+a	正符号
	-	負符号	-a	負符号
	*	乗算	a * b	乗算
	/	除算	a / b	除算
	%	剰余	a % b	割り算の余りを求める。
	+	加算、文字列の連結	a + b	加算、両辺が文字列なら連結する
	-	減算	a - b	減算

演算子名	記号	名　前	記述例	機　能
ビット演算子	~	ビットごとの否定	~ a	ビットごとの論理否定を求める
	<<	左シフト	a << b	ビットを左にシフトする
	>>	算術右シフト	a >> b	ビットを右にシフトする
	>>>	論理右シフト	a >>> b	ビットを右にシフトする。先頭は0で埋められる
	&	ビットごとの論理積	a & b	ビットごとの論理積を求める
	^	ビットごとの排他的論理和	a ^ b	ビットごとの排他的論理和を求める
	¦	ビットごとの論理和	a ¦ b	ビットごとの論理和を求める

演算子名	記号	名　前	記述例	機　能
関係演算子	==	等値	a == b	両辺が等しければtrue、そうでなければfalseとなる
	!=	非等値	a != b	両辺が等しくなければtrue、そうでなければfalseとなる
	<	左不等	a < b	左辺より右辺が大きければtrue、そうでなければfalseとな
	<=	等価左不等	a <= b	左辺が右辺以上ならtrue、そうでなければfalseとなる
	>	右不等	a > b	左辺より右辺が小さければtrue、そうでなければfalseとな
	>=	等価右不等	a >= b	左辺が右辺以下ならtrue、そうでなければfalseとなる

演算子名	記号	名　前	記述例	機　能
論理演算子 (boolean型に適用)	!	論理否定	!a	trueならfalse、falseならtrueになる
	&	論理積	a & b	両辺がtrueのときのみtrueとなる
	^	排他的論理和	a ^ b	両辺が異なるときのみtrueとなる
	¦	論理和	a ¦ b	少なくとも一方がtrueならtrueとなる
	&&	条件AND	a && b	両辺がtrueのときのみtrueとなる
	¦¦	条件OR	a ¦¦ b	少なくとも一方がtrueならtrueとなる

演算子名	記号	名　前	記述例	機　能
代入演算子	=	単純代入	a = b	右辺の式の値を左辺に代入する
	/=	除算代入	a /= b	a = a / bと同じ
	%=	余剰代入	a %= b	a = a % bと同じ
	*=	乗算代入	a *= b	a = a * bと同じ
	+=	加算代入	a += b	a = a + bと同じ
	-=	減算代入	a -= b	a = a - bと同じ
	<<=	左シフト代入	a <<= b	a = a << bと同じ
	>>=	算術右シフト代入	a >>= b	a = a >> bと同じ
	>>>=	論理右シフト代入	a >>>= b	a = a >>> bと同じ
	&=	ビット積代入	a &= b	a = a & bと同じ
	^=	ビット差代入	a ^= b	a = a ^ bと同じ
	¦=	ビット和代入	a ¦= b	a = a ¦ bと同じ

演算子名	記号	名　前	記述例	機　能
その他	(型名)	型変換(キャスト)	(int)a	型変換を行う
	instanceof		a instanceof b	aがbのインスタンスならtrue、そうでなければfalseとなる
	? :	条件	a ? b : c	aがtrueならb、aがfalseならcとなる

定数フィールド

Javaの定数の定義方法は以下の２通りあります。

・final staticを使用
・enumを使用

final staticを使う方法は、以下のように、finalをつけて変数を宣言することにより、書き換えることのできない変数を定数とする方法です。

```
class Test {
    public final static int SCREEN_WIDTH  = 240;    // 画面幅
    public final static int SCREEN_HEIGHT = 240;    // 画面高さ
}
```

インタフェースの場合は、すべてのフィールドに「public static final」が宣言されているものと見なされるので、以下のような文法が推奨されています。

```
interface Test {
    int SCREEN_WIDTH  = 240;    // 画面幅
    int SCREEN_HEIGHT = 240;    // 画面高さ
}
```

enumは、Java2 5.0(J2SE 1.5.0)で新規追加されたものです。Javaのenumは以下の特徴を持ち、C／C++のそれとは若干異なります。enumを使用すると、型に安全な定数を列挙型として定義することが簡単にできるようになります。

・enumはクラス
・enumはjava.lang.Enumを暗黙的に継承
・個々の列挙値は整数ではない
・enumはpublicなコンストラクタを持たない
・enumの値はpublic、static、finalの属性を持つ
・enumはjava.util.Comparableを実装する
・enumはtoStringメソッドをオーバーライドする
・enumはvalueOfメソッドを提供する
・enumはfinal ordinalメソッドを提供する
・enumはvaluesメソッドを提供する

enumは以下のように宣言します。

```
enum GameScene {
    LOGO,
    TITLE,
    STAGE1,
    STAGE2,
    WISEMAN,
    ENDING,
}
```

enumで宣言された定数は、「クラス名. 定数名」でアクセスします。これは、staticであるためです。たとえば、上のGameSceneのLOGOを使う場合は、「GameScene.LOGO」となります。なお、case文の場合のみ、クラス名を省略して「LOGO」のように記述できます。