オブジェクト指向と
ゲームプログラミング

○　Java 入門編 － 第７回　文字列　○

文字列

Javaで文字列を扱うには、java.langパッケージのStringクラスを使用します。

C／C++では、char型の配列に末尾を'\0'止めした文字列を使用していましたが、JavaのStringオブジェクトは、末尾に'\0'はなく、終端文字について意識することなく文字列を操作できるようになっています。また、unicodeを採用しており、char型と同様に、半角も全角も１文字として扱います。
Stringオブジェクトでは、配列と同じように、文字列にアクセスするときには境界のチェックがつねに行われるため、オブジェクトが確保した領域の外に飛び出してしまうことはありません。さらに、Stringオブジェクトには、文字列を操作するさまざまなメソッドが用意されています。

Stringオブジェクトは、他のオブジェクトと違いnewを使わずに作成することができます。文字列をString型の変数に代入するだけで、インスタンスが作成されます。

```
String   str;
str = "Java Programming";
```

１行目は、Stringオブジェクトのstrの宣言です。２行目でインスタンスが作成され、文字列"Java Programming"で初期化されます。まとめて１行にすることもできます。

```
String   str = "Java Programing";
```

他のオブジェクトと同じように、newを使ってインスタンスを作成することもできます。

```
String   str = new String("Java Programming");
```

文字列の長さは、Stringクラスのlengthメソッドで取得できます。

```
String   str    = "Java Programing";
int      strlen = str.length();          // strlenは15になる
```

Javaでは、""で囲まれた文字列をStringオブジェクトとして扱います。この性質を利用すると、以下のようなこともできます。

```
int    strlen = "Java Programimng".length();    // strlenは文字列長16が代入
char[] array  = "KTU".toCharArray();            // 文字列をchar配列に変換
```

文字列の操作

・文字列の結合
Javaでは、+演算子による文字列の結合ができます。文字列同士の結合だけでなく、文字列とデータ型も結合することができます。

```
String   str1 = "変数iの内容は";
String   str2 = "です";
int      i    = 100;

str1 = str1 + i + str2;
```

この場合、オブジェクトstr1は、str1とStringオブジェクトに変換された変数iそしてオブジェクトstr2が結合され、「変数iの内容は100です」となります。

Stringクラスにはconcatというメソッドがあり、文字列同士の結合が可能です。+演算子のようにデータ型との結合はできません。

```
String   str1 = "ABC";
String   str2 = "DEF";

str1 = str1.concat(str2);
```

この場合、オブジェクトstr1はオブジェクトstr2と結合され、「ABCDEF」となります。

・char型配列への変換
　文字列の処理内容によっては、char型の配列を使いたい場合があります。Stringクラスには、文字列をchar型配列に変換するtoCharArrayメソッドがあります。

```
String   str1  = "ABCDEFG";
char[]   array = str1.toCharArray();
```

　この場合、文字列「ABCDEFG」が配列arrayに１文字ずつ分解されて格納されます。配列arrayの各要素は演算が可能です。なお、文字列の最後に'\0'は入りません。

```
array[0] += 1; // 'A'に1がプラスされ'B'になる
array[1] -= 1; // 'B'から1がマイナスされ'A'になる
```

　Stringオブジェクトには、charAtというメソッドもあり、特定の文字をchar型として取り出すことができます。

```
String   str1 = "ABCDEFG";
char     ch1   = str1.charAt(0);    // ch1は'A'
char     ch2   = str1.charAt(5);    // ch2は'F'
```

　charAtメソッドには取り出す位置(オフセット)を指定します。文字列の先頭は0です。

　char型の配列をStringオブジェクトに変換したい場合は、Stringクラスのコンストラクタを使います。

```
char     char_array = {'A', 'B', 'C', 'D', 'E', 'F', 'G'};
String   str1 = new String(char_array);
```

　Stringクラスは、オーバーロードされたコンストラクタが用意されており、さまざまなデータ型をStringオブジェクトに変換できます。

・Stringオブジェクトのコピー
　Stringオブジェクトを他のStringオブジェクトに=演算子で代入しても、コピーが作成されるわけではありません。=演算子では、同じ領域を指すオブジェクトが作成されます。

```
String   str1, str2;

str1 = "ABCDEFG";
str2 = str1;
```

　この場合は、オブジェクトstr1とstr2は名前こそ違うものの、同一のメモリ領域に作成されたStringオブジェクトを指しています。str2を書き換えると、str1も書き変わります。独立したコピーは、newを使って作成します。

```
String   str1, str2;

str1 = "ABCDEFG";
str2 = new String(str1);
```

　こうすると、オブジェクトstr1とstr2は、同じ「ABCDEFG」という文字列を持ちますが、別の領域に作成されます。

・Stringオブジェクトの比較
　ソートなどで、文字列の比較が必要になる場合があります。２つの文字列を==演算子で比較すると、同じ領域を指すかどうかという比較になります。文字列自体の比較は、Stringクラスのequalsメソッドで行います。大文字と小文字を区別しないで比較したい場合は、equalsIgnoreCaseメソッドを使います。

```
String   str1, str2, str3, str4;

str1 = "ABCDEFG";
str2 = str1;
str3 = new String(str1);
str4 = "abcdefg";

boolean   b1 = (str1 == str2);              // 真
boolean   b2 = (str1 == str3);              // 偽
boolean   b3 = str1.equals(str2);           // 真
boolean   b4 = str1.equalsIgnoreCase(str4); // 真
```

辞書順に文字列を比較する場合は、compareToメソッドを使います。<や>演算子では比較できません。このメソッドは、オブジェクトが保持する文字列と引数の文字列を辞書順に比較し、自分の方が小さい場合は負、等しい場合はゼロ、大きい場合は正の値を返します。大文字と小文字は区別され、大文字の方が大きいものとして扱われます。区別しない場合は、compareToIgnoreCaseメソッドを使います。

```
String  str1, str2, str3;

str1 = "ABCDEFG";
str2 = "VWXYZ";
str3 = "ABCDEFG";

int   res1 = str1.compareTo(str2);  // res1は負
int   res2 = str1.compareTo(str3);  // res2はゼロ
int   res3 = str2.compareTo(str1);  // res3は正
```

・Stringクラスの主なメソッド

これまでに紹介したメソッド以外にも、Stringクラスにさまざまなメソッドが用意されています。主なものを以下に紹介します。

・コンストラクタ

プロトタイプ	機　能
String()	Stringオブジェクトを生成し、空の文字列で初期化する
String(char[] value)	Stringオブジェクトを生成し、文字配列valueで初期化する
String(char[] value, int offset, int count)	Stringオブジェクトを生成し、文字配列valueのoffset以降からcountで指定される文字数で初期化する
String(String value)	Stringオブジェクトを生成し、文字列valueで初期化する
String(StringBuffer buffer)	Stringオブジェクトを生成し、文字列bufferで初期化する

・インスタンスメンバ

プロトタイプ	機　能
boolean endsWith(String suffix)	文字列が文字列suffixで終わるかを返す
boolean equals(Object anObject)	オブジェクトanObjectが同一文字列を表すStringオブジェクトであるかを返す。大文字と小文字を区別する。
boolean equalsIgnoreCase(String AnotherString)	文字列anotherStringと同一文字列であるかを返す。大文字小文字を区別しない
boolean regionMatches(int toffset, String other, int ooffset, int len)	位置toffset以降の文字列と、文字列otherの位置ooffset以降の文字列が、lenで指定される文字数分一致しているかを返す。大文字と小文字は区別される
boolean regionMatches(boolean igneoreCase, int toffset, String other, int ooffset, int len)	位置toffset以降の文字列と、文字列otherの位置ooffset以降の文字列が、lenで指定される文字数分一致しているかを返す。ignoreCaseがfalseなら大文字と小文字は区別され、trueなら区別されない
boolean startsWith(String prefix)	文字列が文字列prefixで始まるかを返す
char charAt(int index)	位置indexにある文字を返す
char[] toCharArray()	文字列をchar配列に変換する
int compareTo(String anotherString)	文字列anotherStringと辞書順で比較する。大文字と小文字を区別する
int compareToIgnoreCase(String str)	文字列strと辞書順で比較する。大文字と小文字を区別しない
int indexOf(int ch)	文字chが最初に出現する位置を返す
int indexOf(int ch, int fromIndex)	位置fromIndex以降で文字chが最初に出現する位置を返す
int indexOf(String str)	文字列strが最初に出現する位置を返す
int indexOf(String str, int fromIndex)	位置fromIndex以降で文字列strが最初に出現する位置を返す
int lastIndexOf(int ch)	文字chが最後に出現する位置を返す
int lastIndexOf(int ch, int fromIndex)	位置fromIndex以降で文字chが最後に出現する位置を返す
int lastIndexOf(String str)	文字列strが最後に出現する位置を返す
int lastIndexOf(String str, int fromIndex)	位置fromIndex以降で文字列strが最後に出現する位置を返す
int length()	文字列の長さを返す
String concat(String str)	文字列の最後に文字列strを結合する
String replace(char oldChar, char newChar)	文字列内の文字oldCharをすべてnewCharで置き換える
String substring(int beginIndex)	位置beginIndex以降の文字列を取り出す
String substring(int beginIndex, int endIndex)	位置beginIndexからendIndex - 1までの文字列を取り出す
String toLowerCase()	文字列内の大文字をすべて小文字に変換する
String toUpperCase()	文字列内の小文字をすべて大文字に変換する
String trim()	文字列の両端の空白を取り除く

・静的メンバ

プロトタイプ	機　能
static String valueOf(boolean b)	boolean型変数bを文字列に変換する
static String valueOf(char c)	char型変数cを文字列に変換する
static String valueOf(char[] data)	文字配列dataを文字列に変換する
static String valueOf(char[] data, int offset, int count)	文字配列dataの位置offsetからcountで指定される文字数を文字列に変換する
static String valueOf(double d)	double型変数dを文字列に変換する
static String valueOf(float f)	float型変数fを文字列に変換する
static String valueOf(int i)	int型変数iを文字列に変換する
static String valueOf(long l)	long型変数lを文字列に変換する

StringBufferクラス

Stringクラスは、いったんオブジェクトを作成すると、オブジェクトの内容を直接変更することはできません。文字列を挿入したり削除したりする場合は、java.langパッケージのStringBufferクラスを使います。
StringBufferクラスには、文字列中の１文字を編集したり、文字列の挿入や削除を行うメソッドが用意されています。また、文字列を格納するバッファのサイズを自由に変更することができます。

StringクラスよりStringBufferクラスの方が利点がたくさんあるため、つねにStringBufferクラスを使用したくなりますが、特別な理由がない限り、Stringクラスを使うようにします。なぜなら、Javaでは""で囲まれた文字列はStringオブジェクトとして扱われ、また、メソッドの引数が文字列を受け取る場合、ほとんどがString型であるからです。
StringクラスとStringBufferクラスには､双方のクラスに変換を行うメソッドが用意されているので、文字列の処理内容に合わせてそれぞれのクラスを使用することができます。内容が変更されない文字列にはStringオブジェクトを使い、内容やバッファサイズを変更する必要がある文字列にはStringBufferオブジェクトを使います。

StringBufferオブジェクトの作成

StringBufferオブジェクトのインスタンスは、Stringオブジェクトのように=演算子で作成することはできません。以下のように記述してもエラーになります。

```
StringBuffer   strbuf = "ABC";  // エラー！
```

StringBufferオブジェクトは、newでコンストラクタを呼び出して作成します。StringBufferクラスには、３つのコンストラクタがあります。

```
String         str = "ABCDEFG";

StringBuffer   strbuf1 = new StringBuffer();       // 16文字の空のバッファ
StringBuffer   strbuf2 = new StringBuffer(32);     // 32文字の空のバッファ
StringBuffer   strbuf3 = new StringBuffer(str);    // strで初期化されたバッファ
StringBuffer   strbuf4 = new StringBuffer("KTU"); // 文字列"KTU"で初期化されたバッファ
```

変数strbuf1は、引数がないのでデフォルトコンストラクタが起動されます。デフォルトコンストラクタは、16文字分の空の文字列バッファを作成します。
変数strbuf2は、引数が整数なので、コンストラクタは指定された文字数分の空のバッファを作成します。
変数strbuf3とstrbuf4は、引数がStringオブジェクトなので、コンストラクタは指定された文字列で初期化したバッファを作成します。

StringBufferオブジェクトとStringオブジェクトの変換

StringBufferクラスのtoStringメソッドは、StringBufferからStringに変換します。また、StringクラスにはStringBufferオブジェクトを引数とするコンストラクタがあり、StringBufferオブジェクトからStringオブジェクトを作成することができます。

```
String         str1    = "ABCD";
StringBuffer   strbuf1 = new StringBuffer(str1); // StringBufferオブジェクトを作成
String         str2    = strbuf1.toString();     // StringBufferからStringへ変換
String         str3    = new String(strbuf1);    // Stringオブジェクトを作成
```

StringBufferオブジェクトの編集

StringBufferクラスでは、文字列の内容を編集することができます。appendメソッドは、さまざまなデータ型を文字列に変換し、バッファに追加することができます。insertメソッドは、さまざまなデータ型を文字列に変換し、バッファの任意の場所に挿入することができます。deleteメソッドは、任意の場所の文字列を削除することができます。また、setCharAtメソッドで文字列中の１文字を置き換えることができます。

```
StringBuffer   strbuf = new StringBuffer("ABCDEF");
char[]         chstr  = {'X', 'Y', 'Z'};
```

```
strbuf.append(100);          // int型の結合
strbuf.insert(2, chstr);     // char配列の挿入
strbuf.delete(6, 9);         // 文字列の削除
strbuf.setCharAt(6, '2');    // 文字の置き換え
```

StringBufferのサイズ取得

StringBufferオブジェクトが格納できる文字列のサイズは、capacityメソッドで取得できます。このサイズは、格納されている文字列の長さに関わらず、setLengthメソッドで自由に変更することができます。文字列自体の長さは、lengthメソッドで取得できます。

```
StringBuffer   strbuf = new StringBuffer("ABCDEF");

int   bufsize = strbuf.capacity();      // バッファサイズの取得
strbuf.setLength(4);                    // バッファの長さを４に設定
int   strlen  = strbuf.length();        // 文字列の長さを取得
```

StringBufferクラスの主メソッド

StringBufferクラスの主なメソッドを以下に紹介します。

・コンストラクタ

プロトタイプ	機　能
StringBuffer()	StringBufferオブジェクトを生成し、16文字分の空の文字列バッファを作成する
StringBuffer(int length)	StringBufferオブジェクトを生成し、lengthで指定される文字数分の空の文字列バッファを作成する
StringBuffer(String str)	StringBufferオブジェクトを生成し、文字列バッファを文字列strで初期化

・インスタンスメンバ

プロトタイプ	機　能
char charAt(int index)	位置indexにある文字を返す
int capacity()	文字列バッファのサイズを返す
int length()	文字列の長さを返す
String substring(int start)	位置start以降の文字列を取り出す
String substring(int start, int end)	位置startからend - 1までの文字列を取り出す
String toString()	文字列バッファの内容をStringオブジェクトに変換する
StringBuffer append(boolean b)	boolean型変数bを文字列に変換して文字列バッファの最後に追加する
StringBuffer append(char c)	char型変数cを文字列に変換して文字列バッファの最後に追加する
StringBuffer append(char[] str)	char配列strを文字列に変換して文字列バッファの最後に追加する
StringBuffer append(char[] str, int offset, int len)	char配列strの位置offset以降を、lenで指定される分だけ文字列に変換して文字列バッファの最後に追加する
StringBuffer append(double d)	double型変数dを文字列に変換して文字列バッファの最後に追加する
StringBuffer append(float f)	float型変数fを文字列に変換して文字列バッファの最後に追加する
StringBuffer append(int i)	int型変数iを文字列に変換して文字列バッファの最後に追加する
StringBuffer append(long l)	long型変数lを文字列に変換して文字列バッファの最後に追加する
StringBuffer append(Object obj)	オブジェクトobjを文字列に変換して文字列バッファの最後に追加する
StringBuffer append(String str)	文字列strを文字列バッファの最後に追加する
StringBuffer delete(int start, int end)	位置startからend - 1までの文字を削除し、文字列バッファを切りつめる
StringBuffer deleteCharAt(int index)	位置indexにある文字を削除し、文字列バッファを切りつめる
StringBuffer insert(int offset, boolean b)	boolean型変数bを文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int offset, char c)	char型変数cを文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int offset, char[] str)	char配列strを文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int index, char[] str, int offset, int len)	char配列strの位置index以降を、lenで指定される分だけ文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int offset, double d)	double型変数dを文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int offset, float f)	float型変数fを文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int offset, int i)	int型変数iを文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int offset, long l)	long型変数lを文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int offset, Object obj)	オブジェクトobjを文字列に変換して文字列バッファの位置offsetに挿入する
StringBuffer insert(int offset, String str)	文字列strを文字列バッファの位置offsetに挿入する
StringBuffer replace(int start, int end, String str)	文字列バッファの位置startからend - 1にある文字を削除し、文字列strを挿入する
StringBuffer reverse()	文字列を逆の並び順にする
void setCharAt(int index, char ch)	位置indexにある文字を文字chに置き換える
void setLength(int newLength)	文字列バッファのサイズをnewLengthに設定する