Java プログラミング入門

— 文字・文字列 —

早稲田大学

文字（1）

char 型

- char 型は，“16 ビットの符号なし整数型” と定義されている
- 文字が整数値として扱われていることを意味する
- 文字はすべて “文字コード” にしたがって整数値で表される
- プログラム中で文字を表す場合は，その文字を ’ ’ で囲む

```
char c1 = 'a', c2 = 97;

System.out.println("c1 : " + c1);
System.out.println("c2 : " + c2);
```

```
c1 : a
c2 : a
```

String 型

- 文字列を扱うための型
- 変数の基本型（int 型，double 型，char 型等）とは異なり，クラスとして定義されている
- 文字列を表す場合は，その文字列を " " で囲む

String 型の宣言

```
String s1;
String s2 = "efgh";
```

String 型変数の使用

```
s1 = "abcd";                    — s1 に文字列"abcd"を代入
System.out.println(s1);
s1 = s1 + s2;                   — 文字列 s1 に文字列 s2 を連結
System.out.println(s2);
```

String 型

- 文字列を扱うための型
- 変数の基本型（int 型，double 型，char 型等）とは異なり，クラスとして定義されている
- 文字列を表す場合は，その文字列を " " で囲む

String 型の宣言

```
String s1;
String s2 = "efgh";
```

String 型変数の使用

```
s1 = "abcd";                    — s1 に文字列"abcd"を代入
System.out.println(s1);
s1 = s1 + s2;                   — 文字列 s1 に文字列 s2 を連結
System.out.println(s2);
```

String 型

- 文字列を扱うための型
- 変数の基本型（int 型，double 型，char 型等）とは異なり，クラスとして定義されている
- 文字列を表す場合は，その文字列を " " で囲む

String 型の宣言

```
String s1;
String s2 = "efgh";
```

String 型変数の使用

```
s1 = "abcd";                    — s1 に文字列"abcd"を代入
System.out.println(s1);
s1 = s1 + s2;                   — 文字列 s1 に文字列 s2 を連結
System.out.println(s2);
```

文字列のキーボードからの読み込み

```
public static String inputString() {
  String s = "";

  try {
    InputStreamReader isr = new InputStreamReader(System.in);
    BufferedReader br = new BufferedReader(isr);
    s = br.readLine();
  }
  catch(Exception e) {
    System.out.println(e);
    System.exit(1);
  }
  return s;
}
```

```
String s;

System.out.print("Input a String : ");
s = inputString();
```

String 型のメソッド（1）

- String 型はクラスである
- クラスには利用するためのメソッドが準備されている
- String 型のクラスにも，文字列処理のための便利なメソッドが存在する
- 使用法は，“変数名 . メソッド名”

メソッド
int length()
char charAt(int index)
int indexOf(String str)
boolean equals(Object anObject)
boolean startsWith(String str)
boolean endsWith(String str)
String toLowerCase()
String toUpperCase()
String substring(int begin, int end)
String replace(char old, char new)
char[] toCharArray()

String 型のメソッド（1）

- String 型はクラスである
- クラスには利用するためのメソッドが準備されている
- String 型のクラスにも，文字列処理のための便利なメソッドが存在する
- 使用法は，“変数名 . メソッド名”

メソッド
`int length()`
`char charAt(int index)`
`int indexOf(String str)`
`boolean equals(Object anObject)`
`boolean startsWith(String str)`
`boolean endsWith(String str)`
`String toLowerCase()`
`String toUpperCase()`
`String substring(int begin, int end)`
`String replace(char old, char new)`
`char[] toCharArray()`

String 型のメソッド（2）

```
String s1 = "abcdefghijklmn";
```

length()

文字列の長さ（文字数）を返す.

```
int n;
n = s1.length();
System.out.println(n);
```

```
14
```

charAt(n)

文字列の n+1 番目の文字を返す. 先頭を 0 番目と数える.

```
char c;
c = s1.charAt(5);
System.out.println(c);
```

```
f
```

String 型のメソッド（2）

```
String s1 = "abcdefghijklmn";
```

length()

文字列の長さ（文字数）を返す.

```
int n;
n = s1.length();
System.out.println(n);
```

```
14
```

charAt(n)

文字列の n+1 番目の文字を返す．先頭を 0 番目と数える.

```
char c;
c = s1.charAt(5);
System.out.println(c);
```

```
f
```

String 型のメソッド（2）

```
String s1 = "abcdefghijklmn";
```

length()

文字列の長さ（文字数）を返す.

```
int n;
n = s1.length();
System.out.println(n);
```

```
14
```

charAt(n)

文字列の n+1 番目の文字を返す．先頭を 0 番目と数える.

```
char c;
c = s1.charAt(5);
System.out.println(c);
```

```
f
```

String型のメソッド（3）

```
String s1 = "abcdefghijklmn";
```

indexOf(s)

文字列 s が含まれるかを調べ，含まれていなければ “-1” を返す．

```
int n;
n = s1.indexOf("klm");
System.out.println(n);
```

```
10
```

equals(s)

文字列 s と等しいかどうかを調べる．“true” または “false” を返す．

```
String s2 = "abcdefghij";
boolean b;
b = s1.equals(s2);
System.out.println(b);
```

```
false
```

String 型のメソッド（3）

```
String s1 = "abcdefghijklmn";
```

indexOf(s)

文字列 s が含まれるかを調べ，含まれていなければ “-1” を返す．

```
int n;
n = s1.indexOf("klm");
System.out.println(n);
```

```
10
```

equals(s)

文字列 s と等しいかどうかを調べる．“true” または “false” を返す．

```
String s2 = "abcdefghij";
boolean b;
b = s1.equals(s2);
System.out.println(b);
```

```
false
```

String 型のメソッド（3）

```
String s1 = "abcdefghijklmn";
```

indexOf(s)

文字列 s が含まれるかを調べ，含まれていなければ “-1” を返す．

```
int n;
n = s1.indexOf("klm");
System.out.println(n);
```

```
10
```

equals(s)

文字列 s と等しいかどうかを調べる．“true” または “false” を返す．

```
String s2 = "abcdefghij";
boolean b;
b = s1.equals(s2);
System.out.println(b);
```

```
false
```

String 型のメソッド（4）

```
String s1 = "abcdefghijklmn";
```

toUpperCase()

文字列の各文字を大文字に変換し，変換した新たな文字列を返す．

```
String s2;
s2 = s1.toUpperCase();
System.out.println(s2);
```

```
ABCDEFGHIJKLMN
```

substring(m,n)

文字列の m+1 番目から n 番目の部分を返す（先頭は 0 番目）．

```
String s2;
s2 = s1.substring(4, 9);
System.out.println(s2);
```

```
efghi
```

String 型のメソッド（4）

```
String s1 = "abcdefghijklmn";
```

toUpperCase()

文字列の各文字を大文字に変換し，変換した新たな文字列を返す.

```
String s2;
s2 = s1.toUpperCase();
System.out.println(s2);
```

```
ABCDEFGHIJKLMN
```

substring(m,n)

文字列の m+1 番目から n 番目の部分を返す（先頭は 0 番目）.

```
String s2;
s2 = s1.substring(4, 9);
System.out.println(s2);
```

```
efghi
```

String 型のメソッド（4）

```
String s1 = "abcdefghijklmn";
```

toUpperCase()

文字列の各文字を大文字に変換し，変換した新たな文字列を返す.

```
String s2;
s2 = s1.toUpperCase();
System.out.println(s2);

ABCDEFGHIJKLMN
```

substring(m,n)

文字列の m+1 番目から n 番目の部分を返す（先頭は 0 番目）.

```
String s2;
s2 = s1.substring(4, 9);
System.out.println(s2);

efghi
```

String 型のメソッド（4）

```
String s1 = "abcdefghijklmn";
```

toUpperCase()

文字列の各文字を大文字に変換し，変換した新たな文字列を返す.

```
String s2;
s2 = s1.toUpperCase();
System.out.println(s2);
```

```
ABCDEFGHIJKLMN
```

substring(m,n)

文字列の m+1 番目から n 番目の部分を返す（先頭は 0 番目）.

```
String s2;
s2 = s1.substring(4, 9);
System.out.println(s2);
```

```
efghi
```