初級プログラミング Ⅰ

第7回　文字と文字列

中谷　祐介

第7回の構成

◆ 第1講　文字

- Java における文字の扱いについて，基本事項を理解する.

◆ 第2講　文字列

- Java における文字列の扱いについて，基本事項を理解する.

◆ 第3講　文字列の処理

- 文字列の処理を行うための基本事項を理解する.

◆ 第4講　文字列の配列

- 文字列の配列を扱うための基本事項とその利用について理解する.

第7回　文字と文字列

第1講　文字

第1講の学習目標

- Java における文字の扱いについて，基本事項を理解する：
  - 文字
  - 文字コード
  - 数字・アルファベットの文字コード
  - 文字の入力

前回までの講義では…

- プログラムへの入出力を行うデータとして数値を扱った.
  - `int` 型：整数値
  - `double` 型：実数値
- プログラムの中で，データとして文字や文字列を扱いたい場合がある.
  - `char` 型：文字型. `char` 型の変数は，1文字格納できる.
  - `String` 型：文字列を扱うためのもの.

- 今回は，データとして文字や文字列を扱うプログラムを作成することを目的に，学習を進める.

文字（1）

[Sample_07_01.java]

- Java において**文字**を扱う場合は，変数には **char 型**を使用する．
- char 型の変数は，1文字を格納できる．

```
char c1 = 'A';
char c2 = 'ABC';     ← エラー
```

  - プログラム上で文字を表す場合は，「'」（**シングルクオテーション**）で囲む．

- 各文字には，コンピュータ内部では番号（整数値）が振られている．これを**文字コード**という．
- 文字コードの表示例

```
char c1 = 'a', c2 = 'あ';
System.out.printf("%c %d %x¥n", c1, (int)c1, (int)c1);
System.out.printf("%c %d %x¥n", c2, (int)c2, (int)c2);
```

```
a 97 61
あ 12354 3042
```

  - char 型の変数 c1, c2 に，それぞれ文字「a」，「あ」を代入．
  - 変数の内容を，文字（%c），10進数の文字コード（%d），16進数の文字コード（%x）で表示．

文字（2）

[Sample_07_01.java]

- char 型の変数には，文字コードを代入して文字を表すことができる：

```
char c3 = 'a';
char c4 = 97;
char c5 = 0x61;
System.out.println(c3 + " " + c4 + " " + c5);
```

```
a a a
```

  - 変数 c4 には，文字「a」の10進数での文字コード（97）を代入．
  - 変数 c5 には，文字「a」の16進数での文字コード（0x61）を代入．
  - 整数値に「0x」をつけることで，16進数であることを意味する．
  - 「'a'」，「97」，「0x61」は，いずれも文字「a」を表す．

- 文字「あ」に対しても，文字コードを代入することで文字を表せることを確認しよう：
  - 10進数での文字コード：12354，16進数での文字コード：0x3042．

文字コード（1）

◆ Java での文字コードには，Unicode と呼ばれるものが使用されている．

- 最初の128文字は，ASCIIコードと同じ．

Unicode の文字コード表（数字，アルファベット）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	NU							BL	BS	HT	LF	VT	FF	CR	SO	SI
001												EC				
002		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DL

数字

アルファベット大文字

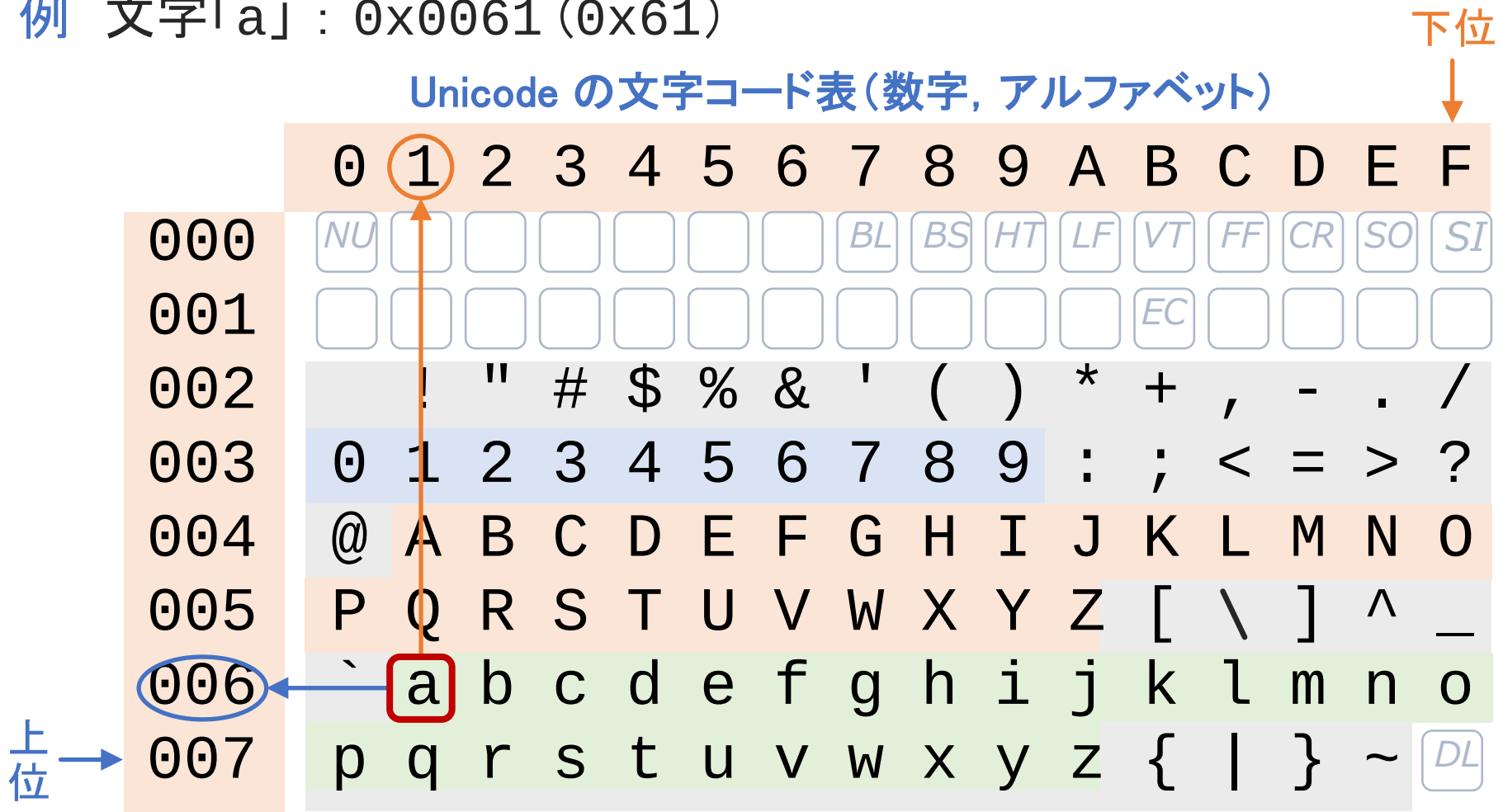
アルファベット小文字

ASCII（アスキー）：American Standard Code for Information Interchange

文字コード（2）

◆ 文字コード（16進数）は，文字コード表に従って割り当てられる．

- 例　文字「a」：0x0061（0x61）



文字コード（3）

◆ ひらがな等についても，文字コード（16進数）が割り当てられている．

- 例　文字「あ」：0x3042

Unicode の文字コード表（ひらがな）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
304		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
305	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
306	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
307	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
308	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
309	ゐ	ゑ	を	ん	ゔ	ゕ	ゖ									

数字・アルファベットの文字コード

[Sample_07_02.java]

- 数字・アルファベットの文字コードは，1ずつ異なる連続した値が割り当てられている：
  - 数字：'0' ～ '9' ⇔ 0x0030 ～ 0x0039
  - アルファベット（大文字）：'A' ～ 'Z' ⇔ 0x0041 ～ 0x005A
  - アルファベット（小文字）：'a' ～ 'z' ⇔ 0x0061 ～ 0x007A
- 文字コードが連続していることを利用して，次のようなことができる：

```
for (int i=0; i<26; i++) {
    System.out.print((char)('a'+i) + " ");
    if ((i+1)%10 == 0) System.out.println();
}
```

```
a b c d e f g h i j
k l m n o p q r s t
u v w x y z
```

  - 'a' は，文字「a」の文字コード．
  - 「'a'+i」は，文字「a」から i 番あとの文字の文字コード．

文字の入力

[Sample_07_03.java]

- Scanner クラスには，数値をプログラム内に読み込む道具がある：
  - nextInt メソッド：整数値（int 型）を読み込むメソッド
  - nextDouble メソッド：実数値（double 型）を読み込むメソッド
- Scanner クラスには文字列を読み込む next メソッドがあり，これを利用して文字を読み込む．
- 標準入力に入力した文字をプログラム内に読み込む例：

```
Scanner scan = new Scanner(System.in);

char c = scan.next().charAt(0);
System.out.println(c);
```

  - 「scan.next()」により，文字列を読み込む．
  - 「charAt(0)」により，文字列の先頭の文字を取り出す．
  - 実行時の入力例

標準入力

```
A
```

標準出力

```
A↵
```

例題7-1（1）

例題7-1

標準入力にアルファベット1文字（半角文字）を入力すると，それが小文字であるかどうかを判定し，小文字であればそのアルファベットを大文字に変換して標準出力に出力するプログラムを作成してください．

◆プログラムの考え方

- char 型の変数 c が小文字であるかどうか．
  - 小文字アルファベットの文字コード（'a' ～ 'z'）は，1ずつ異なる連続した値．
  - 変数 c の文字コードが 'a' 以上 'z' 以下であれば，c は小文字．
- 小文字のアルファベットを大文字に変換．
  - 入力した小文字は，文字「a」から何番あとかを計算 ⇒ x 番あと
  - 「'A'+x」は，文字「A」から x 番あとの文字の文字コード

例題7-1（2）

- char 型の変数 c が小文字であるかどうか.

```
if (('a'<=c) && (c<='z')) {
    System.out.println("小文字です.");
}
```

  - 小文字アルファベットの文字コード（'a' ～ 'z'）は，1ずつ異なる連続した値.
  - 変数 c の文字コードが 'a' 以上 'z' 以下であれば，c は小文字.

- 変数 c に格納された小文字のアルファベットを大文字に変換.

```
c = (char)('A' + (c-'a'));
```

  - **c-'a'** :c から文字「a」の文字コードを引くことで，c が文字「a」から何番あとかを計算.
  - 「'A' + **x**」は，文字「A」から x 番あとの文字の文字コード

「例題7-1」のプログラム例

Example_07_01.java

```
1  import java.util.Scanner;
2
3  public class Example_07_01 {
4      public static void main(String[] args) {
5          Scanner scan = new Scanner(System.in);
6
7          char c = scan.next().charAt(0);
8          if (('a'<=c) && (c<='z')) {
9              System.out.println("小文字です. ");
10             c = (char)('A' + (c-'a'));
11             System.out.println("大文字 : " + c);
12         }
13     }
14 }
```

◆ 実行例1

標準入力	標準出力
n	小文字です.↵ 大文字 : N↵

◆ 実行例2

標準入力	標準出力
Y	

第1講のまとめ

- Java における文字の扱いについて，基本事項を理解した：
  - 文字
  - 文字コード
  - 数字・アルファベットの文字コード
  - 文字の入力

第7回　文字と文字列

第1講　文字

終わり

第7回　文字と文字列

第2講　文字列

第2講の学習目標

- Java における文字列の扱いについて，基本事項を理解する：
  - 文字列
  - 文字列の入力・出力
  - 文字列の長さ
  - 基本型と参照型

文字列（1）

[Sample_07_04.java]

- 文字列を扱うために，文字（char 型）の配列を利用することができる：

```
char[] cc = {'東', '京', '通', '信', '大', '学'};

for (int i=0; i<cc.length; i++) {
    System.out.print(cc[i]);
}
```

```
東京通信大学
```

  - char 型の配列 cc の先頭要素から順に文字を格納．
  - for 文を使用して，1文字ずつ出力．

- 一般に，Java の場合，文字列を扱うためには String 型というものを利用する．

文字列（2）

- Java において**文字列**を扱う場合は，**`String` 型**を使用する．
- `String` 型は，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．
  - これまで，データを入力するツールとして使用した Scanner クラスもクラスの一つ．
  - 文字列を扱うための道具（メソッド）が準備されている．
- `String` 型の変数は，文字列を扱うことができる．

```
String s1 = "ABC";
String s2 = "A";
String s3 = 'A';     ← エラー
```

  - プログラム上で文字列を表す場合は，「`"`」（ダブルクオテーション）で囲む．
  - `String` 型では，文字（「`'`」で囲んだもの）は扱えない．
  - 文字を扱う場合は，1文字の文字列とする．

文字列の使用例

[Sample_07_05.java]

String 型変数の宣言と初期化

```
String str1;
String str2 = "efgh";
```

- 基本型(int 型，double 型)等と同様に，変数の宣言，初期化を行える．
- 文字列は「"」で囲む．

変数への文字列の対応付けと文字列の連結

```
str1 = "abcd";
System.out.println(str1);

str1 = str1 + str2;
System.out.println(str1);
```

```
abcd
abcdefgh
```

- 「=」を使用することで，文字列を変数に対応させる(代入に相当)．
- 「+」演算子を使用することで，2つの文字列を連結する．

文字列の入力・出力（1）

[Sample_07_06.java]

- 本科目では，データの入力に Scanner クラスを使用する．
  - 数値の入力には，nextInt メソッド（int 型），nextDouble メソッド（double 型）を使用した．
  - 文字列の入力には，**next メソッド**，**nextLine メソッド**を使用する．
- next メソッドにより，標準入力に入力した文字列をプログラム内に読み込む例：

```
Scanner scan = new Scanner(System.in);

String str = scan.next();
System.out.println(str);    ← print，println メソッドにより，文字列の内容を出力
```

  - next メソッドは，空白文字等を区切りとして，区切りまでの文字列を読み込む．
  - 実行時の入力例1

標準入力 | Tokyo

標準出力 | Tokyo↵

  - 実行時の入力例2

標準入力 | Tokyo Online University

標準出力 | Tokyo↵

文字列の入力・出力（2）

[Sample_07_07.java]

◆ 標準入力に入力した内容を，1行単位でプログラム内に読み込む例：

```
Scanner scan = new Scanner(System.in);

String str = scan.nextLine();
System.out.printf("%s¥n", str);
```

- nextLine メソッドは，行の終わりまでを文字列として読み込む．
- printf メソッドによる出力の際，文字列を出力する場合は書式指定子に「%s」を指定する．
- 実行時の入力例

標準入力

```
Tokyo Online University
```

標準出力

```
Tokyo Online University↵
```

文字列の長さ

[Sample_07_08.java]

◆ 文字列の長さ(文字数)は,「変数名.length()」で取得することができる:

```
String  str = "Tokyo Online University";
System.out.println("文字列strの長さ:" + str.length());
```

```
文字列strの長さ:23
```

- String 型には, 文字列を扱うための道具(メソッド)が準備されている.
- length メソッドは, 文字列の長さを取得するための道具.
- String 型のメソッドについては, 第3講で学習する.

◆ [復習] 配列の長さ(要素数)は,「配列名.length」で取得することができる:

```
int[] a = {10, 20, 30, 40, 50};
System.out.println("配列aの長さ:" + a.length);
```

```
配列aの長さ:5
```

- 配列 a は要素数「5」の配列.
- a.length で, 配列 a の要素数「5」を取得できる.

基本型と参照型（1）

- String 型は，クラスとして宣言されている．
  - 正確には **String クラス**と呼ぶ．
  - クラスは，**参照型**と呼ばれる型に属する．
  - 参照型は，変数の**基本型**（int 型，double 型，char 型，等）とは性質が異なる．

- 基本型（int 型，double 型，char 型，等）

```
int a = 123;
double x = 12.34;
```

  - 基本型の変数は，値が変数の中に直接格納されるイメージ．

- 参照型（String 型）

```
String str = "abcd";
```

  - 「abcd」という文字列の実体が作成される．
  - String 型の変数 str が，文字列「abcd」の実体を参照する．

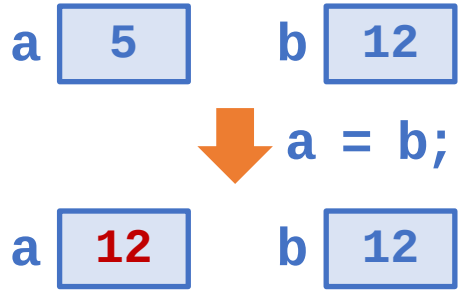
基本型と参照型（2）

[Sample_07_09.java]

基本型の代入

```
int a = 5, b = 12;
System.out.println("a:" + a + ", b:" + b);
a = b;
System.out.println("a:" + a + ", b:" + b);
```

```
a:5, b:12
a:12, b:12
```

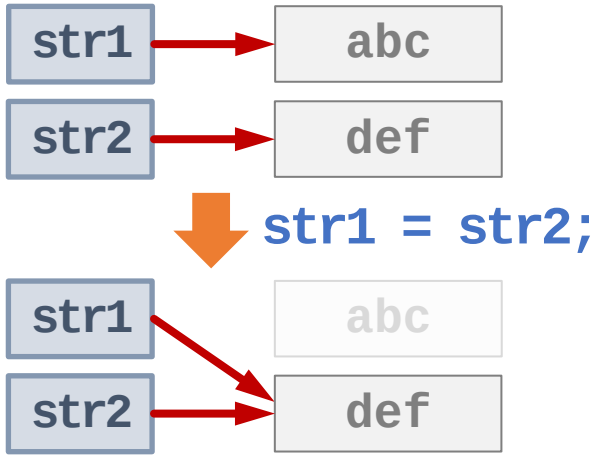


- 「a = b;」により，変数 b の値が変数 a に代入される．

参照型の代入

```
String  str1 = "abc", str2 = "def";
System.out.println("str1:" + str1 + ", str2:" + str2);
str1 = str2;
System.out.println("str1:" + str1 + ", str2:" + str2);
```

```
str1:abc, str2:def
str1:def, str2:def
```



- 「str1 = str2;」により，str2 の参照先を str1 の参照先とする．
- 2つの変数 str1, str2 が同じ実体を参照する．

例題7-2（1）

例題7-2

標準入力に2つの文字列（空白を含まない）を入力すると，長い方の文字列を標準出力に出力するプログラムを作成してください．長さが同じ場合は，両方の文字列を出力してください．

◆プログラムの考え方

- 標準入力への2つの文字列の入力．
  - Scanner クラスの next メソッドを利用．
- 長い方の文字列の出力
  - length メソッドの利用．
  - 条件分岐による処理．

「例題7-2」のプログラム例

Example_07_02.java

```
1  import java.util.Scanner;
2
3  public class Example_07_02 {
4    public static void main(String[] args) {
5      Scanner scan = new Scanner(System.in);
6
7      String str1 = scan.next();
8      String str2 = scan.next();
9      if (str1.length() > str2.length()) {
10       System.out.println(str1);
11     } else if (str1.length() < str2.length()) {
12       System.out.println(str2);
13     } else {
14       System.out.println(str1 + "¥n" + str2);
15     }
16   }
17 }
```

◆ 実行例1

標準入力	標準出力
Tokyo Online	Online↵

◆ 実行例2

標準入力	標準出力
あいうえお abcde	あいうえお↵ abcde↵

第2講のまとめ

- Java における文字列の扱いについて，基本事項を理解した：
  - 文字列
  - 文字列の入力・出力
  - 文字列の長さ
  - 基本型と参照型

第7回　文字と文字列

第2講　文字列

終わり

第7回　文字と文字列

第3講　文字列の処理

第3講の学習目標

- 文字列の処理を行うための基本事項を理解する:
  - 文字列
  - `String` 型（クラス）のメソッド
    - `length` メソッド
    - `charAt` メソッド
    - `equals` メソッド
    - `indexOf` メソッド
    - `toUpperCase` メソッド
    - `substring` メソッド
    - `toCharArray` メソッド

文字列

- Java において文字列を扱う場合は，String 型を使用する．
- String 型は，クラスとして定義されている．
- クラスには，そのクラスを便利に利用するための**メソッド**と呼ばれるものが準備されている．
  - データを入力するツールとして使用する Scanner クラスもクラスの一つ．
  - Scanner クラスには，nextInt メソッド，nextDouble メソッド，等が準備されている．
- String 型（クラス）にも，文字列の処理を行うための多くのメソッドが準備されている．

- String 型（クラス）のメソッドを理解し，文字列の処理を行う．

String 型（クラス）のメソッド（1）

◆ String 型（クラス）のメソッドには，下表に示すものがある．

String 型（クラス）のメソッドの例

メソッド名	説　明
int **length()**	文字列の長さ（文字数）を返す．
char **charAt(***int* **n)**	文字列の先頭を 0 番目として，n 番目の文字を返す．
int **indexOf(***String* **str)**	文字列内で，文字列 str が最初に現れる位置を返す．
boolean **equals(***String* **str)**	文字列が，str と等しいかどうかを返す．
boolean **startWith(***String* **str)**	文字列が，str で始まっているかどうかを返す．
boolean **endWith(***String* **str)**	文字列が，str で終わっているかどうかを返す．
String **toLowerCase()**	文字列内の各文字を，小文字に変換する．
String **toUpperCase()**	文字列内の各文字を，大文字に変換する．
String **substring(***int* **m,** *int* **n)**	文字列の m番目からn番目までの部分文字列を返す．
String **replace(***char* **old,** *char* **new)**	文字列内の文字 old を new に変換する．
char[] **toCharArray()**	文字列を文字（char 型）の配列に変換する．

String 型（クラス）のメソッド（2）

◆ 前頁の表に示したメソッドの使い方 ： **例** charAt メソッド

char **charAt(***int* **n)** ： 文字列の先頭を 0 番目として，n 番目の文字を返す．

( ) 内には int 型のデータを指定する．

charAt メソッドの結果は，char 型のデータになる．これを「char 型のデータを返す」と表現する．

◆ charAt メソッドの使用例

```
String str = "abcdefg";
char c = str.charAt(3);
```

- 文字列 str に対して charAt メソッドを使用．
- charAt メソッドの ( ) 内に int 型の値「3」を指定．
  ⇒ 文字列 str の3番目の文字を取り出す．
- charAt メソッドの結果は char 型のデータ（文字「d」）となる．
- この結果を char 型の変数 c に代入．

length メソッド

[Sample_07_10.java]

int length()

- 文字列の長さ(文字数)を int 型の値で返す.
- ( ) 内には, 何も記述しない.

length メソッドの使用例

```
String str1 = "Tokyo Online University 情報マネジメント学部";

int n = str1.length();
System.out.println("n : " + n);
```

```
n : 34
```

- 文字列 str1 の長さを求め, int 型の変数 n に代入.

charAt メソッド

[Sample_07_10.java]

- *char* **charAt(*int* n)**
  - 文字列の n 番目の文字(char 型)を返す. その際, 先頭の文字を 0 番とする.
  - ( ) 内には, 何番目かを指定する値 n を整数値(int 型)で与える.

- charAt メソッドの使用例

```
String str1 = "Tokyo Online University 情報マネジメント学部";
                0番       10番              24番          33番
int n = str1.length();
char c1 = str1.charAt(10);
char c2 = str1.charAt(n-10);
System.out.println("c1 : " + c1 + ", c2 : " + c2);
```

```
c1 : n, c2 : 情
```

  - 文字列 str1 の長さ n は「34」. str1 の文字は 0 番～33 番.
  - 文字列 str1 の最後の文字は「n-1」番.「n-10」は, 文字列の末尾から10番目.

equals メソッド

[Sample_07_10.java]

boolean **equals(*String* str)**

- 文字列が他の文字列(str)と等しいかどうかを，大文字，小文字の違いを含めて判定し，その結果を真偽値(true または false)で返す．
- ( ) 内には，比較の対象となる他の文字列を指定する．

equals メソッドの使用例

```
String str1 = "Tokyo Online University 情報マネジメント学部";
String str2 = "Tokyo online university 情報マネジメント学部";

boolean b1 = str1.equals(str2);
boolean b2 = str1.equalsIgnoreCase(str2);
System.out.println("b1 : " + b1 + ", b2 : " + b2);
```

```
b1 : false, b2 : true
```

- 文字列 str1 と文字列 str2 は，一部，大文字と小文字の違いがある．
- equalsIgnoreCase メソッドは，大文字と小文字を区別せずに，2つの文字列が等しいかどうかを判定する．

indexOf メソッド

[Sample_07_11.java]

int indexOf(*String* str)

- indexOf メソッドは，文字列内において指定する文字列 str の探索を行うメソッド．
- 文字列内で，対象の文字列 str が最初に現れる位置を返す．その際，先頭の文字を 0 番とする．対象の文字列 str が現れない場合は，「-1」を返す．
- ( ) 内には，探索を行う文字列 str を指定する．

indexOf メソッドの使用例

```
String str1 = "Tokyo Online University Java programming";
                  0番       9番                    37番
String str2 = "university";
int p = str1.indexOf("in");     // 探索を行う文字列「in」を直接指定．
int q = str1.indexOf(str2);     // 探索を行う文字列を変数str2で指定．
System.out.println("p : " + p + ", q : " + q);
```

```
p : 9, q : -1
```

toUpperCase メソッド

[Sample_07_11.java]

String toUpperCase()

- 文字列に含まれるアルファベットの各文字を大文字に変換した新たな文字列を生成し，その文字列を返す．
- ( ) 内には，何も記述しない．

toUpperCase メソッドの使用例

```
String str1 = "Tokyo Online University Java programming";

String str3 = str1.toUpperCase();
System.out.println("str1 : " + str1);
System.out.println("str3 : " + str3);
```

```
str1 : Tokyo Online University Java programming
str3 : TOKYO ONLINE UNIVERSITY JAVA PROGRAMMING
```

- 大文字に変換した新たな文字列を生成する．文字列 str1 の内容を書き換えるわけではない．

substring メソッド

[Sample_07_11.java]

String **substring(*int* m, *int* n)**

- 文字列の m 番目～ n-1 番目の部分を新たな文字列として生成し，その部分文字列を返す．その際，先頭の文字を 0 番とする．
- ( ) 内には，部分文字列の先頭(m)と末尾(n)を表す値を int 型で与える．

substring メソッドの使用例

```
String str1 = "Tokyo Online University Java programming";
                0番    6番         15番
String str4 = str1.substring(6, 16);
System.out.println("str4 : " + str4);
```

```
str4 : Online Uni
```

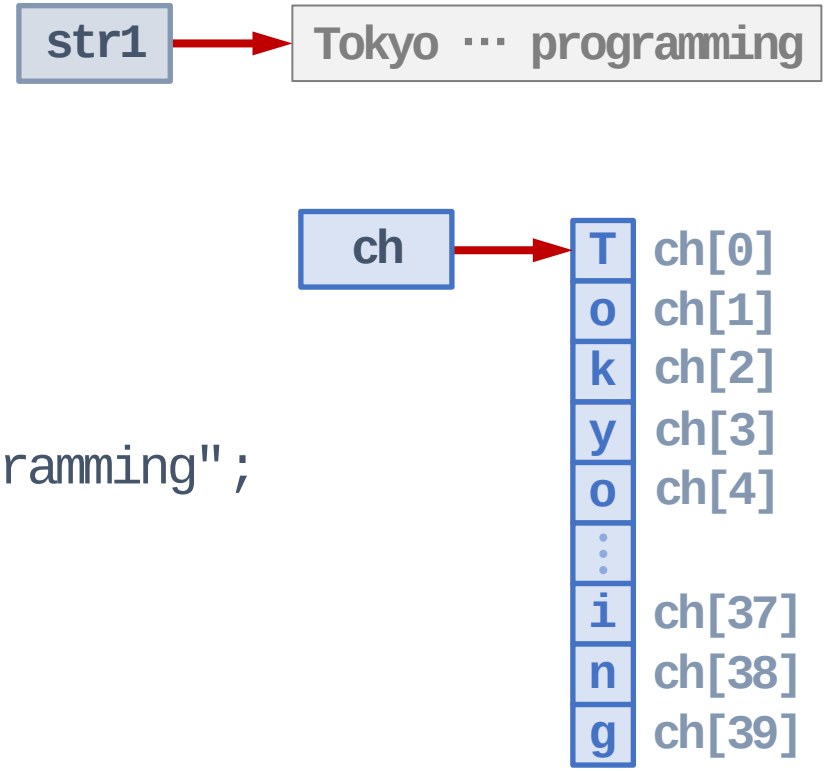
- 文字列 str1 の6番目～15番目の部分を新たな文字列として生成し，str4 とする．
- 文字列 str1 の内容は，変わらない．

toCharArray メソッド

[Sample_07_11.java]

char[] toCharArray()

- 文字列を，文字（char 型）の配列に変換した新たな配列を生成し，その文字配列を返す．
- ( ) 内には，何も記述しない．



toCharArray メソッドの使用例

```
String str1 = "Tokyo Online University Java programming";

char[] ch = str1.toCharArray();
System.out.println("str1 : " + str1);
System.out.print("  ch : ");
for (int i=0; i<ch.length; i++) System.out.print(ch[i]);
```

```
str1 : Tokyo Online University Java programming
  ch : Tokyo Online University Java programming
```

- 配列 ch の長さは，ch.length で取得できる．

例題7-3（1）

例題7-3

標準入力に文字列（空白を含む）を入力すると，その文字列の末尾から先頭に向かって逆順に，標準出力に出力するプログラムを作成してください．

◆プログラムの考え方

- 文字列 str の長さ（文字数）は，str.length() で取得できる．

```
int n = str.length();
```

- 文字列 str の先頭から i 番目の文字は，str.charAt(i) で取得できる．
  - 先頭の文字は「str.charAt(0)」，末尾の文字は「str.charAt(n-1)」．
- 文字列の末尾から先頭に向かって，1文字ずつ出力すればよい．

「例題7-3」のプログラム例

Example_07_03.java

```
1  import java.util.Scanner;
2
3  public class Example_07_03 {
4      public static void main(String[] args) {
5          Scanner scan = new Scanner(System.in);
6
7          String str = scan.nextLine();
8
9          int n = str.length();
10         for (int i=n-1; i>=0; i--) {
11             char c = str.charAt(i);
12             System.out.print(c);
13         }
14     }
15 }
```

◆ 実行例1

標準入力	標準出力
Online Univ.	.vinU enilnO

◆ 実行例2

標準入力	標準出力
東京通信大学	学大信通京東

「例題7-3」のプログラム例（別解）

Example_07_03_2.java

```
1  import java.util.Scanner;
2
3  public class Example_07_03_2 {
4      public static void main(String[] args) {
5          Scanner scan = new Scanner(System.in);
6
7          String str = scan.nextLine();
8
9          char[] ch = str.toCharArray();
10         for (int i=ch.length-1; i>=0; i--) {
11             System.out.print(ch[i]);
12         }
13     }
14 }
```

◆ 実行例1

標準入力	標準出力
Online Univ.	.vinU enilnO

◆ 実行例2

標準入力	標準出力
東京通信大学	学大信通京東

第3講のまとめ

◆ 文字列の処理を行うための基本事項を理解した：

- 文字列
- String 型（クラス）のメソッド
  - length メソッド
  - charAt メソッド
  - equals メソッド
  - indexOf メソッド
  - toUpperCase メソッド
  - substring メソッド
  - toCharArray メソッド

第7回　文字と文字列

第3講　文字列の処理

終わり

第7回　文字と文字列

第4講　文字列の配列

第4講の学習目標

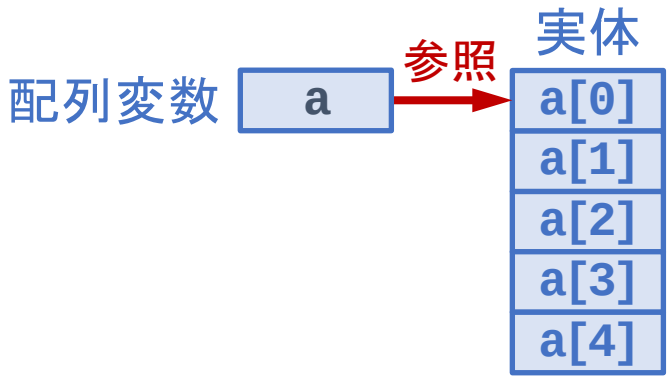
- 文字列の配列を扱うための基本事項とその利用について理解する:
  - 配列と文字列
  - 文字列の配列
  - コマンドライン引数
  - 文字列から数値への変換

配列と文字列

- 配列は，同じ型の変数を複数個まとめて扱うことができるデータの構造である．
  - **例** 要素数「5」の int 型の配列 a の宣言

```
int[] a = new int[5];
```

    - 配列の要素 a[i] は，int 型の変数．
  - String 型の配列を作成することもできる．



- 文字列を扱う場合は，String 型を使用する．
  - **例** 文字列「abcd」を扱う String 型の変数 s の宣言

```
String s = "abcd";
```

- String 型の配列を作成することで，文字列の配列を扱える．

文字列の配列

- 文字列の配列は，String 型の配列として実現できる．
- 文字列の配列を宣言する例：要素数「3」の String 型の配列 sa

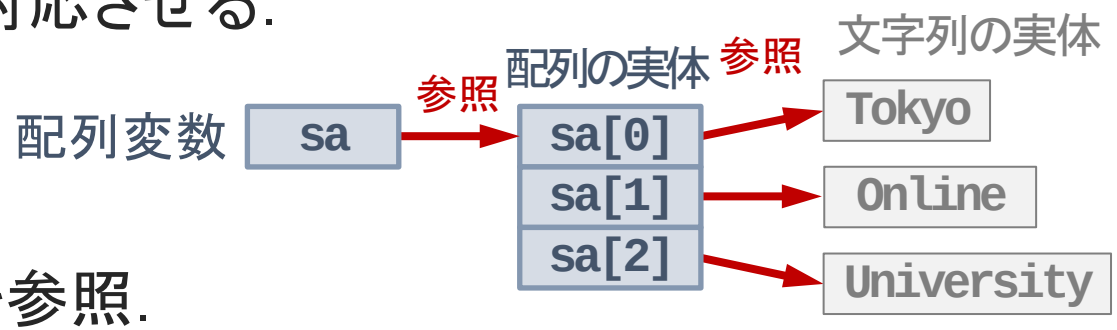
```
String[] sa = new String[3];
```

  - 配列変数 sa が，String 型の配列の実体を参照．
  - 配列の要素 sa[i] は，String 型の変数．

- 配列の要素 sa[i] に具体的な文字列を対応させる．

```
sa[0] = "Tokyo";
sa[1] = "Online";
sa[2] = "University";
```

  - 配列の要素 sa[i] が文字列の実体を参照．



- 文字列の配列は，次のように初期化することができる：

```
String[] sa = {"Tokyo", "Online", "University"};
```

文字列の配列の利用

[Sample_07_12.java]

◆ 文字列の配列の各要素に保存した文字列の文字数を出力する例

```
String[] sa = new String[3];
sa[0] = "Tokyo";
sa[1] = "Online";
sa[2] = "University";

for (int i=0; i<sa.length; i++) {
    int n = sa[i].length();
    System.out.println(sa[i] + " : " + n + "文字");
}
```

- 要素数「3」の String 型の配列 sa を宣言．各要素 sa[i] は String 型の変数．
- 配列 sa の各要素 sa[i] に具体的な文字列を対応させる．
- 各要素の文字列の文字数を出力．
  - 配列 sa の長さは，「sa.length」で取得．
  - 文字列 sa[i] の長さ（文字数）は，「sa[i].length()」で取得．

コマンドライン引数（1）

- 文字列の配列の宣言は，「String[] 配列名」とする．
- これまでプログラムを作成する際に記述していた main メソッドにもこの記述がある：

```
public class クラス名 {
    public static void main(String[] args) {

        /* プログラムの本体 */

    }
}
```

  - 文字列の配列 args を利用している．
  - main メソッドの ( ) 内を**コマンドライン引数**と呼ぶ．

- コマンドライン引数の役割は，プログラムの起動時に args 内のデータをプログラム内に読み込むというもの．
  - 演習環境「@CODE ROOM」では，「コマンドライン入力」欄に入力したデータが args 内に格納される．

コマンドライン引数（2）

[Sample_07_13.java]

◆ コマンドライン引数によるプログラムへのデータの読み込み

```
public static void main(String[] args) {
    int n = args.length;
    System.out.println("n : " + n);

    for (int i=0; i<n; i++) {
        String str = args[i];
        System.out.println(i + " : " + str);
    }
}
```

- 配列 args の長さ（args.length）は，コマンドライン引数に与えたデータの数．
- 配列 args の各要素（args[i]）は，コマンドライン引数に与えたデータを参照．
- 実行例

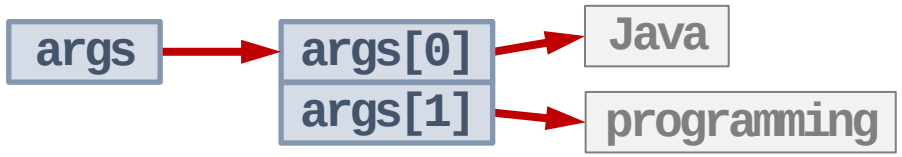
コマンドライン入力

```
Java programming
```

args[0]　args[1]

標準出力

```
n : 2
0 : Java
1 : programming
```



標準入力とコマンドライン入力

- 標準入力
  - プログラムの起動後に，プログラム内で必要な型（`int` 型，`double` 型，等）のデータを読み込む．
  - 本科目では，Scanner クラスを使用して，標準入力に入力したデータを読み込む．
- コマンドライン入力
  - プログラムの起動時に，プログラム内で必要なデータを文字列（`String` 型）として読み込む．
- 演習環境「@CODE ROOM」では，標準入力，コマンドライン入力ともに，プログラムの実行時に，プログラム内で必要なデータを読み込む，という点では同じ．

文字列から数値への変換（1）

- コマンドライン引数（String[] args）は，文字列の配列である．
  - コマンドライン引数を通してプログラム内に読み込まれるデータは文字列．
  - コマンドライン引数を通して数値等を読み込みたい場合，変換が必要．
- 読み込みたいデータの型に応じて，メソッドが準備されている．
  - **例** int 型：Integer クラスの parseInt メソッド．

文字列から数値等へ変換するメソッド

メソッド名	説　明
Boolean.parseBoolean(*String* str)	真偽値（true または false）へ変換
Short.parseShort(*String* str)	整数（short 型）への変換
Integer.parseInt(*String* str)	整数（int 型）への変換
Long.parseLong(*String* str)	整数（long 型）への変換
Float.parseFloat(*String* str)	単精度浮動小数点数（float 型）への変換
Double.parseDouble(*String* str)	倍精度浮動小数点数（double 型）への変換

文字列から数値への変換（2）

[Sample_07_14.java]

◆ 文字列から数値へ変換を行う例（円の面積の計算）

```
String str1 = "4";
String str2 = "3.14";

int r = Integer.parseInt(str1);
double pi = Double.parseDouble(str2);

double a = pi*r*r;
System.out.println("a : " + a);
```

```
a : 50.24
```

- 文字列「4」，「3.14」を，それぞれ文字列 str1，str2 として宣言．
- Integer クラスの parseInt メソッドを使用して，文字列「4」を int 型に変換．
- Double クラスの parseDouble メソッドを使用して，文字列「3.14」を double 型に変換．

例題7-4

例題7-4

コマンドライン入力に1つ以上の正の整数値を入力すると，入力したすべての整数値の和を計算し，標準出力に出力するプログラムを作成してください．

コマンドライン入力	標準出力
1 2 3 4 5	15↵

◆プログラムの考え方

- コマンドライン入力に入力したデータは，コマンドライン引数（String[] args）を通して文字列としてプログラム内に読み込まれる．
  - 入力したデータ数は，args.length により取得できる．
  - 入力したデータは，先頭から順に args[0], args[1], args[2], … に対応する．
- 読み込んだ文字列（args[0], args[1], args[2], … ）を整数値（int 型）に変換して和を計算する．

「例題7-4」のプログラム例

Example_07_04.java

```
1  public class Example_07_04 {
2      public static void main(String[] args) {
3          int n = args.length;
4
5          int total = 0;
6          for (int i=0; i<n; i++) {
7              total += Integer.parseInt(args[i]);
8          }
9
10         System.out.println(total);
11     }
12 }
```

◆ 実行例

コマンドライン入力	標準出力
1 2 3 4 5	15↵

第4講のまとめ学習目標

◆ 文字列の配列を扱うための基本事項とその利用について理解した：

- 配列と文字列
- 文字列の配列
- コマンドライン引数
- 文字列から数値への変換

第7回　文字と文字列

第4講　文字列の配列

終わり