Theory: Characters

8 minutes 5 / 12 problems solved

Start practicing

The `char` type is used to represent letters (both uppercase and lowercase), digits, and other symbols. Each character is just a symbol enclosed in single quotes.

```
char lowerCaseLetter = 'a';
char upperCaseLetter = 'Q';
char number = '1';
char space = ' ';
char dollar = '$';
```

This type can represent all characters in all languages as well as some special and computer symbols. It corresponds to the **Unicode** (UTF-16) format. Unicode is a computer encoding methodology that assigns a unique number for every character. It doesn't matter what language, or computer platform it's on. This is important in a global, networked world, and for computer systems that must accommodate multiple languages and special characters. Unicode truly unifies all of these into a single standard.

§1. Initializing characters with codes

A character can be also created using its hexadecimal code in the Unicode table. The code starts with `\u`.

```
char ch = '\u0040'; // it represents '@'
System.out.println(ch); // @
```

Although we use a sequence of characters to represent such code, the code represents exactly one character.

As an example, Latin capital letters have hexadecimal codes from `'\u0041'` to `'\u005A'`, and Latin small letters have codes from `'\u0061'` to `'\u007A'`.

The `char` type has a minimum value encoded as `'\u0000'` and the maximum value encoded as `'\uffff'`.

It is also possible to initialize a char with a positive integer number.

```
char ch = 64;
System.out.println(ch); // @
```

The number `64` just corresponds to the Unicode hexadecimal code `'\u0040'`.

Any `char` variable may be considered as an unsigned integer value in the range from 0 to 65535.

§2. Retrieving subsequent characters

There are two operators for adding (`+`) and subtracting (`-`) integer numbers in order to get the next and previous character according to the Unicode order.

```
char ch = 'b';
ch += 1; // 'c'
ch -= 2; // 'a'
```

It is also possible adding and subtracting one character to / from another one.

```
char ch = 'b';
ch += 'a';
ch -= 'b';
System.out.println(ch); // prints 'a' without quotes
```

Current topic:

✓ Characters — Stage 1

Topic depends on:

✓ Increment and decrement — Stage 1

Topic is required for:

String — Stage 2

Table of contents:

Actually, these operations manipulate with codes of characters, `'b'` has the next code after `'a'`.

It is possible to use increment (`++`) and decrement (`--`) operators in prefix and postfix forms.

```
char ch = 'A';
ch += 10;
System.out.println(ch);   // 'K'
System.out.println(++ch); // 'L'
System.out.println(++ch); // 'M'
System.out.println(--ch); // 'L'
```

§3. Escape sequences

There are some special characters starting with backslash `\` which are known as the escape or control sequences. They do not have corresponding symbols and cannot be found on a keyboard. To represent such characters we use a pair of regular symbols. In a program, this pair will be considered as exactly one character with the appropriate code.

- `'\n'` is the newline character;
- `'\t'` is the tab character;
- `'\r'` is the carriage return character;
- `'\\'` is the backslash character itself;
- `'\''` is the single quote mark;
- `'\"'` is the double quote mark.

Here are several examples:

```
System.out.print('\t'); // makes a tab
System.out.print('a');  // prints 'a'
System.out.print('\n'); // goes to the new line
System.out.print('c');  // prints 'c'
```

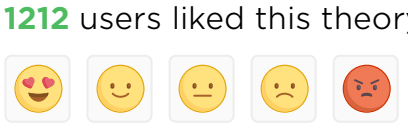
This code prints:

```
  a
c
```

There is also a character to represent a single space `' '`. It is just a regular character, not an escape sequence.

Report a typo

1212 users liked this theory. **14** didn't like it. **What about you?**



Start practicing

Comments (18) Hints (0) Useful links (0) Show discussion