Java - Character Class

Normally, when we work with characters, we use primitive data types char.

```
char ch = 'a';

// Unicode for uppercase Greek omega character
char uniChar = '\u039A';

// an array of chars
char[] charArray ={ 'a', 'b', 'c', 'd', 'e' };
```

However in development, we come across situations where we need to use objects instead of primitive data types. In order to achieve this, Java provides wrapper class **Character** for primitive data type char.

The Character class offers a number of useful class (i.e., static) methods for manipulating characters. You can create a Character object with the Character constructor −

Character ch = new Character('a');

The Java compiler will also create a Character object for you under some circumstances. For example, if you pass a primitive char into a method that expects an object, the compiler automatically converts the char to a Character for you. This feature is called autoboxing or unboxing, if the conversion goes the other way.

```
// Here following primitive char 'a'
// is boxed into the Character object ch
Character ch = 'a';

// Here primitive 'x' is boxed for method test,
// return is unboxed to char 'c'
char c = test('x');
```

Escape Sequences

A character preceded by a backslash (\) is an escape sequence and has a special meaning to the compiler.

The newline character (\n) has been used frequently in this tutorial in System.out.println() statements to advance to the next line after the string is printed.

Following table shows the Java escape sequences −

Escape Sequence	Description
\t	Inserts a tab in the text at this point.
\b	Inserts a backspace in the text at this point.
\n	Inserts a newline in the text at this point.
\r	Inserts a carriage return in the text at this point.
\f	Inserts a form feed in the text at this point.
\'	Inserts a single quote character in the text at this point.
\"	Inserts a double quote character in the text at this point.
\\	Inserts a backslash character in the text at this point.

When an escape sequence is encountered in a print statement, the compiler interprets it accordingly.

Example

If you want to put quotes within quotes, you must use the escape sequence, \", on the interior quotes −

Live Demo

```java
public class Test {

   public static void main(String args[]) {
      System.out.println("She said \"Hello!\" to me.");
   }
```

```
}
```

This will produce the following result −

Output

She said "Hello!" to me.

Character Methods

Following is the list of the important instance methods that all the subclasses of the Character class implement −

Sr.No.	Method & Description
1	isLetter() Determines whether the specified char value is a letter.
2	isDigit() Determines whether the specified char value is a digit.
3	isWhitespace() Determines whether the specified char value is white space.
4	isUpperCase() Determines whether the specified char value is uppercase.
5	isLowerCase() Determines whether the specified char value is lowercase.
6	toUpperCase() Returns the uppercase form of the specified char value.
7	toLowerCase() Returns the lowercase form of the

		specified char value.
8	toString()	
		Returns a String object representing the specified character value that is, a one-character string.

For a complete list of methods, please refer to the java.lang.Character API specification.

What is Next?

In the next section, we will be going through the String class in Java. You will be learning how to declare and use Strings efficiently as well as some of the important methods in the String class.

References

- www.tutorialspoint.com/java