AP Computer Science Backgrounder™

String Parsing and Manipulation

The following are the most-commonly-useful methods available for working with strings. Methods marked as *static* can be called with the class name alone; you don't need to create a string object.

String Parsing

*int **strobj.indexOf**(String target)*
*int **strobj.indexOf**(String target, int startHere)*

```
int commaLoc = myString.indexOf(",");
int commaLoc = myString.indexOf(",", 10);
```

- Returns the location (that is, the character location within the string) of the first instance of the target string.
- Starts at *startHere*, if supplied.
- The first character in the string is location 0.
- Returns –1 if *target* is not found.

*String **strobj.substring**(int startHere)*
*String **strobj.substring**(int startHere, int endHere)*

```
String familyName = myString.substring(12);
String firstName = myString.substring(0,10);
```

- Returns a substring starting at location *startHere* in the string; the first character position is numbered zero.
- If *endHere* is provided, the substring will extend from startHere up to, *but not including*, the *endHere* location.
- If *endHere* is not provided, the substring will extend from startHere to the end of the string.

Modifying Strings

*String **strobj.concat**(String addition)*

```
String newString = myString.concat(", Tuba Hunter”);
```

- Adds *addition* to the current string and returns the result as a new String.

*String **strobj.toLowerCase**()*
*String **strobj.toUpperCase**()*

```
String newString = myString.toLowerCase(“BOOM Chakka-lakka-lakka”);
```

- Converts the characters in the current string to lower or upper case and returns the result as a new String.

String ***strobj.trim***()

```
String newString = myString.trim("   spacey!   ");
```

- Removes any starting or trailing whitespace characters from the current string and returns the result as a new String.

Comparing Strings

int ***strobj.compareTo***(*String target*)
int ***strobj.compareToIgnoreCase***(*String target*)

```
int result = myString.compareTo("Halt!");
```

- Compare the current string lexicographically (that is, alphabetically) with *target* and returns an integer:
  - –1 if *target* < the current string
  - 0 if *target* == the current string
  - 1 if *target* > the current string
- *compareToIgnoreCase* ignores the case of the current and target strings.

Conversion to Other Data Types

The conversion of a string to another data type, such as int, is carried out by class methods of the various data type classes; thus:

Integer ***integerobj.parseInt***(*String s*)

```
Int intObj = Integer.parseInt("246");
```

- Converts *s* to an integer and returns an Integer object.
  - Note that to convert this to an int you need to use the `intValue()` method of the Integer:

    ```
    int i = Integer.parseInt("246").intValue();
    ```

Miscellaneous

Integer ***integerobj.length***()

```
int len = strObj.length();
```

- Returns the number of characters in the current String.