**Administrative**

**Today’s session**

Strings

Wrapper classes

Characters

String tokenizing

**Session Topics**

**Strings**

● A **string** is zero or characters enclosed in double quotes.

● A string is an object that is **immutable** – it doesn’t change once it is created.

● String methods are actions we can perform on a string variable (or string constant).

● Syntax:

**<string-variable>.<string-method>(<parameter-list>);**

Where …

**<string-variable>** is declared prior to the method call.

**<string-method>** is a valid String method.

**<parameter-list>** is a list of zero or more inputs needed by the method.

● Here are some String methods used with a string object. See the complete list at [docs.oracle.com/javase/8/docs/api/java/lang/String.html](http://docs.oracle.com/javase/8/docs/api/java/lang/String.html).

| Method | Purpose |
| --- | --- |
| length() | Return the number of characters in the string. |
| equals(<test-string>) | Return *true* if <test-string> equals string, otherwise return *false*. |
| equalsIgnoreCase(<test-string>) | Return *true* if <test-string> equals string, ignoring case. Otherwise, return *false*. |
| compareTo(<test-string>) | ● If <test-string> is alphabetically before string, return the positive difference between the first two letters.  ● If <test-string> is identical to the string, return 0.  ● If <test-string> is alphabetically after the string, return the negative difference between the first two letters. |
| substring(<start-index>, <end-index>) | Return part of the string, from <start-index> to <end-index> – 1. |
| trim() | Return the string with leading and trailing spaces removed. |
| toUpperCase() | Return the string with all characters in upper case. |
| toLowerCase() | Return the string with all characters in lower case. |
| charAt(<index>) | Return the character from the string at <index>. |
| concat(<string-expr>) | Add <string-expr> to the end of the string. |
| indexOf(<test-string>) | ● If <test-string> occurs within the string, return the index of the first occurrence. Otherwise, return -1. |

● Here are some **static** String methods. See the complete list at [docs.oracle.com/javase/8/docs/api/java/lang/String.html](http://docs.oracle.com/javase/8/docs/api/java/lang/String.html).

| Method | Purpose |
| --- | --- |
| String.format(<format-spec>,<expr-list>) | Return a string formed from <format-spec> and <expr-list>. These are the same parameters as the **printf** method. The returned string may be:  ● Stored in a string variable.  ● Used in a larger string expression.  ● Used in a graphical user interface to format strings appearing in messages. |
| String.valueOf(<expr>) | Return <expr> converted to a string. |

● See **String methods** sample application on Blackboard.

**Wrapper classes**

● A **wrapper class** enables a primitive data type to be treated as an object.

● Here are the primitive data types and their corresponding wrapper classes:

|  |  |
| --- | --- |
| Primitive type | Wrapper class |
| Integer types | |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| Real types | |
| float | Float |
| double | Double |
| Other types | |
| boolean | Boolean |
| char | Character |

● A wrapper class:

✓ Is required to store primitive data types in an ArrayList.

✓ Has fields for testing the limits of a primitive data type.

✓ Has methods from converting data to primitive data types.

● A wrapper class has several fields and methods for working with primitive data:

| Field / method | Purpose |
| --- | --- |
| MIN\_VALUE | Return the minimum possible value of data type. |
| MAX\_VALUE | Return the maximum possible value of data type. |
| parseByte(<expr>)  parseShort(<expr>)  parseInt(<expr>)  parseLong(<expr>)  parseFloat(<expr>)  parseDouble(<expr>) | Attempt to convert <expr> to a number and return a **primitive value**. If it's an invalid number, exception NumberFormatException is triggered. |
| valueOf(<expr>) | Attempt to convert <expr> to a number and return an **object**. If it's an invalid number, exception NumberFormatException is triggered. |

● **Boxing** is the conversion of a primitive value to an object of its corresponding wrapper class.

● **Unboxing** is the conversion of an object of a wrapper class to its corresponding primitive value.

● See **Wrapper classes** sample application on Blackboard.

**Characters**

● A **character** is a single character enclosed in single quotes.

● A character literal is a [Unicode character](http://unicode-table.com/en/) in the range 0 to 65,535 (or hex FFFF).

● Character is a wrapper class containing several fields and methods for working with Unicode characters.

● Here are some **static** Character methods. See the complete list at [docs.oracle.com/javase/8/docs/api/java/lang/Character.html](http://docs.oracle.com/javase/8/docs/api/java/lang/Character.html).

| Method | Purpose |
| --- | --- |
| isDigit(<char>) | Return *true* if <char> is a digit, otherwise return *false*. |
| isLetter(<char>) | Return *true* if <char> is a letter, otherwise return *false*. |
| isSpaceChar(<char>) | Return *true* if <char> is a space, otherwise return *false*. |
| isWhitespace(<char>) | Return *true* if <char> is a whitespace character, otherwise return *false*. |
| isUpperCase(<char>) | Return *true* if <char> is an upper case character, otherwise return *false*. |
| isLowerCase(<char>) | Return *true* if <char> is a lower case character, otherwise return *false*. |
| compare(<char-1>,<char-2>) | ● If <char-1> is alphabetically after <char-2>, return the positive difference between the characters.  ● If <char-1> and <char-2> are identical, return 0.  ● If <char-1> is alphabetically before <char-2>, return the negative difference between the characters. |
| toUpperCase(<char>) | Return upper case equivalent of <char>. |
| toLowerCase(<char>) | Return lower case equivalent of <char>. |

● See **Character methods** sample application on Blackboard.

**String tokenizing**

● The **StringTokenizer** class gives a way to parse a line from the keyboard or a file.

● By default, a StringTokenizer object parses a string into substrings/tokens using the following default delimiters:

|  |  |
| --- | --- |
| Character | Description |
| ‘ ‘ | Space |
| \t | Tab |
| \n | Newline |
| \r | Carriage return |
| \f | Form feed |

● A StringTokenizer object splits a string into tokens that do not include the delimiters.

● The set of delimiters may be customized.

● See **String parser** sample application on Blackboard.