Unit 09

Character, String, and StringBuilder classes.

CMPS 251, Fall 2020, Dr. Abdulaziz Al-Ali

Objectives

- Introduce…
  - Character
  - String
  - StringBuilder

Character Class

- A character is a integer value that represents a single character
  - ‘a’, ‘A’, ‘5’, ‘!’, etc.

- There is a `Character` class with lots of useful static methods.

Character Class Methods

- Method **isDefined**:
  - determines whether a character is defined in the Unicode character set.
- Method **isDigit**:
  - determines whether a character is a defined Unicode digit.
- Method **isLetter**:
  - determines whether a character is a letter.
- Method **isLetterOrDigit**:
  - determines whether a character is a letter or a digit.

Character Class Methods

- Method **isLowerCase**:
  - determines whether a character is a lowercase letter.
- Method **isUpperCase**:
  - determines whether a character is an uppercase letter.
- Method **toUpperCase**: a → A
  - converts a character to its uppercase equivalent.
- Method **toLowerCase**: A → a
  - converts a character to its lowercase equivalent.

Demo

- See class *TestChars* inside the sample code of this unit.

String Class

- Constructors?
  - new String()
    - Creates an empty string
  - new String("Hi there")
    - Creates a string containing "Hi there"
  - And more, but these are the most common.

Comment on Strings

- String objects are **immutable**.
  - Their character contents cannot be changed after they created.
  - Class String does not provide methods that allow the contents of a String object to be modified.

  - Once a String object is created, you cannot change it. But, you can create a **new** modified-**copy** of it.

Methods of the String Class

- **length()**
  - Determines the number of characters in a string.
- **charAt(int index)**
  - Returns the character at a specific position in the String.
- **startsWith(String a)** and **endsWith**(String a)
  - Determines whether strings start with or end with a particular set of characters.
- **indexOf** and **lastIndexOf**:
  - search for a specified character or substring in a String.

What happens if

- We use str.charAt(-1) or (5) when str = “abc”?
  - Accessing a character outside the bounds of a String (index less than 0 or greater or equal to the length of the string) results in an exception:

    StringIndexOutOfBoundException

Methods of String Class

- **substring**:
  - returns a new String object by copying part of an existing String object.
  - The method returns a new String object.
  - With one integer argument:
    - specifies the starting index in the original String from which characters are to be copied.
  - With two integer arguments:
    - the starting index from which to copy characters in the original String and the index one beyond the last character to copy.

Methods of String Class

- Method **concat**:
  - concatenates two String objects
  - returns a new String object containing the characters from both original Strings.
  - The original Strings are not modified.
- Method **replace**:
  - return a new String object in which every occurrence of the first char argument is replaced with the second.
  - An overloaded version enables you to replace substrings rather than individual characters.

Methods of String Class

- Method **toUpperCase**:
  - generates a new String with uppercase letters.
- Method **toLowerCase**:
  - returns a new String object with lowercase letters.
- Method **trim**:
  - generates a new String object that removes all whitespace characters that appear at the beginning or end of the String on which trim operates.

Comparing Strings

- Strings are compared using the numeric codes of the characters in the strings.

- Various comparisons
  - **equals**
  - **equalsIgnoreCase**
  - **compareTo**
  - **regionMatches**
  - What happens when we use the equality operator **==** to compare String objects?

Comparing Strings

- Method **equals**:
  - tests any two objects for equality
  - The method returns true if the **contents** of the objects are equal, and false otherwise.
- Method **equalsIgnoreCase**:
  - ignores whether the letters in each String are uppercase or lowercase when performing the comparison.

Tokenization

- When you read a sentence, your mind breaks it into **tokens**
  - individual words and punctuation marks that convey meaning.
- Compilers also perform **tokenization**.
- String method **split**:
  - breaks a String into its component tokens
  - returns an array of Strings.
- Tokens are separated by **delimiters**
  - Typically white-space characters such as space, tab, newline and carriage return.
  - Other characters can also be used as delimiters to separate tokens.

Final Note on Strings

- There are many more methods for use with Strings
- Check the API for details

Demo

- See classes *LookForWord*, *TestStrings* (*until TODO 7*), and (optionally) *SearchBoxController* *(the most fun one!)*

Immutable?

- Remember that Strings are immutable.
- What if we want a string that we can modify and manipulate?

StringBuilder class

- Used to create and manipulate dynamic string information.
  - Modifiable strings
  - Use it when you have lots of string concatenation or modification.
- A StringBuilder can grow dynamically as needed

StringBuilder Constructors

- new StringBuilder():
  - creates a **StringBuilder** with no characters in it and an initial capacity of 16 characters.
- new StringBuilder(int a):
  - creates a **StringBuilder** with no characters in it and the initial capacity specified by the integer argument.
- new StringBuilder(String s):
  - creates a **StringBuilder** containing the characters in the String argument. The initial capacity is the number of characters in the String argument plus 16.
- Method **toString** of class **StringBuilder** returns the **StringBuilder** contents as a **String**.

StringBuilder Methods

- **length** and **capacity** :
  - return the number of characters currently in a **StringBuilder** and the number of characters that can be stored in a without allocating more memory, respectively.
- **setLength**:
  - increases or decreases the length of a **StringBuilder**.
  - If the specified length is less than the current number of characters, the buffer is truncated to the specified length.
  - If the specified length is greater than the number of characters, null characters are appended until the total number of characters in the **StringBuilder** is equal to the specified length.

StringBuilder Methods

- **charAt**:
  - takes an integer argument and returns the character in the StringBuilder at that index. .
- **setCharAt**:
  - takes an integer and a character argument
  - sets the character at the specified position in the **StringBuilder** to the character argument.
- **reverse**:
  - reverses the contents of the **StringBuilder**.
- **append:**
  - Adds more content at the end of the current length.

StringBuilder Methods

- Methods **delete** and **deleteCharAt**:
  - delete characters at any position in a **StringBuilder**.
  - Method **delete** takes two arguments:
    - the starting index and
    - the index one past the end of the characters to delete.
  - Method **deleteCharAt** takes one argument:
    - the index of the character to delete

Demo

- See class *TestStrings* (TODOs 8 – 10)

Check your knowledge

- How can we check if str1 = str2?

- What is the output of this code?

```
String abc = "Alphabets";
abc.substring(0,4);
System.out.println(abc);
```