CIS2571 – Intro to Java

Chapter 9 → Strings

Information consolidated from a variety of textbook and online resources

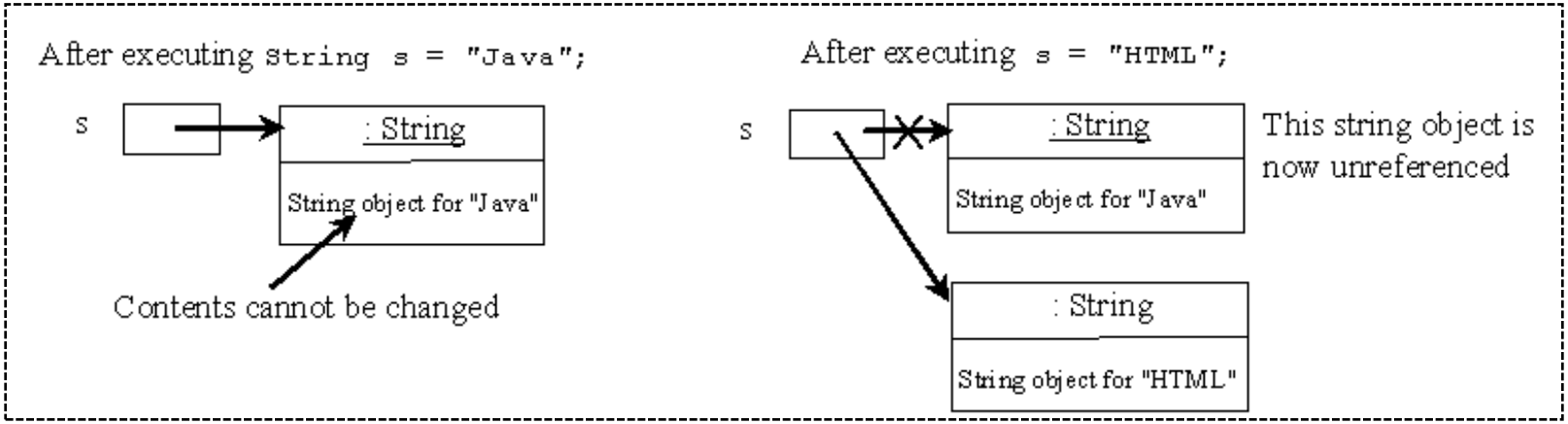
Topic Objectives

- String Class
- Character Class
- StringBuilder/StringBuffer Classes
- Command-Line Arguments

String Class

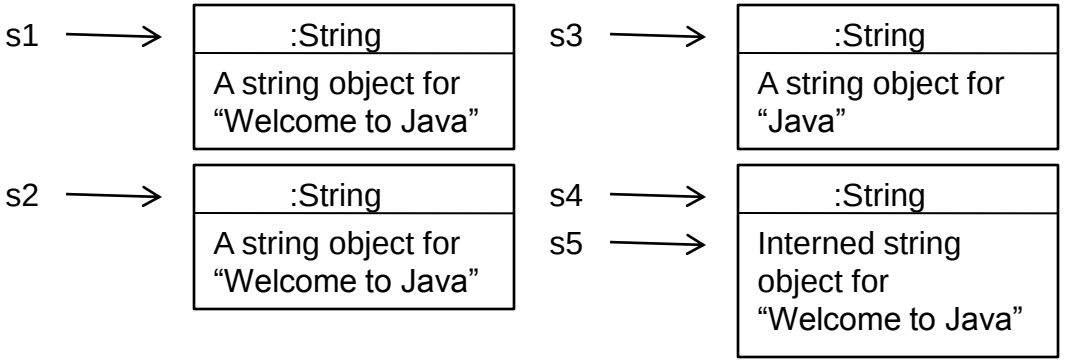
- Many languages treat strings as arrays of characters
- String objects are immutable; contents cannot be changed

```
String s = "Java";
s = "HTML";
```



- To improve efficiency, JVM uses unique instance for string literals with same character sequence
  - Referred to as an interned instance

Strings



- Several overloaded constructors (new object is created)

```
String s1 = new String("Welcome to Java");
String s2 = new String(s1);
String s3 = new String(new char[] {'J', 'a', 'v', 'a'});
```

  - Java also provides a shorthand initializer
    - No new object created if interned object already created

```
String s4 = "Welcome to Java";
String s5 = "Welcome to Java";
```

s1 == s2 is false
s1 == s4 is false
s4 == s5 is true
s1.equals(s4) is true
s1.compareTo(s4) returns value 0
s1.compareTo(s3) returns value > 0
// since ASCII value 'W' > 'J'

- Methods for comparison

```
equals, equalsIgnoreCase,
compareTo, compareToIgnoreCase,
regionMatches, startsWith, endsWith
```

Strings

java.lang.String	
+equals(s1: String): boolean	Returns true if this string is equal to string s1.
+equalsIgnoreCase(s1: String): boolean	Returns true if this string is equal to string s1 case-insensitive.
+compareTo(s1: String): int	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater than, equal to, or less than s1.
+compareToIgnoreCase(s1: String): int	Same as compareTo except that the comparison is case-insensitive.
+regionMatches(toffset: int, s1: String, offset: int, len: int): boolean	Returns true if the specified subregion of this string exactly matches the specified subregion in string s1.
+regionMatches(ignoreCase: boolean, toffset: int, s1: String, offset: int, len: int): boolean	Same as the preceding method except that you can specify whether the match is case-sensitive.
+startsWith(prefix: String): boolean	Returns true if this string starts with the specified prefix.
+endsWith(suffix: String): boolean	Returns true if this string ends with the specified suffix.

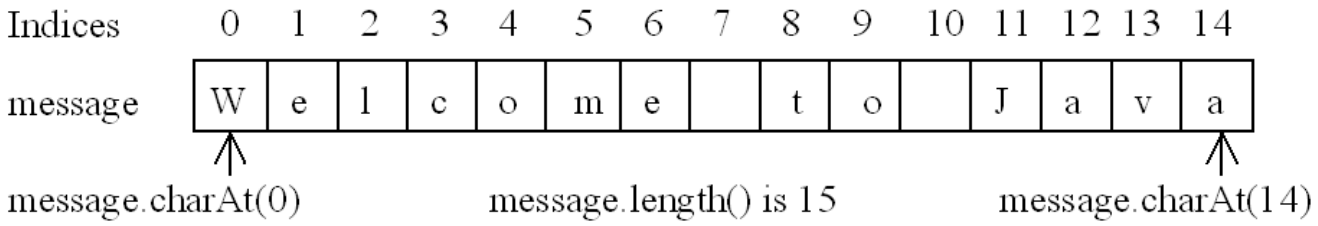
```
System.out.println("Welcome to Java".
      regionMatches(11, "Java", 0, 4));
  // displays true
```

Strings

- Methods for obtaining length, retrieving individual characters, and concatenating Strings

java.lang.String	
+length(): int	Returns the number of characters in this string.
+charAt(index: int): char	Returns the character at the specified index from this string.
+concat(s1: String): String	Returns a new string that concatenate this string with string s1.

```
String message = "Welcome to Java";
message.length() → returns 15
message.[0] → invalid
message.charAt(0) → returns 'W'
```

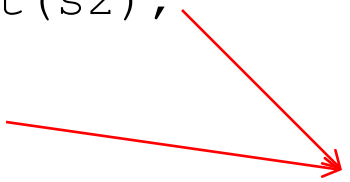


Strings

- Methods for obtaining length, retrieving individual characters, and concatenating Strings

java.lang.String	
+length(): int	Returns the number of characters in this string.
+charAt(index: int): char	Returns the character at the specified index from this string.
+concat(s1: String): String	Returns a new string that concatenate this string with string s1.

```
String s1 = new String("Welcome");
String s2 = new String(" to Java");
String s3 = s1.concat(s2);
OR
String s3 = s1 + s2;
```



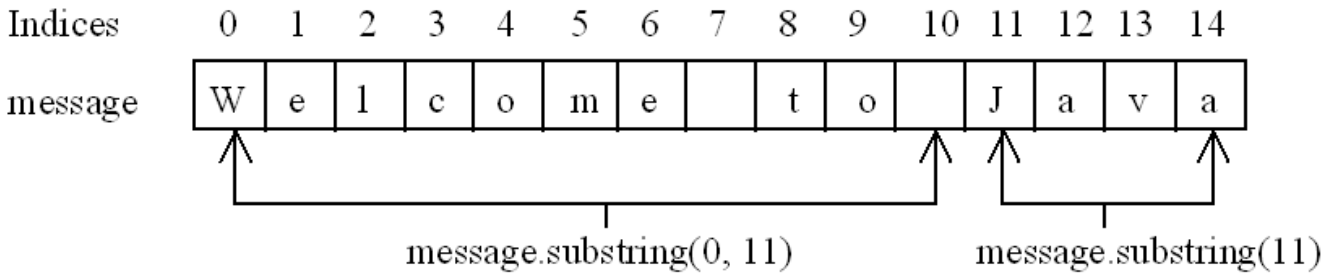
"Welcome to Java"

Strings

- Methods for obtaining substrings

```
String s1 = "Welcome to Java";
String s2 = s1.substring(0, 11) + "HTML";
```

java.lang.String	
+subString(beginIndex: int): String	Returns this string's substring that begins with the character at the specified beginIndex and extends to the end of the string, as shown in Figure 9.6.
+subString(beginIndex: int, endIndex: int): String	Returns this string's substring that begins at the specified beginIndex and extends to the character at index endIndex – 1, as shown in Figure 9.6. Note that the character at endIndex is not part of the substring.



Strings

- Methods for converting, replacing, and splitting Strings

java.lang.String	
+toLowerCase(): String	Returns a new string with all characters converted to lowercase.
+toUpperCase(): String	Returns a new string with all characters converted to uppercase.
+trim(): String	Returns a new string with blank characters trimmed on both sides.
+replace(oldChar: char, newChar: char): String	Returns a new string that replaces all matching character in this string with the new character.
+replaceFirst(oldString: String, newString: String): String	Returns a new string that replaces the first matching substring in this string with the new substring.
+replaceAll(oldString: String, newString: String): String	Returns a new string that replace all matching substrings in this string with the new substring.
+split(delimiter: String): String[]	Returns an array of strings consisting of the substrings split by the delimiter.

→Examples

Strings

- Methods for converting, replacing, and splitting Strings

"Welcome".toLowerCase() → returns a new string, welcome.

"Welcome".toUpperCase() → returns a new string, WELCOME.

"\tGood Night \n".trim() → returns a new string, Good Night.

"Welcome".replace('e', 'A') → returns a new string, WAlcomA.

"Welcome".replaceFirst("e", "AB") → returns a new string, WABlcome.

"Welcome".replace("e", "AB") → returns a new string, WABlcomAB.

"Welcome".replace("el", "AB") → returns a new string, WABcome.

```
String[] tokens = "Java#HTML#Perl".split("#", 0);
for (int i = 0; i < tokens.length; i++)
  System.out.print(tokens[i] + " ");
```

→ **displays Java HTML Perl**

Strings

- Methods for matching, replacing or splitting a string by specifying a pattern

Regular Expression	Matches	Example
x	a specified character x	Java matches Java
.	any single character	Java matches J..a
(ab|cd)	a, b, or c	ten matches t(en|im)
[abc]	a, b, or c	Java matches Ja[uvwx]a
[^abc]	any character except a, b, or c	Java matches Ja[^ars]a
[a-z]	a through z	Java matches [A-M]av[a-d]
[^a-z]	any character except a through z	Java matches Jav[^b-d]
[a-e[m-p]]	a through e or m through p	Java matches [A-G[I-M]]av[a-d]
[a-e&&[c-p]]	intersection of a-e with c-p	Java matches [A-P&&[I-M]]av[a-d]
\d	a digit, same as [1-9]	Java2 matches "Java[\\d]"
\D	a non-digit	$Java matches "[\\D][\\D]ava"
\w	a word character	Java matches "[\\w]ava"
\W	a non-word character	$Java matches "[\\W][\\w]ava"
\s	a whitespace character	"Java 2" matches "Java\\s2"
\S	a non-whitespace char	Java matches "[\\S]ava"
*p**	zero or more occurrences of pattern *p*	Java matches "[\\w]*"
p+	one or more occurrences of pattern *p*	Java matches "[\\w]+"
p?	zero or one occurrence of pattern *p*	Java matches "[\\w]?Java" Java matches "[\\w]?ava"
p{n}	exactly n occurrences of pattern *p*	Java matches "[\\w]{4}"
p{n,}	at least n occurrences of pattern *p*	Java matches "[\\w]{3,}"
p{n,m}	between n and m occurrences (inclusive)	Java matches "[\\w]{1,9}"

Strings

java.lang.String	
+matches(regex: String): boolean	Returns true if this string matches the pattern.
+replaceAll(regex: String, replacement: String): String	Returns a new string that replaces all matching substrings with the replacement.
+replaceFirst(regex: String, replacement: String): String	Returns a new string that replaces the first matching substring with the replacement.
+split(regex: String): String[]	Returns an array of strings consisting of the substrings split by the matches.

- Methods for matching, replacing or splitting a string by specifying a pattern
  - Regular Expresson → See Supplement III.H
    - .* → matches any zero or more characters

returns true

```
"Java".matches("Java");
"Java".equals("Java");
"Java is fun".matches("Java.*");
"Java is cool".matches("Java.*");
```

Strings

- Methods for matching, replacing or splitting a string by specifying a pattern
  - Regular Expresson → See Supplement III.H
    - [$+#] → matches $, +, or #

```
String s = "a+b$#c".replaceAll("[$+#]", "NNN");
```

`System.out.println(s);` → displays aNNNbNNNNNNc

```
String[] tokens =
"Java,C?C#,C++".split("[.,:;?]");
for (int i = 0; i < tokens.length; i++)
  System.out.println(tokens[i]);
```

→ string is split into array tokens, output one to a line

Strings

- Methods for **finding character or substring** in a String

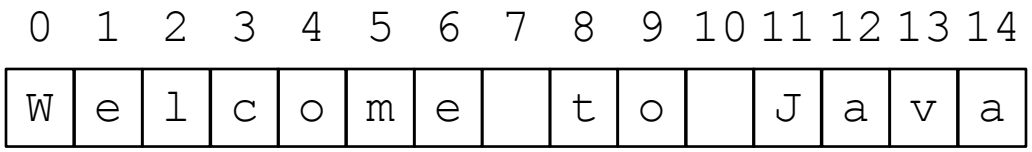
java.lang.String	
+indexOf(ch: char): int	Returns the index of the first occurrence of ch in the string. Returns -1 if not matched.
+indexOf(ch: char, fromIndex: int): int	Returns the index of the first occurrence of ch after fromIndex in the string. Returns -1 if not matched.
+indexOf(s: String): int	Returns the index of the first occurrence of string s in this string. Returns -1 if not matched.
+indexOf(s: String, fromIndex: int): int	Returns the index of the first occurrence of string s in this string after fromIndex. Returns -1 if not matched.
+lastIndexOf(ch: int): int	Returns the index of the last occurrence of ch in the string. Returns -1 if not matched.
+lastIndexOf(ch: int, fromIndex: int): int	Returns the index of the last occurrence of ch before fromIndex in this string. Returns -1 if not matched.
+lastIndexOf(s: String): int	Returns the index of the last occurrence of string s. Returns -1 if not matched.
+lastIndexOf(s: String, fromIndex: int): int	Returns the index of the last occurrence of string s before fromIndex. Returns -1 if not matched.

→Examples

Strings

- Methods for **finding character or substring** in a String

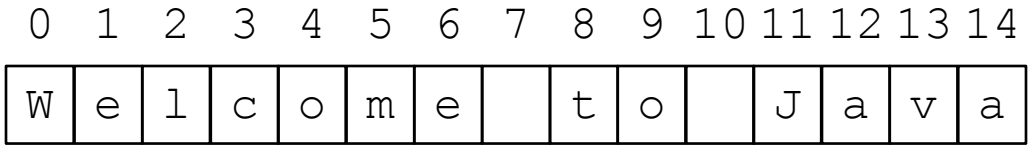
```
"Welcome to Java".indexOf('W') returns 0.
"Welcome to Java".indexOf('o') returns 4.
"Welcome to Java".indexOf('o', 5) returns 9.
"Welcome to Java".indexOf("come") returns 3.
"Welcome to Java".indexOf("Java", 5) returns 11.
"Welcome to Java".indexOf("java", 5) returns -1.
```



Strings

- Methods for **finding character or substring** in a String

```
"Welcome to Java".lastIndexOf('W') returns 0.
"Welcome to Java".lastIndexOf('o') returns 9.
"Welcome to Java".lastIndexOf('o', 5) returns 4.
"Welcome to Java".lastIndexOf("come") returns 3.
"Welcome to Java".lastIndexOf("Java", 5) returns -1.
"Welcome to Java".lastIndexOf("Java") returns 11.
```



Strings

- Methods for **conversion between Strings and Arrays**
  - Strings are not arrays, but a **string can be converted into an array of characters**

```
char[] chars = "Java".toCharArray();
```

→ creates array chars with elements 'J', 'a', 'v', 'a'

```
char[] dst = {'J', 'A', 'V', 'A', '1', '3', '0', '1'};
```

→ `getChars(int srcBegin, int srcEnd, char[] dest, int destBegin)`

```
"CIS2571".getChars(3, 7, dst, 4); // copies chars
```

→ dst becomes {'J', 'A', 'V', 'A', '2', '5', '7', '1'}

  - An **array of characters can be converted into a string**

```
String str = new String(new char[] {'J', 'a', 'v', 'a'});
String str = String.valueOf(new char[] {'J', 'a', 'v', 'a'});
```

Strings

- Static method for **converting characters and numeric values to Strings**
  - Convert character and numeric values to strings with different parameter types (`char`, `double`, `long`, `int`, and `float`)

`String.valueOf(5.44)` → returns string "5.44"

`Double.parseDouble("5.44")` → returns double 5.44

- Static methods for **formatting Strings**
  - Similar to `System.out.printf()` method, except formatted string is returned

```
String s = String.format("%5.2f", 45.556);
```

→ returns formatted string "45.56"

See 9.1 CheckPalindrome.java
See 9.2 HexToDecimalConversion.java

Character Class

- In **java.lang** package
- Enables primitive data type to be treated as object

java.lang.Character	
+Character(value: char)	Constructs a character object with char value
+charValue(): char	Returns the char value from this object
+compareTo(anotherCharacter: Character): int	Compares this character with another
+equals(anotherCharacter: Character): boolean	Returns true if this character equals to another
+isDigit(ch: char): boolean	Returns true if the specified character is a digit
+isLetter(ch: char): boolean	Returns true if the specified character is a letter
+isLetterOrDigit(ch: char): boolean	Returns true if the character is a letter or a digit
+isLowerCase(ch: char): boolean	Returns true if the character is a lowercase letter
+isUpperCase(ch: char): boolean	Returns true if the character is an uppercase letter
+toLowerCase(ch: char): char	Returns the lowercase of the specified character
+toUpperCase(ch: char): char	Returns the uppercase of the specified character

Character Class

- Examples:

```
Character charObject = new Character('b');
```

```
charObject.compareTo(new Character('a'))  → returns 1
charObject.compareTo(new Character('b'))  → returns 0
charObject.compareTo(new Character('c'))  → returns -1
charObject.compareTo(new Character('d')   → returns –2
charObject.equals(new Character('b'))     → returns true
charObject.equals(new Character('d'))     → returns false
charObject.charValue()                    → returns 98 (ASCII value for 'b')
charObject.isCharacter(charObject)        → returns true
Character.isDigit(charObject)             → returns false
```

See 9.3 CountEachLetter.java

StringBuilder/StringBuffer Class

- Can be used wherever a string is used
- More flexible than String
  - Can add, insert, or append new contents (*mutable*)
  - String object fixed when created
  - Some optimization such as sharing interned strings
- StringBuilder and StringBuffer have same constructors and methods
- StringBuffer → should be used if needs to be accessible by multiple tasks concurrently
  - Thread-safe
- StringBuilder → more efficient if accessed by single task

StringBuilder/StringBuffer Class Constructors and Methods

java.lang.StringBuilder	
+StringBuilder()	Constructs an empty string builder with capacity 16.
+StringBuilder(capacity: int)	Constructs a string builder with the specified capacity.
+StringBuilder(s: String)	Constructs a string builder with the specified string.

java.lang.StringBuilder	
+toString(): String	Returns a string object from the string builder.
+capacity(): int	Returns the capacity of this string builder.
+charAt(index: int): char	Returns the character at the specified index.
+length(): int	Returns the number of characters in this builder.
+setLength(newLength: int): void	Sets a new length in this builder.
+substring(startIndex: int): String	Returns a substring starting at startIndex.
+substring(startIndex: int, endIndex: int): String	Returns a substring from startIndex to endIndex-1.
+trimToSize(): void	Reduces the storage size used for the string builder.

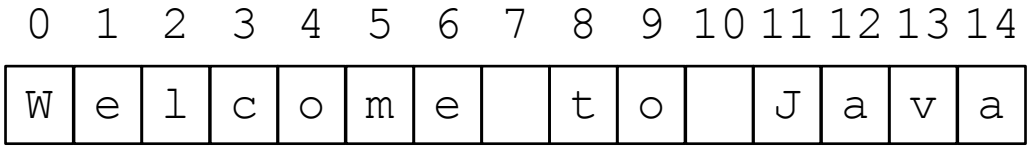
StringBuilder/StringBuffer Class Methods

java.lang.StringBuilder	
+append(data: char[]): StringBuilder	Appends a char array into this string builder.
+append(data: char[], offset: int, len: int): StringBuilder	Appends a subarray in data into this string builder.
+append(v: *aPrimitiveType*): StringBuilder	Appends a primitive type value as a string to this builder.
+append(s: String): StringBuilder	Appends a string to this string builder.
+delete(startIndex: int, endIndex: int): StringBuilder	Deletes characters from startIndex to endIndex.
+deleteCharAt(index: int): StringBuilder	Deletes a character at the specified index.
+insert(index: int, data: char[], offset: int, len: int): StringBuilder	Inserts a subarray of the data in the array to the builder at the specified index.
+insert(offset: int, data: char[]): StringBuilder	Inserts data into this builder at the position offset.
+insert(offset: int, b: *aPrimitiveType*): StringBuilder	Inserts a value converted to a string into this builder.
+insert(offset: int, s: String): StringBuilder	Inserts a string into this builder at the position offset.
+replace(startIndex: int, endIndex: int, s: String): StringBuilder	Replaces the characters in this builder from startIndex to endIndex with the specified string.
+reverse(): StringBuilder	Reverses the characters in the builder.
+setCharAt(index: int, ch: char): void	Sets a new character at the specified index in this builder.

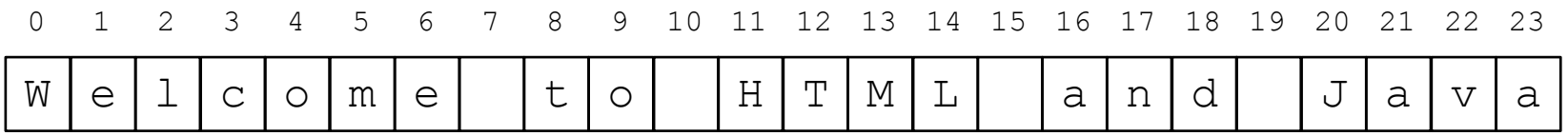
StringBuilder/StringBuffer Class Methods

- Examples:

```
StringBuilder sb = new StringBuilder();
System.out.println(sb.capacity()); // displays 16
System.out.println(sb.length()); // displays 0
sb.append("Welcome to");
sb.append(' ');
sb.append("Java"); // Welcome to Java
System.out.println(sb.capacity()); // displays 16
System.out.println(sb.length()); // displays 15
```



StringBuilder/StringBuffer Class Methods



- Examples:

```
sb.insert(11, "HTML and ");
      // Welcome to HTML and Java
System.out.println(sb.capacity()); // displays 34
System.out.println(sb.length()); // displays 24
sb.delete(8, 20); // Welcome Java
System.out.println(sb.capacity()); // displays 34
System.out.println(sb.length()); // displays 12
sb.reverse(); // avaJ emocleW
sb.setCharAt(3, 'j'); // avaj emocleW
```

See 9.4 PalindromeIgnoreNonAlphanumeric.java

Command-Line Arguments

- Methods can have passed parameters → main is a method

```
public static void main(String[] args)
```

- Strings are passed to a main method from the command line

```
java ClassNameWithMain arg0 arg1 arg2
java TestMain "First num" alpha 53
```

- When main is invoked, Java interpreter creates an array to hold the command line arguments and passes array reference

args[0] → *"First num"*

args[1] → *"alpha"*

args[2] → *"53"*

args.length → *3*

See 9.5 Calculator.java