Characters and Strings

Introduction to Programming and Computational Problem Solving - 2

CSE 8B

Lecture 4

Announcements

- Assignment 1 is due today, 11:59 PM
- Assignment 2 will be released today
  - Due April 19, 11:59 PM
- Educational research study
  - April 14, weekly survey

Characters and strings

- Character data type (i.e., `char`)
- Comparing and testing characters
- String data type (i.e., `String`)
- Simple string methods (e.g., number of characters in a string)
- Reading a character and string from the console

Data types

- Java is a strongly typed language
  - **Programmers must explicitly identify the type of every variable, method, and object**

char data type

```
char letter = 'A'; // ASCII
char numChar = '4'; // ASCII
char letter = '\u0041'; // Unicode
char numChar = '\u0034'; // Unicode
```

- Java characters use Unicode, a 16-bit encoding scheme established by the Unicode Consortium to support the interchange, processing, and display of written texts in the world's diverse languages

- Unicode takes two bytes, preceded by \u, expressed in four hexadecimal numbers that run from \u0000 to \uFFFF
    - Unicode can represent 65536 characters

Common and special characters

Characters	Code Value in Decimal	Unicode Value
`'0'` to `'9'`	48 to 57	\u0030 to \u0039
`'A'` to `'Z'`	65 to 90	\u0041 to \u005A
`'a'` to `'z'`	97 to 122	\u0061 to \u007A

Escape Sequence	Name	Unicode Code	Decimal Value
`\b`	Backspace	`\u0008`	8
`\t`	Tab	`\u0009`	9
`\n`	Linefeed	`\u000A`	10
`\f`	Formfeed	`\u000C`	12
`\r`	Carriage Return	`\u000D`	13
`\\`	Backslash	`\u005C`	92
`\"`	Double Quote	`\u0022`	34

Comparing and testing characters

```
if (ch >= 'A' && ch <= 'Z')
  System.out.println(ch + " is an uppercase letter");
else if (ch >= 'a' && ch <= 'z')
  System.out.println(ch + " is a lowercase letter");
else if (ch >= '0' && ch <= '9')
  System.out.println(ch + " is a numeric character");
```

Relational and logical operators will be covered next lecture

Comparing and testing characters

- The `Character` class
  - Java 8 API documentation
    - https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html
  - Java 11 API documentation
    - https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Character.html

Method	Description
isDigit(ch)	Returns true if the specified character is a digit.
isLetter(ch)	Returns true if the specified character is a letter.
isLetterOfDigit(ch)	Returns true if the specified character is a letter or digit.
isLowerCase(ch)	Returns true if the specified character is a lowercase letter.
isUpperCase(ch)	Returns true if the specified character is an uppercase letter.
toLowerCase(ch)	Returns the lowercase of the specified character.
toUpperCase(ch)	Returns the uppercase of the specified character.

Casting between char and numeric data types

```
int i = 'a'; // Same as int i = (int)'a';


char c = 97; // Same as char c = (char)97;
```

String type

- The char type only represents one character
- To represent a string of characters, use the String type
- String is a predefined class in the Java library (just like the System class and Scanner class)
  - Java 8 API documentation
    - https://docs.oracle.com/javase/8/docs/api/java/lang/String.html
  - Java 11 API documentation
    - https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html

String type

- `String` is a predefined class in the Java library

    `String message = "Welcome to Java";`

- The `String` type is not a primitive type; it is known as a reference type

    – Any Java class can be used as a reference type for a variable

Simple `String` methods

Method	Description
`length()`	Returns the number of characters in this string.
`charAt(index)`	Returns the character at the specified index from this string.
`concat(s1)`	Returns a new string that concatenates this string with string s1.
`toUpperCase()`	Returns a new string with all letters in uppercase.
`toLowerCase()`	Returns a new string with all letters in lowercase.
`trim()`	Returns a new string with whitespace characters trimmed on both sides.

- These methods can only be invoked from a specific string instance
 - These methods are called instance methods

Instance methods vs static methods

- These methods can only be invoked from a specific string instance
  - These methods are called instance methods
  - The syntax to invoke an instance method is
    `referenceVariable.methodName(arguments)`
- A non-instance method is called a static method
  - **A static method can be invoked without using an object** (i.e., they are not tied to a specific object instance)
  - The syntax to invoke a static method is
    `ClassName.methodName(arguments)`
  - For example, all the methods defined in the `Math` class are static methods

Methods will be covered next week

Getting characters from a string

```
String message = "Welcome to Java";
System.out.println("The first character in message is "
    + message.charAt(0));
```
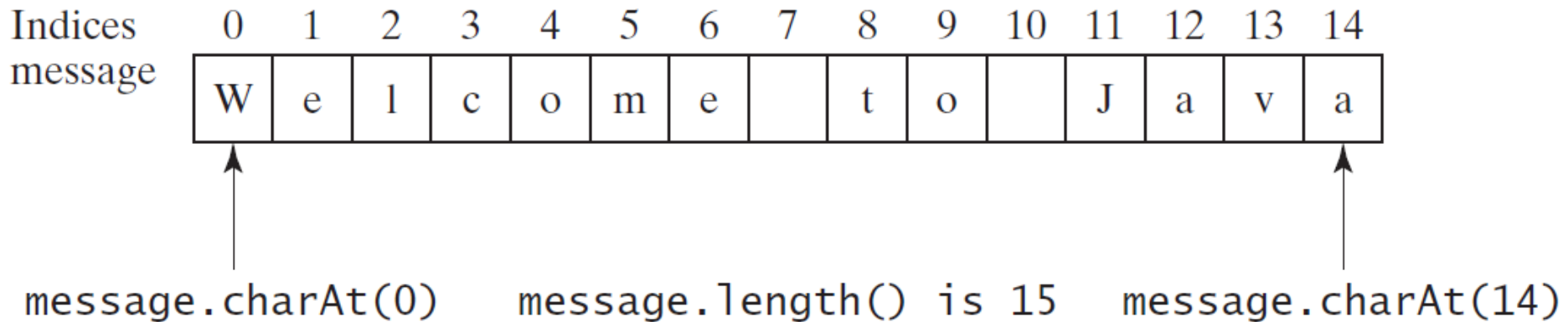
String concatenation

```
String s3 = s1.concat(s2); // These two are
String s3 = s1 + s2;        // equivalent

// Three strings are concatenated
String message = "Welcome " + "to " + "Java";

// String Chapter is concatenated with number 2
String s = "Chapter" + 2; // s becomes Chapter2

// String Supplement is concatenated with character B
String s1 = "Supplement" + 'B'; // s1 becomes SupplementB
```

Reading a string from the console

```
Scanner input = new Scanner(System.in);
System.out.print("Enter three words separated by spaces: ");
String s1 = input.next();
String s2 = input.next();
String s3 = input.next();
System.out.println("s1 is " + s1);
System.out.println("s2 is " + s2);
System.out.println("s3 is " + s3);
```

Reading a character from the console

```
Scanner input = new Scanner(System.in);
System.out.print("Enter a character: ");
String s = input.nextLine();
char ch = s.charAt(0);
System.out.println("The character entered is " + ch);
```

Explicit import and implicit Import

- At top of source file
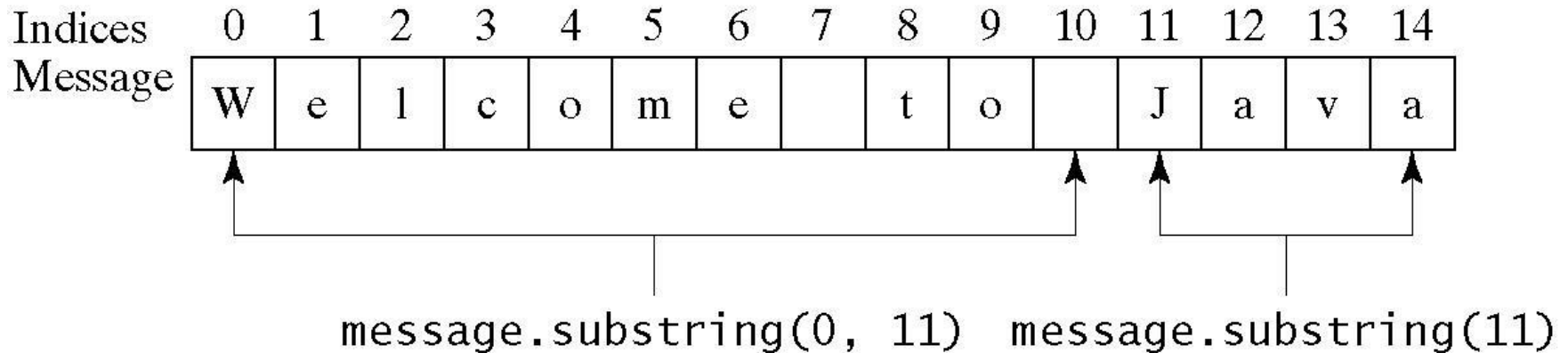
```
import java.util.Scanner; // Explicit Import

import java.util.*; // Implicit import
```

Comparing strings

Method	Description
`equals(s1)`	Returns true if this string is equal to string `s1`.
`equalsIgnoreCase(s1)`	Returns true if this string is equal to string `s1`; it is case insensitive.
`compareTo(s1)`	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)`	Same as `compareTo` except that the comparison is case insensitive.
`startsWith(prefix)`	Returns true if this string starts with the specified prefix.
`endsWith(suffix)`	Returns true if this string ends with the specified suffix.

Substrings

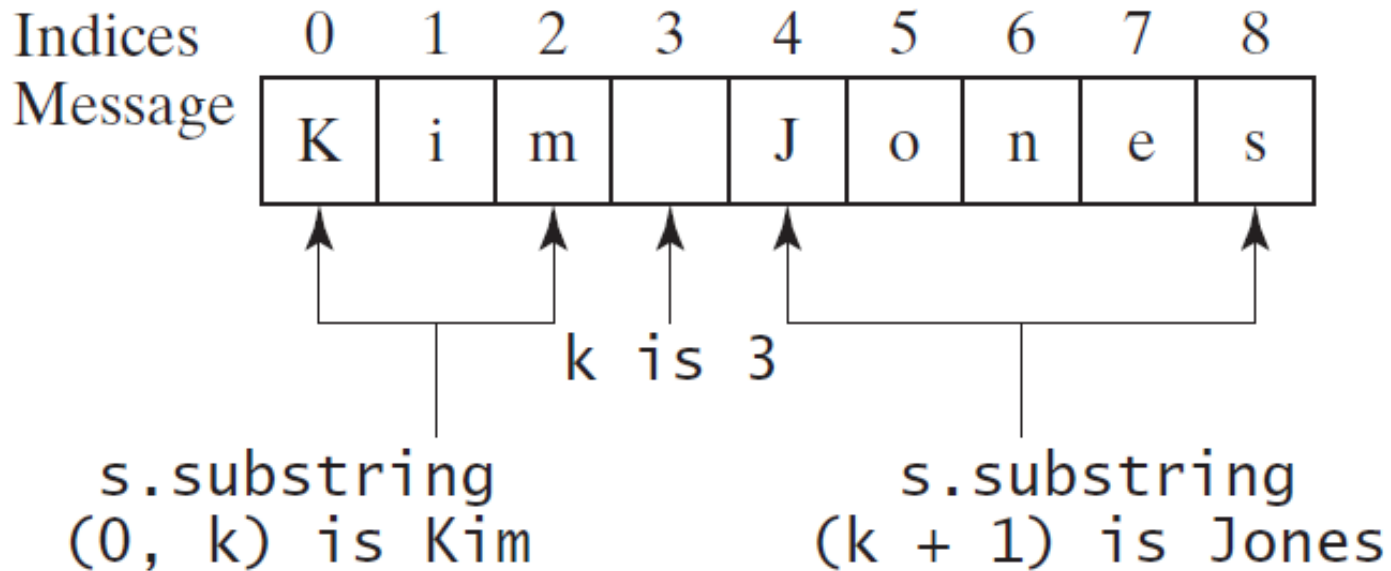| Method	Description
`substring(beginIndex)`	Returns this string's substring that begins with the character at the specified `beginIndex` and extends to the end of the string.
`substring(beginIndex, endIndex)`	Returns this string's substring that begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Note that the character at `endIndex` is not part of the substring.


```
Indices   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
Message   W   e   l   c   o   m   e       t   o       J   a   v   a
```

message.substring(0, 11)   message.substring(11)

Finding a character or a substring in a string

Method	Description
`indexOf(ch)`	Returns the index of the first occurrence of `ch` in the string. Returns −1 if not matched.
`indexOf(ch, fromIndex)`	Returns the index of the first occurrence of `ch` after `fromIndex` in the string. Returns −1 if not matched.
`indexOf(s)`	Returns the index of the first occurrence of string `s` in this string. Returns −1 if not matched.
`indexOf(s, fromIndex)`	Returns the index of the first occurrence of string `s` in this string after `fromIndex`. Returns −1 if not matched.
`lastIndexOf(ch)`	Returns the index of the last occurrence of `ch` in the string. Returns −1 if not matched.
`lastIndexOf(ch, fromIndex)`	Returns the index of the last occurrence of `ch` before `fromIndex` in this string. Returns −1 if not matched.
`lastIndexOf(s)`	Returns the index of the last occurrence of string `s`. Returns −1 if not matched.
`lastIndexOf(s, fromIndex)`	Returns the index of the last occurrence of string `s` before `fromIndex`. Returns −1 if not matched.

Finding a character or a substring in a string

```
int k = s.indexOf(' ');
String firstName = s.substring(0, k);
String lastName = s.substring(k + 1);
```


Indices	0	1	2	3	4	5	6	7	8
Message	K	i	m		J	o	n	e	s

k is 3

s.substring (0, k) is Kim

s.substring (k + 1) is Jones

Conversion between strings and numbers

```
int intValue =
  Integer.parseInt(intString);
double doubleValue =
  Double.parseDouble(doubleString);

String s = number + "";
```
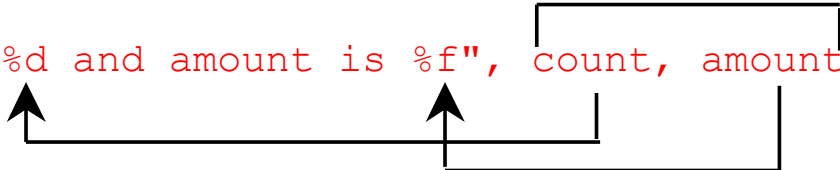
Formatting output

- Use the `printf` statement

  `System.out.printf(format, items);`

- Where `format` is a string that may consist of substrings and format specifiers

  - A format specifier specifies how an item should be displayed

  - Each specifier begins with a percent sign

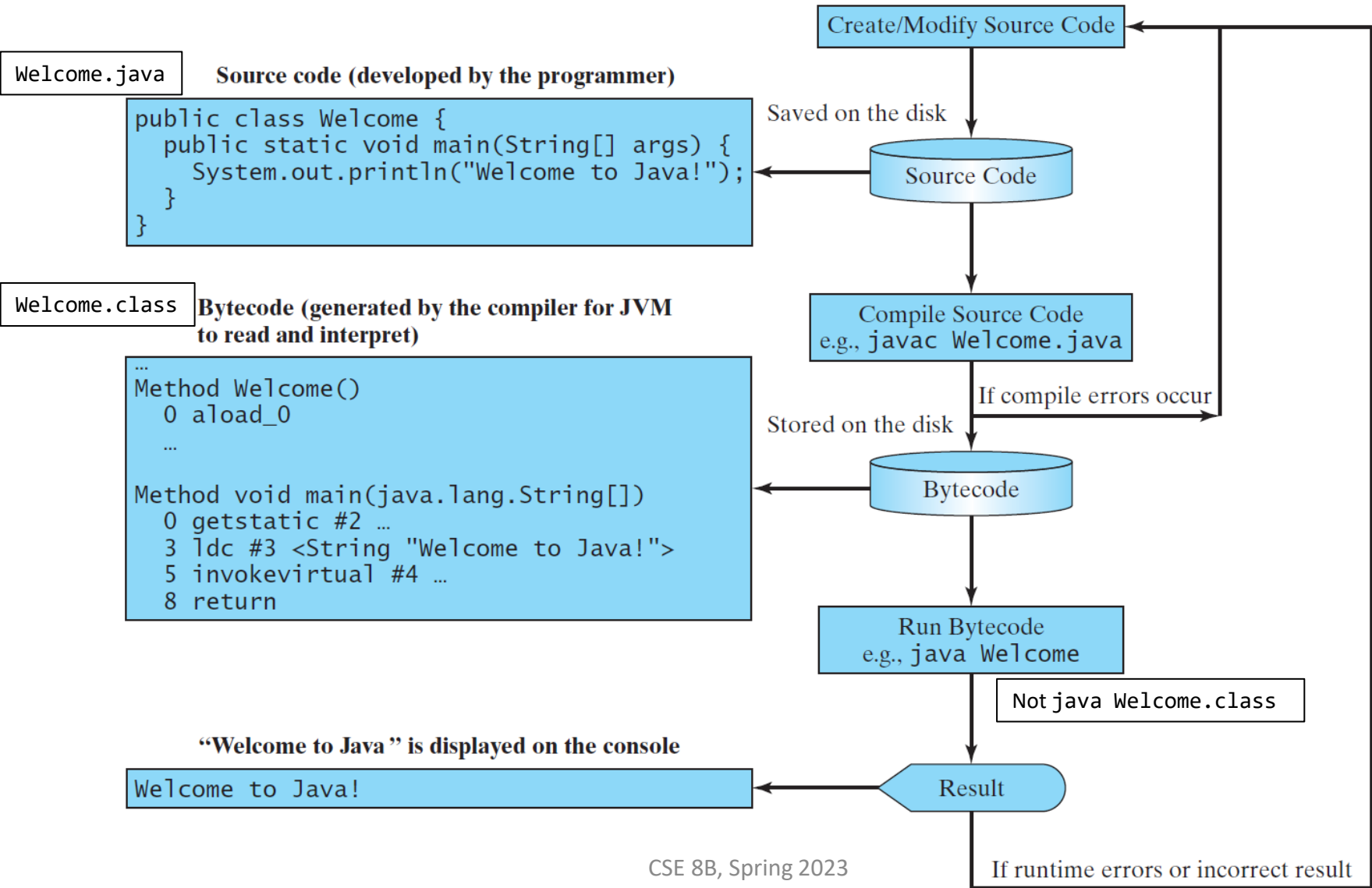 - An item may be a numeric value, character, Boolean value, or a string

Common specifiers

Specifier	Output	Example
%b	a boolean value	true or false
%c	a character	'a'
%d	a decimal integer	200
%f	a floating-point number	45.460000
%e	a number in standard scientific notation	4.556000e+01
%s	a string	"Java is cool"

```
int count = 5;
double amount = 45.56;
System.out.printf("count is %d and amount is %f", count, amount);
```

items

```
display           count is 5 and amount is 45.560000
```

Developing, compiling, and running Java programs

Next Lecture

- Selections