Chapter 4: Mathematical Functions, Characters, and Strings

CS1: Java Programming
Colorado State University

Original slides by Daniel Liang
Modified slides by Chris Wilcox

Motivations
Suppose you need to estimate the area enclosed by four cities, given the GPS locations (latitude and longitude) of these cities, as shown in the following diagram. How would you write a program to solve this problem? You will be able to write such a program after completing this chapter.

Objectives
✦ To solve mathematics problems by using the methods in the Math class (§4.2).
✦ To represent characters using the char type (§4.3).
✦ To encode characters using ASCII and Unicode (§4.3.1).
✦ To represent special characters using the escape sequences (§4.3.2).
✦ To cast a numeric value to a character and cast a character to an integer (§4.3.3).
✦ To compare and test characters using the static methods in the Character class (§4.3.4).
✦ To introduce objects and instance methods (§4.4).
✦ To represent strings using the String objects (§4.4).
✦ To return the string length using the length() method (§4.4.1).
✦ To return a character in the string using the charAt(i) method (§4.4.2).
✦ To use the + operator to concatenate strings (§4.4.3).
✦ To read strings from the console (§4.4.5).
✦ To compare strings using the equals method and the compareTo methods (§4.4.6).
✦ To obtain substrings (§4.4.7).
✦ To find a character or a substring in a string using the indexOf method (§4.4.8).
✦ To program using characters and strings (GuessYourBirthday) (§4.5.1).
✦ To convert a hexadecimal character to a decimal value (HexDigit2Dec) (§4.5.2).
✦ To revise the lottery program using strings (LotteryUsingStrings) (§4.5.3).
✦ To format output using the System.out.printf method (§4.6).

Mathematical Functions
Java provides many useful methods in the Math class for performing common mathematical functions.
The Math Class

✦ Class constants:
  - PI
  - E
✦ Class methods:
  - Trigonometric Methods
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods

Trigonometric Methods

✦ sin(double a)

✦ cos(double a)

✦ tan(double a)

✦ acos(double a)

✦ asin(double a)

✦ atan(double a)

Radians toRadians(90)

Examples:

Math.sin(0) returns 0.0
Math.sin(Math.PI / 6) returns 0.5
Math.sin(Math.PI / 2) returns 1.0
Math.cos(0) returns 1.0
Math.cos(Math.PI / 6) returns 0.866
Math.cos(Math.PI / 2) returns 0

Exponent Methods

✦ exp(double a)

Returns e raised to the power of a.

✦ log(double a)

Returns the natural logarithm of a.

✦ log10(double a)

Returns the 10-based logarithm of a.

✦ pow(double a, double b)

Returns a raised to the power of b.

✦ sqrt(double a)

Returns the square root of a.

Examples:

Math.exp(1) returns 2.71
Math.log(2.71) returns 1.0
Math.pow(2, 3) returns 8.0
Math.pow(3, 2) returns 9.0
Math.pow(3.5, 2.5) returns 22.91765
Math.sqrt(4) returns 2.0
Math.sqrt(10.5) returns 3.24

Rounding Methods

✦ double ceil(double x)

x rounded up to its nearest integer. This integer is returned as a double value.

✦ double floor(double x)

x is rounded down to its nearest integer. This integer is returned as a double value.

✦ int round(float x)

Return (int) Math.floor(x+0.5).

✦ long round(double x)

Return (long) Math.floor(x+0.5).
### Rounding Methods Examples

- `Math.ceil(2.1)` returns 3.0  
- `Math.ceil(2.0)` returns 2.0  
- `Math.ceil(-2.0)` returns -2.0  
- `Math.ceil(-2.1)` returns -2.0  
- `Math.floor(2.1)` returns 2.0  
- `Math.floor(2.0)` returns 2.0  
- `Math.floor(-2.0)` returns -2.0  
- `Math.floor(-2.1)` returns -3.0  
- `Math.rint(2.1)` returns 2.0  
- `Math.rint(2.0)` returns 2.0  
- `Math.rint(-2.0)` returns -2.0  
- `Math.rint(-2.1)` returns -2.0  
- `Math.rint(2.5)` returns 2.0  
- `Math.rint(-2.5)` returns -2.0  
- `Math.round(2.6f)` returns 3  
- `Math.round(2.0)` returns 2  
- `Math.round(-2.0f)` returns -2  
- `Math.round(-2.6)` returns -3

### min, max, and abs

#### max(a, b) and min(a, b)
- Returns the maximum or minimum of two parameters.

#### abs(a)
- Returns the absolute value of the parameter.

#### random()
- Returns a random double value in the range [0.0, 1.0).

### The random Method

Generates a random double value greater than or equal to 0.0 and less than 1.0 (0 <= Math.random() < 1.0).

**Examples:**

- `(int)(Math.random() * 10)` returns a random integer between 0 and 9.
- `50 + (int)(Math.random() * 50)` returns a random integer between 50 and 99.

In general,

- `a + Math.random() * b` returns a random number between `a` and `a + b`, excluding `a + b`.

### Case Study: Computing Angles of a Triangle

Write a program that prompts the user to enter the x- and y-coordinates of the three corner points in a triangle and then displays the triangle’s angles.
Character Data Type

```java
char letter = 'A'; (ASCII)
char numChar = '4'; (ASCII)
char letter = 'u0041'; (Unicode)
char numChar = 'u0034'; (Unicode)
```

Note: The increment and decrement operators can also be used on char variables to get the next or preceding Unicode character. For example, the following statements display character b.

```java
char ch = 'a';
System.out.println(++ch);
```

Unicode Format

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’. So, Unicode can represent 65535 + 1 characters.

```java
Unicode \u03b1 \u03b2 \u03b3 for three Greek letters
```

ASCII Code for Commonly Used Characters

<table>
<thead>
<tr>
<th>Characters</th>
<th>Code Value in Decimal</th>
<th>Unicode Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>'0' to '9'</td>
<td>48 to 57</td>
<td>\u0030 to \u0039</td>
</tr>
<tr>
<td>'A' to 'Z'</td>
<td>65 to 90</td>
<td>\u0041 to \u005A</td>
</tr>
<tr>
<td>'a' to 'z'</td>
<td>97 to 122</td>
<td>\u0061 to \u007A</td>
</tr>
</tbody>
</table>

Escape Sequences for Special Characters

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Name</th>
<th>Unicode Code</th>
<th>Decimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>Backspace</td>
<td>\u0008</td>
<td>8</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
<td>\u0009</td>
<td>9</td>
</tr>
<tr>
<td>\n</td>
<td>Linefeed</td>
<td>\u000A</td>
<td>10</td>
</tr>
<tr>
<td>\f</td>
<td>Formfeed</td>
<td>\u000C</td>
<td>12</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage Return</td>
<td>\u000D</td>
<td>13</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
<td>\u005C</td>
<td>92</td>
</tr>
<tr>
<td>&quot;</td>
<td>Double Quote</td>
<td>\u0022</td>
<td>34</td>
</tr>
</tbody>
</table>
Appendix B: ASCII Character Set

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

<table>
<thead>
<tr>
<th>\u0000</th>
<th>\u0001</th>
<th>\u0002</th>
<th>\u0003</th>
<th>\u0004</th>
<th>\u0005</th>
<th>\u0006</th>
<th>\u0007</th>
</tr>
</thead>
<tbody>
<tr>
<td>\u0008</td>
<td>\u0009</td>
<td>\u000a</td>
<td>\u000b</td>
<td>\u000c</td>
<td>\u000d</td>
<td>\u000e</td>
<td>\u000f</td>
</tr>
<tr>
<td>\u0010</td>
<td>\u0011</td>
<td>\u0012</td>
<td>\u0013</td>
<td>\u0014</td>
<td>\u0015</td>
<td>\u0016</td>
<td>\u0017</td>
</tr>
<tr>
<td>\u0018</td>
<td>\u0019</td>
<td>\u001a</td>
<td>\u001b</td>
<td>\u001c</td>
<td>\u001d</td>
<td>\u001e</td>
<td>\u001f</td>
</tr>
</tbody>
</table>

ASCII Character Set, cont.

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

<table>
<thead>
<tr>
<th>\u0080</th>
<th>\u0081</th>
<th>\u0082</th>
<th>\u0083</th>
<th>\u0084</th>
<th>\u0085</th>
<th>\u0086</th>
<th>\u0087</th>
</tr>
</thead>
<tbody>
<tr>
<td>\u0088</td>
<td>\u0089</td>
<td>\u008a</td>
<td>\u008b</td>
<td>\u008c</td>
<td>\u008d</td>
<td>\u008e</td>
<td>\u008f</td>
</tr>
<tr>
<td>\u0090</td>
<td>\u0091</td>
<td>\u0092</td>
<td>\u0093</td>
<td>\u0094</td>
<td>\u0095</td>
<td>\u0096</td>
<td>\u0097</td>
</tr>
<tr>
<td>\u0098</td>
<td>\u0099</td>
<td>\u009a</td>
<td>\u009b</td>
<td>\u009c</td>
<td>\u009d</td>
<td>\u009e</td>
<td>\u009f</td>
</tr>
</tbody>
</table>

Casting between char and Numeric Types

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

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

Comparing and Testing Characters

```java
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");
```
### Methods in the Character Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isDigit(ch)</td>
<td>Returns true if the specified character is a digit.</td>
</tr>
<tr>
<td>isLetter(ch)</td>
<td>Returns true if the specified character is a letter.</td>
</tr>
<tr>
<td>isLetterOrDigit(ch)</td>
<td>Returns true if the specified character is a letter or digit.</td>
</tr>
<tr>
<td>isLowerCase(ch)</td>
<td>Returns true if the specified character is a lowercase letter.</td>
</tr>
<tr>
<td>isUpperCase(ch)</td>
<td>Returns true if the specified character is an uppercase letter.</td>
</tr>
<tr>
<td>toLowerCase(ch)</td>
<td>Returns the lowercase of the specified character.</td>
</tr>
<tr>
<td>toUpperCase(ch)</td>
<td>Returns the uppercase of the specified character.</td>
</tr>
</tbody>
</table>

### The String Type

The char type only represents one character. To represent a string of characters, use the data type called String. For example,

```java
String message = "Welcome to Java";
```

String is actually a predefined class in the Java library just like the System class and Scanner class. 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. Reference data types will be thoroughly discussed in Chapter 9, “Objects and Classes.” For the time being, you just need to know how to declare a String variable, how to assign a string to the variable, how to concatenate strings, and to perform simple operations for strings.

### Simple Methods for String Objects

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length()</td>
<td>Returns the number of characters in this string.</td>
</tr>
<tr>
<td>charAt(index)</td>
<td>Returns the character at the specified index from this string.</td>
</tr>
<tr>
<td>concat(s1)</td>
<td>Returns a new string that concatenates this string with string s1.</td>
</tr>
<tr>
<td>toUpperCase()</td>
<td>Returns a new string with all letters in uppercase.</td>
</tr>
<tr>
<td>toLowerCase()</td>
<td>Returns a new string with all letters in lowercase.</td>
</tr>
<tr>
<td>trim()</td>
<td>Returns a new string with whitespace characters trimmed on both sides.</td>
</tr>
</tbody>
</table>

Strings are objects in Java. The methods in the preceding table can only be invoked from a specific string instance. For this reason, these methods are called instance methods. A non-instance method is called a static method. A static method can be invoked without using an object. All the methods defined in the Math class are static methods. They are not tied to a specific object instance. The syntax to invoke an instance method is

```java
referenceVariable.methodName(arguments).
```
Getting String Length

String message = "Welcome to Java";
System.out.println("The length of " + message + " is " + message.length());

Getting Characters from a String

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

Converting Strings

"Welcome".toLowerCase() returns a new string, welcome.
"Welcome".toUpperCase() returns a new string, WELCOME.
" Welcome ".trim() returns a new string, Welcome.

String Concatenation

String s3 = s1.concat(s2); or String s3 = s1 + s2;
// 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);

Comparing Strings

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String.equals(s1)</td>
<td>Returns true if this string is equal to string s1.</td>
</tr>
<tr>
<td>String.equalsIgnoreCase(s1)</td>
<td>Returns true if this string is equal to string s1; it is case insensitive.</td>
</tr>
<tr>
<td>String.compareTo(s1)</td>
<td>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.</td>
</tr>
<tr>
<td>String.compareToIgnoreCase(s1)</td>
<td>Same as compareTo except that the comparison is case insensitive.</td>
</tr>
<tr>
<td>String.startsWith(prefix)</td>
<td>Returns true if this string starts with the specified prefix.</td>
</tr>
<tr>
<td>String.endsWith(suffix)</td>
<td>Returns true if this string ends with the specified suffix.</td>
</tr>
</tbody>
</table>

Obtaining Substrings

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String.substring(beginIndex)</td>
<td>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 4.2.</td>
</tr>
<tr>
<td>String.substring(beginIndex, endIndex)</td>
<td>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.</td>
</tr>
</tbody>
</table>

OrderTwoCities Run
### Finding a Character or a Substring in a String

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>indexOf(ch)</code></td>
<td>Returns the index of the first occurrence of <code>ch</code> in the string. Returns -1 if not matched.</td>
</tr>
<tr>
<td><code>indexOf(ch, fromIndex)</code></td>
<td>Returns the index of the first occurrence of <code>ch</code> after <code>fromIndex</code> in the string. Returns -1 if not matched.</td>
</tr>
<tr>
<td><code>indexOf(s)</code></td>
<td>Returns the index of the first occurrence of string <code>s</code> in this string. Returns -1 if not matched.</td>
</tr>
<tr>
<td><code>indexOf(s, fromIndex)</code></td>
<td>Returns the index of the first occurrence of string <code>s</code> after <code>fromIndex</code>. Returns -1 if not matched.</td>
</tr>
<tr>
<td><code>lastIndexOf(ch)</code></td>
<td>Returns the index of the last occurrence of <code>ch</code> in the string. Returns -1 if not matched.</td>
</tr>
<tr>
<td><code>lastIndexOf(ch, fromIndex)</code></td>
<td>Returns the index of the last occurrence of <code>ch</code> before <code>fromIndex</code> in this string. Returns -1 if not matched.</td>
</tr>
<tr>
<td><code>lastIndexOf(s)</code></td>
<td>Returns the index of the last occurrence of string <code>s</code> in this string. Returns -1 if not matched.</td>
</tr>
<tr>
<td><code>lastIndexOf(s, fromIndex)</code></td>
<td>Returns the index of the last occurrence of string <code>s</code> before <code>fromIndex</code>. Returns -1 if not matched.</td>
</tr>
</tbody>
</table>

#### Conversion between Strings and Numbers

```java
int intValue = Integer.parseInt(intString);
double doubleValue = Double.parseDouble(doubleString);
String s = number + "";
```

### Finding a Character or a Substring in a String

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

### Problem: Guessing Birthday

The program can guess your birth date. Run to see how it works.
Mathematics Basis for the Game

19 is 10011 in binary. 7 is 111 in binary. 23 is 11101 in binary

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>00110</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10011</td>
<td>00112</td>
</tr>
<tr>
<td>1100</td>
<td>1101</td>
</tr>
</tbody>
</table>

19 + 7 = 00110 + 00112 = 01101

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00001</td>
</tr>
<tr>
<td>2</td>
<td>00010</td>
</tr>
<tr>
<td>3</td>
<td>00011</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>19</td>
<td>10011</td>
</tr>
<tr>
<td>31</td>
<td>1111</td>
</tr>
</tbody>
</table>

Case Study: Converting a Hexadecimal Digit to a Decimal Value

Write a program that converts a hexadecimal digit into a decimal value.

Case Study: Revising the Lottery Program Using Strings

A problem can be solved using many different approaches. This section rewrites the lottery program in Listing 3.7 using strings. Using strings simplifies this program.

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. An item may be a numeric value, character, boolean value, or a string. Each specifier begins with a percent sign.
### Frequently-Used Specifiers

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Output</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%b</td>
<td>a boolean value</td>
<td>true or false</td>
</tr>
<tr>
<td>%c</td>
<td>a character</td>
<td>'a'</td>
</tr>
<tr>
<td>%d</td>
<td>a decimal integer</td>
<td>200</td>
</tr>
<tr>
<td>%f</td>
<td>a floating-point number</td>
<td>45.460000</td>
</tr>
<tr>
<td>%e</td>
<td>a number in standard scientific notation</td>
<td>4.556000e+01</td>
</tr>
<tr>
<td>%s</td>
<td>a string</td>
<td>&quot;Java is cool&quot;</td>
</tr>
</tbody>
</table>

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

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

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

### FormatDemo

The example gives a program that uses `printf` to display a table.