CHAPTER 4
MATHEMATICAL FUNCTIONS, CHARACTERS, STRINGS

ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH INTRODUCTION TO JAVA PROGRAMMING, LIANG (PEARSON 2014)

MATHEMATICAL FUNCTIONS

- Java provides many useful methods in the **`Math`** class for performing common mathematical functions.
- In order to use them we need to understand:
  - What a method is
  - How to use methods
  - Where we look up possible functions to use

METHODS

- **Methods** are subroutines that we would like to (re)use again and again in code
- For example, would you like a method to compute $\sqrt{x}$ or write a lengthy algorithm every time you wish to use it?
- The Java library provides many useful methods. Some we have seen:
  - Scanner
  - If you have read - Math

INTERPRETING METHODS

- Consider the following from the **`Math`** library:

```
double sqrt(double x);
```

- `sqrt` is an *identifier*, i.e., a name, for this method
- `double x` is called a **parameter**, or an **argument**. This is the *input* to the method.
- `double` is the type of data *output* by the method
- In a few weeks, we will learn to write our own methods. For now, we need to know how to use them.

INVOKING A METHOD

- A method is invoked, or called/used, in code:
- `double x = Math.sqrt(2); //invoke/call and save result`
- `double y = Math.sqrt(2)/2; //use inside of an expression`

INVOKING A METHOD

- Difference between **`Math`** and **`Scanner`**
- Methods sometimes depend on the value of an object/class and sometimes they do not. Common math functions, like `sqrt`, do not need to know anything besides the parameter. However, other things like Scanner needs to know what it is scanning, so we invoke methods from a variable instead:
- `Scanner in = new Scanner(System.in); //Make a variable`
- `double x = in.nextDouble(); //Use the variable`

READING AN API

- API or Application Program Interface tells a programmer how to use a class or piece of code
- Look at one and let us interpret
- Java Math API

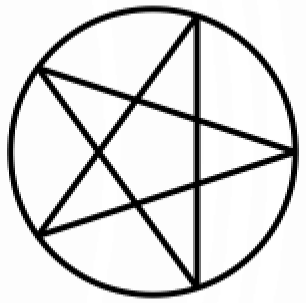
THE MATH CLASS

- Class constants:
  - PI (with **`Math`**`.PI`)
  - E
- Class methods:
  - Trigonometric Methods (examples : **`Math`**`.sin(x)`,**`Math`**`.cos(x)`)
  - Exponent Methods (examples: **`Math`**`.log(x)`,**`Math`**`.pow(x, y)`)
  - Rounding Methods (examples: **`Math`**`.floor(x)`,**`Math`**`.round(x)`)
  - min, max, abs, and random Methods (examples: **`Math`**`.min(x, y)`,**`Math`**`.random()`)

MATH.RANDOM()

- Generates a random double value greater than or equal to 0.0 and less than 1.0 ($0 \leq$ `Math.random()` < 1.0).
  - `a + Math.random() * b` – Returns a random number between a and $a + b$, excluding $a + b$.
  - `(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.

EXERCISE



- With a partner, lets make a program that draws a pentagon or pentagram inscribed in a circle
- Recall the coordinate system in StdDraw is a box from (0, 0) to (1, 1). Change with:
  - `StdDraw.setXscale(-1.5*r, 1.5*r)`
  - `StdDraw.setYscale(-1.5*r, 1.5*r)`
- To draw a circle use: `StdDraw.circle(x, y, r);`
- Recall for a point on a circle at angle θ, $x = r\cos\theta$ and $y = r\sin\theta$

CHARACTER DATA TYPE

- Characters are symbols used predominantly for textual information
- `char letter = 'A'; //ASCII`
- `char numChar = '4'; //ASCII, ASCII is one mapping from binary value to symbols. Found in Appendix B of book`
- `char letter = '\u0041'; //Unicode, mapping that supports other languages characters`
- `char numChar = '\u0034'; //Unicode`
- Literals are denoted with a single quote

ESCAPE SEQUENCES FOR SPECIAL CHARACTERS

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

METHODS IN THE CHARACTER CLASS

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.

THE STRING TYPE

- The char type only represents one character. To represent a message (string of characters), use the data type called **`String`**. For example,

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

- *Note - 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.*

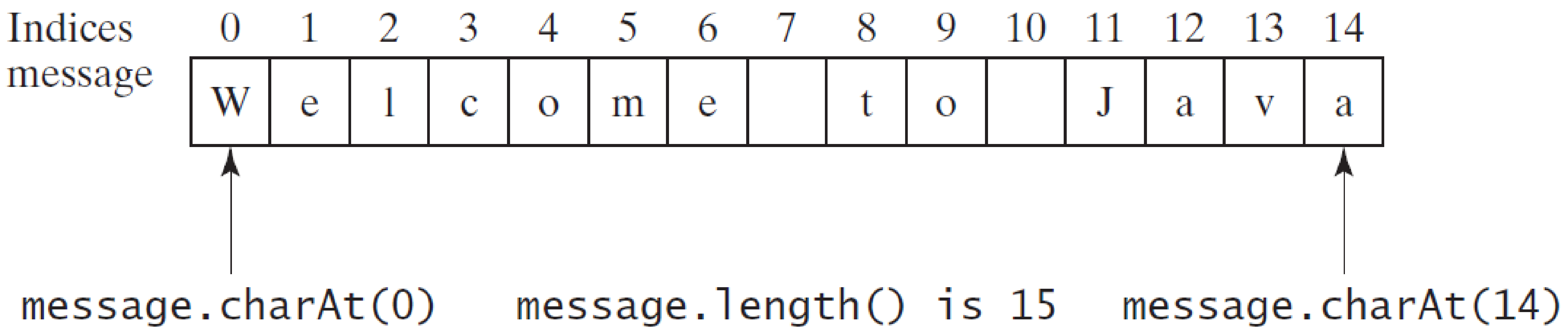
SIMPLE METHODS FOR STRING OBJECTS

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.

GETTING STRING LENGTH

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

GETTING CHARACTERS FROM A STRING



STRING CONCATENATION

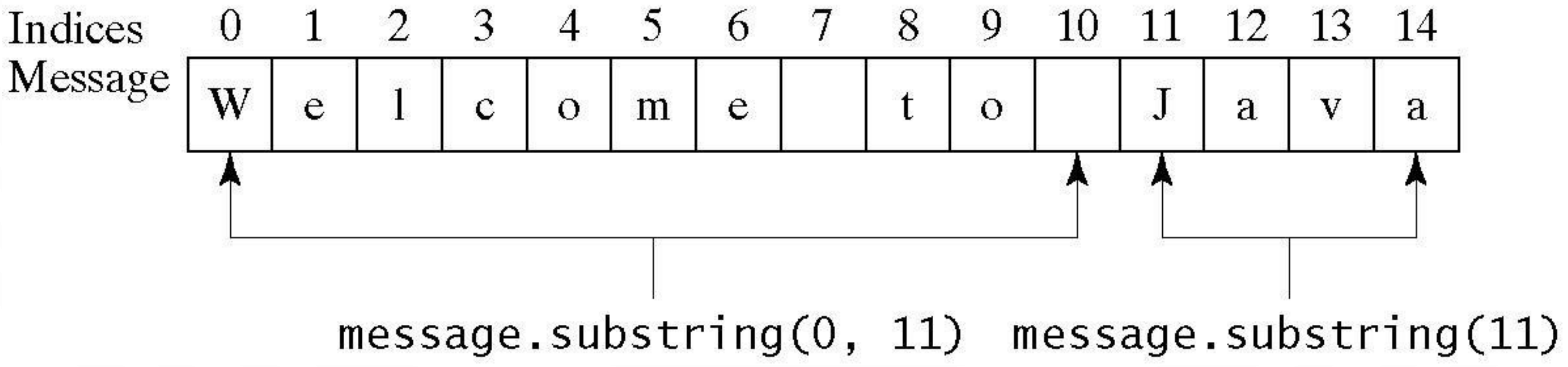
```java
//String s3 = s1.concat(s2); or String s3 = s1 + s2;

// 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

```java
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);
```

OBTAINING 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, as shown in Figure 4.2.
`substring(beginIndex, endIndex)`	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.



CONVERSION BETWEEN STRINGS AND NUMBERS

```
1. //String to a number
2. int intValue = Integer.parseInt(intString);
3. double doubleValue = Double.parseDouble(doubleString);
4. //Number to a string
5. String s = String.valueOf(number);
```