

# Chapter 4

## Mathematical Functions, Characters, and Strings

COSC1046

Ping Luo

# Mathematical Functions

- Java provides many useful methods in the **Math** class for performing common mathematical functions.
- e.g. `Math.random()`

# The Math Class

- Constants:

- PI

```
jshell> Math.PI  
$1 ==> 3.141592653589793
```

- E

```
jshell> Math.E  
$2 ==> 2.718281828459045
```

- Methods:

- Trigonometric Methods
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods

# Trigonometric Methods

**TABLE 4.1** Trigonometric Methods in the Math Class

<i>Method</i>	<i>Description</i>
<code>sin(radians)</code>	Returns the trigonometric sine of an angle in radians.
<code>cos(radians)</code>	Returns the trigonometric cosine of an angle in radians.
<code>tan(radians)</code>	Returns the trigonometric tangent of an angle in radians.
<code>toRadians(degree)</code>	Returns the angle in radians for the angle in degrees.
<code>toDegrees(radians)</code>	Returns the angle in degrees for the angle in radians.
<code>asin(a)</code>	Returns the angle in radians for the inverse of sine.
<code>acos(a)</code>	Returns the angle in radians for the inverse of cosine.
<code>atan(a)</code>	Returns the angle in radians for the inverse of tangent.

`Math.sin(Math.toRadians(270))` returns **-1.0**

`Math.sin(Math.PI / 6)` returns **0.5**

`Math.sin(Math.PI / 2)` returns **1.0**

`Math.cos(0)` returns **1.0**

# Exponent Methods

**TABLE 4.2** Exponent Methods in the Math Class

<i>Method</i>	<i>Description</i>
<code>exp(x)</code>	Returns e raised to power of x ( $e^x$ ).
<code>log(x)</code>	Returns the natural logarithm of x ( $\ln(x) = \log_e(x)$ ).
<code>log10(x)</code>	Returns the base 10 logarithm of x ( $\log_{10}(x)$ ).
<code>pow(a, b)</code>	Returns a raised to the power of b ( $a^b$ ).
<code>sqrt(x)</code>	Returns the square root of x ( $\sqrt{x}$ ) for $x \geq 0$ .

# Rounding Methods

**TABLE 4.3** Rounding Methods in the Math Class

<i>Method</i>	<i>Description</i>
<code>ceil(x)</code>	$x$ is rounded up to its nearest integer. This integer is returned as a double value.
<code>floor(x)</code>	$x$ is rounded down to its nearest integer. This integer is returned as a double value.
<code>rint(x)</code>	$x$ is rounded to its nearest integer. If $x$ is equally close to two integers, the even one is returned as a double value.
<code>round(x)</code>	Returns <code>(int)Math.floor(x + 0.5)</code> if $x$ is a float and returns <code>(long)Math.floor(x + 0.5)</code> if $x$ is a double.

```
jshell> Math.rint(3.5)
$6 ==> 4.0

jshell> Math.round(3.5)
$7 ==> 4
```


# 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)**.


# random

- Generates a random double value greater than or equal to 0.0 and less than 1.0 ( $0 \leq \text{Math.random()} < 1.0$ ).

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.

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



# Assignment 1 – Q3

*(Random month)* Write a program that randomly generates an integer between 1 and 12 and displays the English month names January, February, . . . , December for the numbers 1, 2, . . . , 12, accordingly.

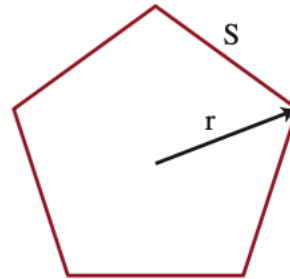
`a + Math.random() * b`  $\longrightarrow$  Returns a random number between a and a + b, excluding a + b.

```
int number = (int)(Math.random() * 12) + 1;
```

```
int number = (int)(System.currentTimeMillis() % 12 + 1);
```

# Exercise

- 4.1** (*Geometry: area of a pentagon*) Write a program that prompts the user to enter the length from the center of a pentagon to a vertex and computes the area of the pentagon, as shown in the following figure.

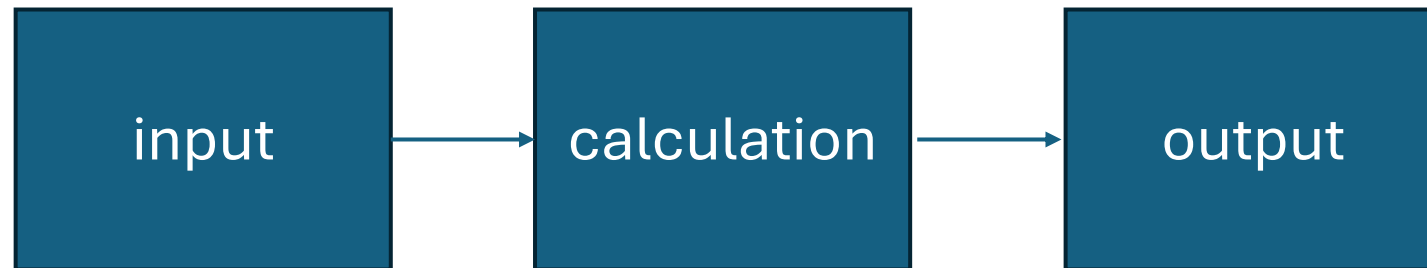


The formula for computing the area of a pentagon is  $Area = \frac{5 \times s^2}{4 \times \tan\left(\frac{\pi}{5}\right)}$ , where

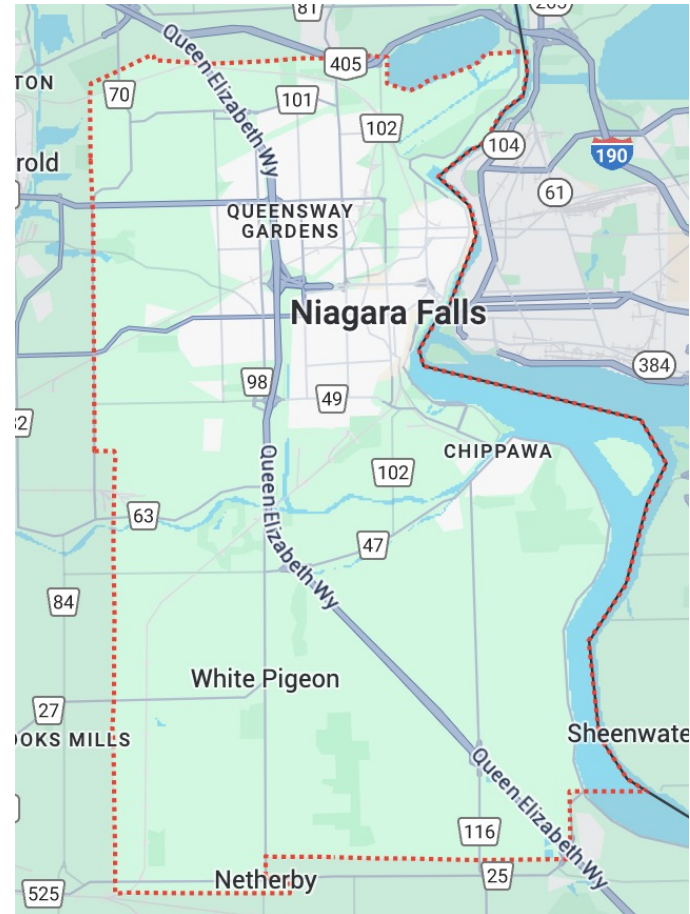
$s$  is the length of a side. The side can be computed using the formula  $s = 2r \sin\frac{\pi}{5}$ , where  $r$  is the length from the center of a pentagon to a vertex. Round up two digits after the decimal point. Here is a sample run:

# Exercise

how to start the coding?



# Area of City



# Character Data Type

- In addition to processing numeric values, we can process **characters** in Java. The character data type, **char**, is used to represent a **single character**. A character literal is enclosed in **single quotation marks**.
- **char** letter = **'A'**;
- **char** numChar = **'4'**;

# Unicode and ASCII Code

- Computers use binary numbers internally. A character is stored in a computer as a sequence of 0s and 1s. Mapping a character to its **binary representation** is called *encoding*. There are different ways to encode a character.
- Java supports **Unicode**, an encoding scheme established by the **Unicode Consortium** to support the interchange, processing, and display of written texts in the world's diverse languages. Unicode was originally designed as a **16-bit** character encoding.
- A **16-bit** Unicode takes two bytes, preceded by `\u`, expressed in four **hexadecimal digits** that run from `\u0000` to `\uFFFF`.

# Hexadecimal Numbering System

- The first **ten** digits are the same as in the decimal system (0 to 9), and the next **six** digits are represented by the letters A to F (or a to f), where:
  - A represents the decimal value 10
  - B represents the decimal value 11
  - C represents the decimal value 12
  - D represents the decimal value 13
  - E represents the decimal value 14
  - F represents the decimal value 15

# Unicode and ASCII Code

- Most computers use ASCII (American Standard Code for Information Interchange), an **8-bit encoding scheme**, for representing all uppercase and lowercase letters, digits, punctuation marks, and control characters.
- Unicode includes ASCII code, with `\u0000` to `\u007F` corresponding to the 128 ASCII characters.



# ASCII Code of Commonly Used Characters

<i>Characters</i>	<i>Code Value in Decimal</i>	<i>Unicode Value</i>
'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

# Unicode and ASCII Code

- You can use ASCII characters such as 'X', '1', and '\$' in a Java program as well as Unicodes. Thus, for example, the following statements are equivalent:
- `char letter = 'A';`  
`char letter = '\u0041'; // Character A's Unicode is 0041`

# NOTE

<i>Characters</i>	<i>Code Value in Decimal</i>	<i>Unicode Value</i>
'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

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.

```
char ch = 'a';
```

```
System.out.println(++ch);
```

# Note

System.out.println("He said "Java is fun"); ----->???

**TABLE 4.5** Escape Sequences

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

Jack said “Java is fun.”

Only use Unicode when ASCII doesn't have the character.

System.out.println("He said \"Java is fun\");

System.out.println("He said '\u0022 Java is fun\");

# Comparing 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");`
- `'a' < 'b'` is true because the Unicode for `'a'` (97) is less than the Unicode for `'b'` (98).
- `'a' < 'A'` is false because the Unicode for `'a'` (97) is greater than the Unicode for `'A'` (65).
- `'1' < '8'` is true because the Unicode for `'1'` (49) is less than the Unicode for `'8'` (56).

# Methods in the Character Class

**TABLE 4.6** Methods in the Character Class

<i>Method</i>	<i>Description</i>
<code>isDigit(ch)</code>	Returns true if the specified character is a digit.
<code>isLetter(ch)</code>	Returns true if the specified character is a letter.
<code>isLetterOrDigit(ch)</code>	Returns true if the specified character is a letter or digit.
<code>isLowerCase(ch)</code>	Returns true if the specified character is a lowercase letter.
<code>isUpperCase(ch)</code>	Returns true if the specified character is an uppercase letter.
<code>toLowerCase(ch)</code>	Returns the lowercase of the specified character.
<code>toUpperCase(ch)</code>	Returns the uppercase of the specified character.

# String

- The char type only represents **one** character. To represent a string of characters, use the data type called String. For example,
- **String message = “Hello World”;**
- 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.

# Caution

- A string literal must be enclosed in double quotation marks (" "). A character literal is a single character enclosed in single quotation marks (' ').
- Therefore, "A" is a string, but 'A' is a character.



# Methods for String Objects

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

# Converting String

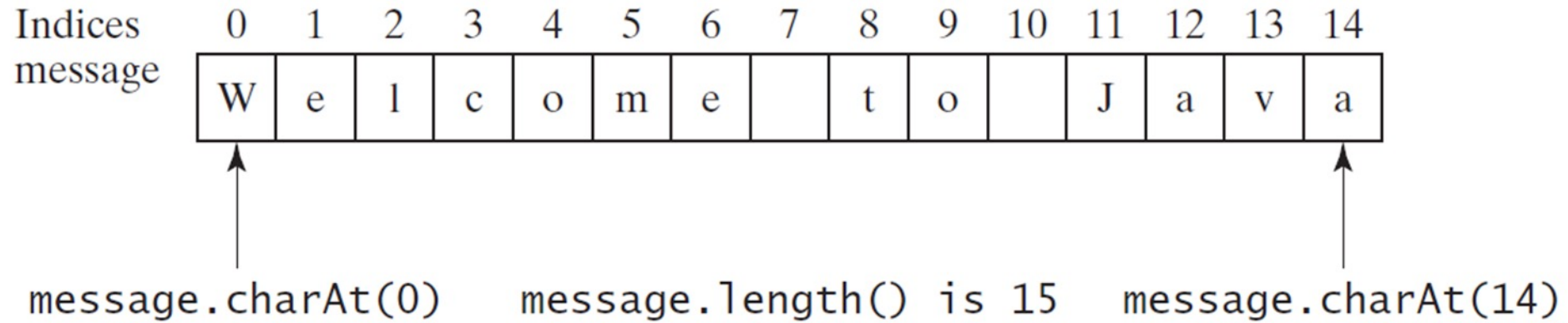
- "Welcome".toLowerCase() returns a new string, welcome.
- "Welcome".toUpperCase() returns a new string, WELCOME.
- The trim() method returns a new string by eliminating **whitespace characters** from both ends of the string. The characters ' ', \t, \f, \r, or \n are known as whitespace characters.
- "\t **Good Night** \n".trim() returns a new string **Good Night**.

# String Length

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

```
System.out.println("The length of “ + message + " is ” + message.length());
```

# Characters from String



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

# Methods for String Objects

- 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
- `referenceVariable.methodName(arguments)`.
- **`message.length()` vs `Math.pow()`**

# String Concatenation

- You can use the concat method to concatenate two strings. The statement given below, for example, concatenates strings s1 and s2 into s3:
  - `String s3 = s1.concat(s2);` or `String s3 = s1 + s2;`
  - `String message = "Welcome " + "to " + "Java";`
  - `String s = "Chapter" + 2; // s becomes Chapter2`
  - `String s1 = "Supplement" + 'B'; // s1 becomes SupplementB`

# Input Character using Scanner

- The next() method reads a string that ends with a **whitespace** character. You can use the **nextLine()** method to read an **entire line** of text. The nextLine() method reads a string that ends with the Enter key pressed.

```
jshell> String s = sc.next()
javais good
s ==> "javais"

jshell> s
s ==> "javais"

jshell> String s = sc.nextLine()
javais good
s ==> " javais good"
```

# Input String and Character

- `char a = sc.next().charAt(0);`
- `String b = sc.next();`
- `String c = sc.nextLine();`



# Case Study: Converting a Hexadecimal Digit to a Decimal Value

- Write a program that converts a hexadecimal digit into a decimal value.
  - The hexadecimal number system has 16 digits: 0–9, A–F.
  - The letters A, B, C, D, E, and F correspond to the decimal numbers 10, 11, 12, 13, 14, and 15

# 10 min Break

# Comparing Strings

**TABLE 4.8** Comparison Methods for `String` Objects

<i>Method</i>	<i>Description</i>
<code>equals(s1)</code>	Returns true if this string is equal to string <code>s1</code> .
<code>equalsIgnoreCase(s1)</code>	Returns true if this string is equal to string <code>s1</code> ; it is case insensitive.
<code>compareTo(s1)</code>	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 <code>s1</code> .
<code>compareToIgnoreCase(s1)</code>	Same as <code>compareTo</code> except that the comparison is case insensitive.
<code>startsWith(prefix)</code>	Returns true if this string starts with the specified prefix.
<code>endsWith(suffix)</code>	Returns true if this string ends with the specified suffix.
<code>contains(s1)</code>	Returns true if <code>s1</code> is a substring in this string.

# Comparing Strings

```
if (string1 == string2)
    System.out.println("string1 and string2 are the same object");
else
    System.out.println("string1 and string2 are different objects");

if (string1.equals(string2))
    System.out.println("string1 and string2 have the same contents");
else
    System.out.println("string1 and string2 are not equal");
```

# Problem: OrderTwoCities

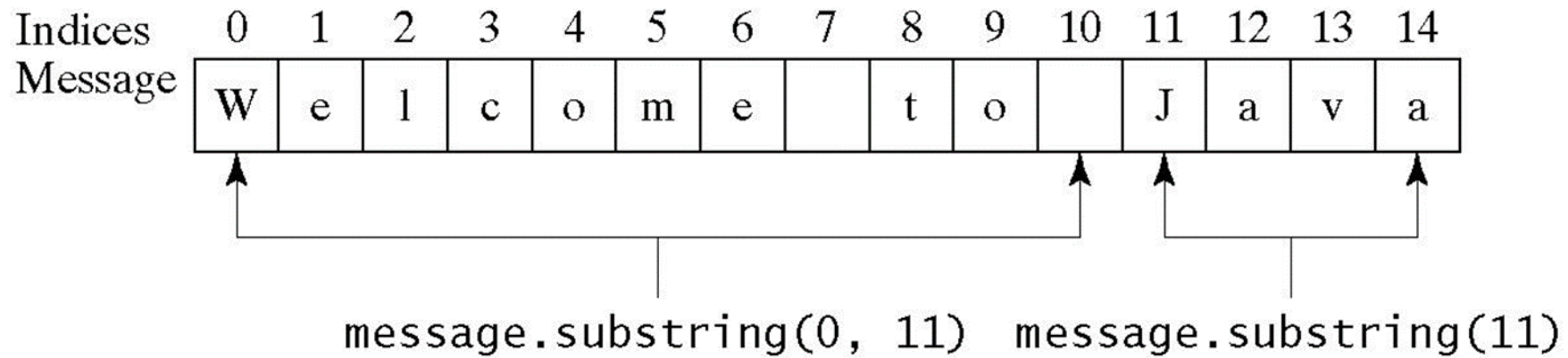
- Ask user to input two cities, compare them and output them in alphabetical order.

```
import java.util.Scanner;

public class OrderTwoCities {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // Enter two cities
        System.out.print(s:"Enter the first city: ");
        String city1 = input.nextLine();
        System.out.print(s:"Enter the second city: ");
        String city2 = input.nextLine();
        if ( )
            System.out.println("The cities in alphabetical order are " +
                city1 + " " + city2);
        else
            System.out.println("The cities in alphabetical order are " + city2 + " " + city1);
    }
}
```

# Substring

You can obtain a substring from a string using the **substring** method in the **String** class.



# Find Character or Substring

- The **String** class provides several versions of **indexOf** and **lastIndexOf** methods to find a character or a substring in a string, as listed in Table 4.10.

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

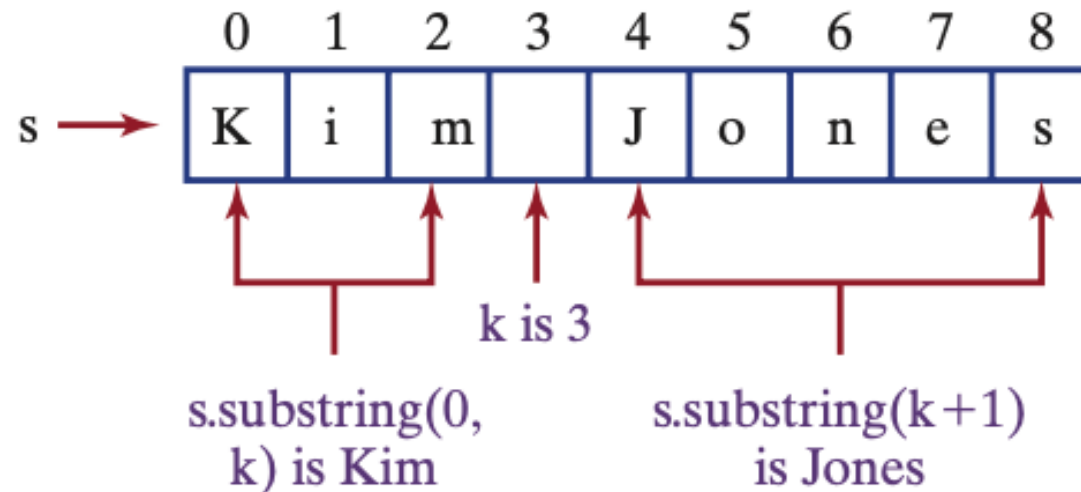
# Find Character or Substring

A system has information of users' name consists of the first name and last name separated by a space.

```
int k = s.indexOf(' ');
```

```
String firstName = s.substring(0, k);
```

```
String lastName = s.substring(k + 1);
```





# Note

- Strings are fundamental in programming. The ability to write programs using strings is essential in learning (Java) programming.
- e.g. LLM models take text as input.

# Case Study: the Lottery Program

- Generates a random two-digit number, prompts the user to enter a two-digit number, and determines whether the user wins according to the following rule:
  1. If the user input matches the lottery number in the exact order, the award is \$10,000.
  2. If all the digits in the user input match all the digits in the lottery number, the award is \$3,000.
  3. If one digit in the user input matches a digit in the lottery number, the award is \$1,000.

```
Enter your lottery pick (two digits): 00   
The lottery number is 00  
Exact match: you win $10,000
```

```

public class LotteryUsingStrings {
    Run | Debug
    public static void main(String[] args) {
        lottery = "" + (int)(Math.random() * 10) + (int)(Math.random() * 10);

        // Get digits from lottery
        char lotteryDigit1 = lottery.charAt(0);
        char lotteryDigit2 = lottery.charAt(1);

        // PEnter a guess
        Scanner input = new Scanner(System.in);
        System.out.print(s:"Enter your lottery pick (two digits): ");
        String guess = input.nextLine();

        // Get digits from guess
        char guessDigit1 = guess.charAt(index:0);
        char guessDigit2 = guess.charAt(index:1);
        System.out.println("The lottery number is " + lottery);
        // Check the guess
        if (_____)
            System.out.println(x:"Exact match: you win $10,000");
        else if (_____ && _____)
            System.out.println(x:"Match all digits: you win $3,000");
        else if (_____)
            System.out.println(x:"Match one digit: you win $1,000");
        else
            System.out.println(x:"Sorry, no match");
    }
}

```

# Format Output

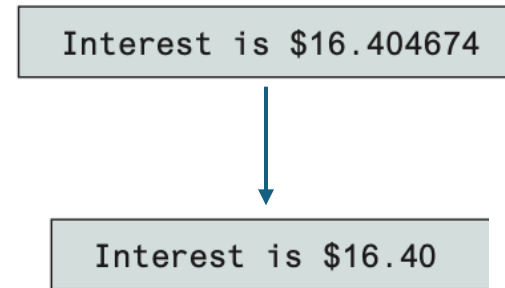
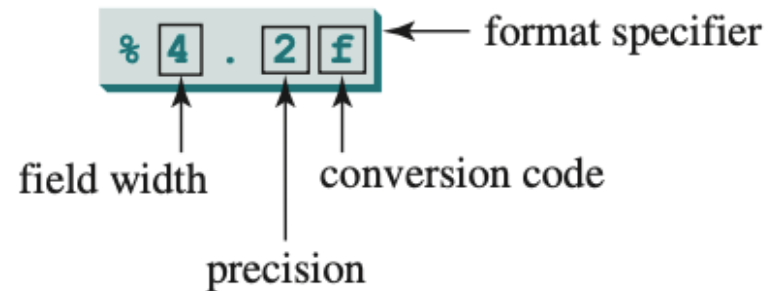
```
double amount = 12618.98;  
double interestRate = 0.0013;  
double interest = amount * interestRate;  
System.out.println("Interest is $" + interest);
```

```
Interest is $16.404674
```

- Use the `System.out.printf()` statement to display formatted output on the console

# Format Output

- `System.out.printf(format, items);`
- `System.out.printf("Interest is $%4.2f", interest);`



# Format Output

**TABLE 4.11** Frequently Used Format Specifiers

<i>Format Specifier</i>	<i>Output</i>	<i>Example</i>
<b>%b</b>	A Boolean value	True or false
<b>%c</b>	A character	'a'
<b>%d</b>	A decimal integer	200
<b>%f</b>	A floating-point number	45.460000
<b>%e</b>	A number in standard scientific notation	4.556000e+01
<b>%s</b>	A string	"Java is cool"

Items must match the format specifiers in order, in number, and in exact type. For example, the format specifier for count is %d and for amount is %f.

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

display

count is 5 and amount is 45.560000

default: 6 digits

# Exercise 01

**\*4.15** (*Phone key pads*) The international standard letter/number mapping found on the telephone is shown below:



Write a program that prompts the user to enter a lowercase or uppercase letter and displays its corresponding number. For a nonletter input, display invalid input.

Enter a letter: A   
The corresponding number is 2



Enter a letter: a   
The corresponding number is 2

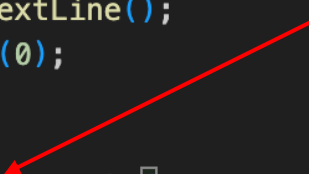


```
import java.util.Scanner;

public class Exercise04_15 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a letter: ");
        String s = input.nextLine();
        char ch = s.charAt(0);

        int number = 0;
        switch (_____) {
            case: _____
                ...
            default: System.out.println(ch + "is an invalid input ");
                    System.exit(1);
        }

        System.out.println("The corresponding number is " + number);
    }
}
```





# Exercise 02

- \*4.17** (*Days of a month*) Write a program that prompts the user to enter the year and the first three letters of a month name (with the first letter in uppercase) and displays the number of days in the month. If the input for month is incorrect, display a message as presented in the following sample runs:

```
Enter a year: 2001   
Enter a month: Jan   
Jan 2001 has 31 days
```



# Exercise 03

**4.22** (*Check substring*) Write a program that prompts the user to enter two strings, and reports whether the second string is a substring of the first string.

```
Enter string s1: ABCD ↵ Enter
Enter string s2: BC ↵ Enter
BC is a substring of ABCD
```

```
import java.util.Scanner;

public class Exercise04_22 {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print(s:"Enter string s1: ");
        String s1 = input.nextLine();

        System.out.print(s:"Enter string s2: ");
        String s2 = input.nextLine();

        if (_____ ) {
            System.out.println(s2 + " is a substring of " + s1);
        }
        else {
            System.out.println(s2 + " is not a substring of " + s1);
        }
    }
}
```

# Exercise 04

**\*4.25** (*Generate vehicle plate numbers*) Assume that a vehicle plate number consists of three uppercase letters followed by four digits. Write a program to generate a plate number.

```
public class Exercise04_25 {  
    Run | Debug  
    public static void main(String[] args) {  
        // how to generate:  
  
        String vehiclePlateNumber = "" + ch1 + ch2 + ch3 + ch4 + ch5 + ch6 + ch7;  
  
        System.out.println("A random vehicle plate number: "  
            + vehiclePlateNumber);  
    }  
}
```

# Q&A