Chapter 4
Mathematical Functions, Characters, and Strings

4.1 Introduction

• The focus of this chapter is to introduce mathematical functions, characters, and strings, and use them to develop programs.

4.2 Common Mathematical Functions

• Java provides many useful methods in the Math class for performing common mathematical functions.

• The Math Class
  o Class constants:
    ▪ PI
    ▪ E (the base of natural logarithms)
  o Class methods:
    ▪ Trigonometric Methods
    ▪ Exponent Methods
    ▪ Rounding Methods
    ▪ min, max, abs, and random Methods
    ▪ random Method

4.2.1 Trigonometric Methods

• Trigonometric Methods:
  o sin(double a)
  o cos(double a)
  o tan(double a)
  o acos(double a)
  o asin(double a)
  o atan(double a)

• 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
4.2.2 Exponent Methods

- 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

4.2.3 The Rounding Methods

- 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.
  - double rint(double x)
    x is rounded to its nearest integer. If x is equally close to two integers, the even one is returned as a double.
  - int round(float x)
    Return (int)Math.floor(x+0.5).
  - long round(double x)
    Return (long)Math.floor(x+0.5).

- 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

4.2.4 The min, max, and abs Methods

- Methods:
  - max(a, b) and min(a, b)
    Returns the maximum or minimum of two parameters.
  - abs(a)
    Returns the absolute value of the parameter.

- Examples:
  Math.max(2, 3) returns 3
  Math.max(2.5, 3) returns 3.0
  Math.min(2.5, 3.6) returns 2.5
  Math.abs(-2) returns 2
  Math.abs(-2.1) returns 2.1

4.2.5 The random Method

- Methods:
  - random()
    Returns a random double value in the range [0.0, 1.0).
    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.
4.2.6 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.

```
import java.util.Scanner;

public class ComputeAngles {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter three points
        System.out.print("Enter three points: ");
        double x1 = input.nextDouble();
        double y1 = input.nextDouble();
        double x2 = input.nextDouble();
        double y2 = input.nextDouble();
        double x3 = input.nextDouble();
        double y3 = input.nextDouble();

        // Compute three sides
        double a = Math.sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3));
        double b = Math.sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 - y3));
        double c = Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));

        // Compute three angles
        double A = Math.toDegrees(Math.acos((a * a - b * b - c * c) / (-2 * b * c)));
        double B = Math.toDegrees(Math.acos((b * b - a * a - c * c) / (-2 * a * c)));
        double C = Math.toDegrees(Math.acos((c * c - b * b - a * a) / (-2 * a * b)));

        // Display results
        System.out.println("The three angles are 
" + Math.round(A * 100) / 100.0 + " " +
                                Math.round(B * 100) / 100.0 + " " +
                                Math.round(C * 100) / 100.0);
    }
}
```

Enter three points: 1 1 6.5 1 6.5 2.5
The three angles are 15.26 90.0 74.74
4.3 Character Data Type and Operations

- The character data type, char, is used to represent a single character.
- A character literal is enclosed in **single** quotation marks.

  ```java
  char letter = 'A';  // Assigns A to char variable letter (ASCII)
  char numChar = '4';  // Assigns numeric character 4 to numChar (ASCII)
  ```

- **Caution:** A string literal must be enclosed in quotation marks. A character literal is a single character enclosed in single quotation marks. So “A” is a string, and ‘A’ is a character.

4.3.1 Unicode and ASCII code

- Java uses 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 (See the Unicode Web site at [www.unicode.org](http://www.unicode.org) for more information.)
- Unicode takes **two** bytes, preceded by \u, expressed in four hexadecimal digits that run from ‘\u0000’ to ‘\uFFFF’. For example, the “coffee” is translated into Chinese using two characters. The Unicode of these two characters are “\u5496\u5561”.

  ```java
  char letter = 'A';
  char letter = '\u0041';  // Character A's Unicode is 0041
  ```

- Unicode can represent 65,536 characters, since FFFF in hexadecimal is 65535.
- Most computer ASCII (American Standard Code for Information Interchange), a **7-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 **128** ASCII characters. (See Appendix B).

  **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>

- **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);
  ```
Appendix B: ASCII Character Set

### 4.3.2 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>

- Suppose you want to print the **quoted** message show below:

  He said “Java is fun”

Here is how to write the statement:

  System.out.println(“He said \"Java is fun\"”);
4.3.3 Casting between char and Numeric Types

- A char can be cast into any numeric type, and vice versa.
- Implicit casting can be used if the result of a casting fits into the target variable. Otherwise explicit casting must be used.
- All numeric operation can be applied to the char operands.
- The char operand is cast into a number if the other operand is a number or a character.
- If the other operand is a string, the character is concatenated with the string.

```java
int i = 'a'; // Same as int i = (int)'a'; // (int) a is 97
char c = 99; // Same as char c = (char)99;

int i = '1' + '2'; // (int) 1 is 49 and (int) 2 is 50
System.out.println("i is " + i);

int j = 1 + 'a'; // (int) a is 97
System.out.println("j is " + 98);
System.out.println(j + " is the Unicode for character " + (char) j);
System.out.println("Chapter " + 2);
```

Output is:
i is 99
j is 98
98 is the Unicode for character b
Chapter 2

4.3.4 Comparing and Testing Characters

- The following code tests whether a character ch is an uppercase letter, a lowercase letter, or a digital character.

```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");
```

<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>
4.4 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, “Classes and Objects.” 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.

### TABLE 4.7 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>

4.4.1 Getting String Length

- You can use the length() method to return the number of characters in a string. For example,

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

  Display:
  The length of Welcome to Java is 15
### 4.4.2 Getting Characters from a String

- The `s.charAt(index)` method can be used to retrieve a specific character in a string `s`, where the index is between 0 and `s.length()`–1.
- For example, `message.charAt(0)` returns the character `W`, as shown in Figure 4.1. Note that the index for the first character in the string is 0.

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

Display:
The first character in message is W

![Figure 4.1](image)

**FIGURE 4.1** The characters in a String object can be accessed using its index.

### 4.4.3 Concatenating Strings

- The plus sign (+) is the **concatenation** operator if one of the operands is a string.
- If one of the operands is a non-string (e.g. a number), the non-string value is **converted** into a string and concatenated with the other string.

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

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

// 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'; // s becomes SupplementB
```

```java
i = 1; j = 3;
System.out.println("i + j is " + i + j);  \(\Rightarrow i + j \text{ is } 13\)
System.out.println("i + j is " + (i + j)); \(\Rightarrow i + j \text{ is } 4\)
```
4.4.4 Converting Strings

- The toLowerCase() method returns a new string with all lowercase letters and the toUpperCase() method returns a new string with all uppercase letters. For example,

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

4.4.5 Reading a String from the Console

- To read a string from the console, invoke the next() method on a Scanner object. For example, the following code reads three strings from the keyboard:

```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);
```

```
Enter three words separated by spaces: Welcome to Java
s1 is Welcome
s2 is to
s3 is Java
```

- 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. For example, the following statements read a line of text.

```java
Scanner input = new Scanner(System.in);
System.out.println("Enter a line: ");
String s = input.nextLine();
System.out.println("The line entered is " + s);
```

```
Enter a line: Welcome to Java
The line entered is Welcome to Java
```

4.4.6 Reading a Character from the Console

- To read a character from the console, use the nextLine() method to read a string and then invoke the charAt(0) method on the string to return a character. For example, the following code reads a character from the keyboard:

```java
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);
```
4.4.7 Comparing Strings

- The String class contains the methods as shown in Table 4.8 for comparing two strings.

**TABLE 4.8 Comparison Methods for String Objects**

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

- The `==` operator checks only whether string1 and string2 refer to the **same object**; it does **not** tell you whether they have the **same contents**.

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

- The `equals` method is used to compare two strings:

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

- For example, the following statements display `true` and then `false`.

```java
String s1 = "Welcome to Java";
String s2 = "Welcome to Java";
String s3 = "Welcome to C++";
System.out.println(s1.equals(s2)); // true
System.out.println(s1.equals(s3)); // false
```

- The `compareTo` method can also be used to compare two strings. For example,

```java
s1.compareTo(s2)
```

- The method returns the value **0** if s1 is equal to s2, a value less than **0** if s1 is lexicographically less than s2, and a value greater than **0** if s1 is lexicographically greater than s2. For example,

```java
String s1 = "abc";
String s2 = "abg";
s1.compareTo(s2); // returns -4.
```
You can also use `str.startsWith(prefix)` to check whether string `str` starts with a specified prefix, `str.endsWith(suffix)` to check whether string `str` ends with a specified suffix, and `str.contains(s1)` to check whether string `str` contains string `s1`. For example,

"Welcome to Java".startsWith("We") returns true.
"Welcome to Java".startsWith("we") returns false.

"Welcome to Java".endsWith("va") returns true.
"Welcome to Java".endsWith("v") returns false.

"Welcome to Java".contains("to") returns true.
"Welcome to Java".contains("To") returns false.

- **Listing 4.2 OrderTwoCities.java**

```java
import java.util.Scanner;

public class OrderTwoCities {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter two cities
        System.out.print("Enter the first city: ");
        String city1 = input.nextLine();
        System.out.print("Enter the second city: ");
        String city2 = input.nextLine();

        if (city1.compareTo(city2) < 0)
            System.out.println("The cities in alphabetical order are " + city1 + " " + city2);
        else
            System.out.println("The cities in alphabetical order are " + city2 + " " + city1);
    }
}
```

Enter the first city: New York
Enter the second city: Boston
The cities in alphabetical order are Boston New York
### 4.4.8 Obtaining Substrings

- You can obtain a substring from a string using the `substring` method in the String class, as shown in Table 4.9. For example,

```java
String message = "Welcome to Java";
String message = message.substring(0, 11) + "HTML";
// The string message now becomes Welcome to HTML.
```

#### TABLE 4.9 The String class contains the method for obtaining substrings.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>substring(beginIndex)</code></td>
<td>Returns this string’s substring that begins with the character at the specified <code>beginIndex</code> and extends to the end of the string, as shown in Figure 4.2.</td>
</tr>
<tr>
<td><code>substring(beginIndex, endIndex)</code></td>
<td>Returns this string’s substring that begins at the specified <code>beginIndex</code> and extends to the character at index <code>endIndex - 1</code>, as shown in Figure 9.6. Note that the character at <code>endIndex</code> is not part of the substring.</td>
</tr>
</tbody>
</table>

#### FIGURE 4.2 The substring method obtains a substring from a string.
4.4.9 Finding a Character or a Substring in a String

- The String class provides several versions of `indexOf` and `lastIndexOf` methods to find a character or a substring in a string.

**TABLE 4.10** The String class contains the methods for finding substrings.

<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> in this string 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>. 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>

- For example,

```
"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.
```

```
"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.
```

- Suppose a string `s` contains the first name and last name separated by a space. You can use the following code to extract the first name and last name from the string:

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

Converting Strings to Integers
- The input returned from the input dialog box is a string. If you enter a numeric value such as 123, it returns “123”. To obtain the input as a number, you have to convert a string into a number.

- To convert a string into an int value, you can use the static parseInt method in the Integer class as follows:

  ```java
  int intValue = Integer.parseInt(intString);
  // where intString is a numeric string such as “123”.
  ```

Converting Strings to Doubles
- To convert a string into a double value, you can use the static parseDouble method in the Double class as follows:

  ```java
  double doubleValue = Double.parseDouble(doubleString);
  // where doubleString is a numeric string such as “123.45”.
  ```

Converting Numbers to Strings
- To convert a number into a string, simply use the string concatenating operator as follows:

  ```java
  String s = number + "";
  ```
4.5 Case Studies

4.5.1 Case Study: Guessing Birthday

- The program can guess your birth date.

- Listing 4.3 GuessBirthday.java

```java
import java.util.Scanner;

public class GuessBirthday {
    public static void main(String[] args) {
        String set1 = "1 3 5 7
                       9 11 13 15
                       17 19 21 23
                       25 27 29 31";

        String set2 = "2 3 6 7
                       10 11 14 15
                       18 19 22 23
                       26 27 30 31";

        String set3 = "4 5 6 7
                       12 13 14 15
                       20 21 22 23
                       28 29 30 31";

        String set4 = "8 9 10 11
                       12 13 14 15
                       24 25 26 27
                       28 29 30 31";

        String set5 = "16 17 18 19
                       20 21 22 23
                       24 25 26 27
                       28 29 30 31";
    }
}
```
```java
int day = 0;

// Create a Scanner
Scanner input = new Scanner(System.in);

// Prompt the user to answer questions
System.out.println("Is your birthday in Set1?\n");
System.out.println("\nEnter 0 for No and 1 for Yes: ");
int answer = input.nextInt();

if (answer == 1)
    day += 1;

// Prompt the user to answer questions
System.out.println("\nIs your birthday in Set2?\n");
System.out.println("\nEnter 0 for No and 1 for Yes: ");
answer = input.nextInt();

if (answer == 1)
    day += 2;

// Prompt the user to answer questions
System.out.println("\nIs your birthday in Set3?\n");
System.out.println("\nEnter 0 for No and 1 for Yes: ");
answer = input.nextInt();

if (answer == 1)
    day += 4;

// Prompt the user to answer questions
System.out.println("\nIs your birthday in Set4?\n");
System.out.println("\nEnter 0 for No and 1 for Yes: ");
answer = input.nextInt();

if (answer == 1)
    day += 8;

// Prompt the user to answer questions
System.out.println("\nIs your birthday in Set5?\n");
System.out.println("\nEnter 0 for No and 1 for Yes: ");
answer = input.nextInt();

if (answer == 1)
    day += 16;

System.out.println("\nYour birthday is " + day + "!");
}
```
The cities in alphabetical order are Boston, New York.
Is your birthday in Set1?
1  3  5  7
9 11 13 15
17 19 21 23
25 27 29 31
Enter 0 for No and 1 for Yes: 1

Is your birthday in Set2?
2  3  6  7
10 11 14 15
18 19 22 23
26 27 30 31
Enter 0 for No and 1 for Yes: 1

Is your birthday in Set3?
4  5  6  7
12 13 14 15
20 21 22 23
28 29 30 31
Enter 0 for No and 1 for Yes: 0

Is your birthday in Set4?
8  9 10 11
12 13 14 15
24 25 26 27
28 29 30 31
Enter 0 for No and 1 for Yes: 0

Is your birthday in Set5?
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31
Enter 0 for No and 1 for Yes: 1

Your birthday is 19!

- Mathematics Basis for the Game

\[
\begin{array}{ccc}
10000 & 00110 & 10000 \\
10 & 10 & 100 \\
\hline
10011 & 00111 & 11101 \\
19 & 7 & 23 \\
\end{array}
\]
4.5.2 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.

- Listing 4.4 HexDigit2Dec.java

```java
import java.util.Scanner;

public class HexDigit2Dec {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a hex digit: ");
        String hexString = input.nextLine();

        // Check if the hex string has exactly one character
        if (hexString.length() != 1) {
            System.out.println("You must enter exactly one character");
            System.exit(1);
        }

        // Display decimal value for the hex digit
        char ch = Character.toUpperCase(hexString.charAt(0));
        if (ch <= 'F' && ch >= 'A') {
            int value = ch - 'A' + 10;
            System.out.println("The decimal value for hex digit 
                + ch + " is " + value);
        } else if (Character.isDigit(ch)) {
            System.out.println("The decimal value for hex digit 
                + ch + " is " + ch);
        } else {
            System.out.println(ch + " is an invalid input");
        }
    }
}
```

Enter a hex digit: AB7C
You must enter exactly one character

Enter a hex digit: B
The decimal value for hex digit B is 11

Enter a hex digit: 8
The decimal value for hex digit 8 is 8

Enter a hex digit: T
T is an invalid input
4.5.3 Case Study: Revising the Lottery Program Using Strings

- The program in Listing 3.8 uses an integer to store the number. Listing 4.5 gives a new program that generates a random **two-digit string** instead of a number and receives the user input as a string instead of a number.

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.

- Listing 4.5 LotteryUsingStrings.java

```
import java.util.Scanner;

public class LotteryUsingStrings {
    public static void main(String[] args) {
        // Generate a lottery as a two-digit string
        String lottery = "" + (int)(Math.random() * 10) + (int)(Math.random() * 10);

        // Prompt the user to enter a guess
        Scanner input = new Scanner(System.in);
        System.out.print("Enter your lottery pick (two digits): ");
        String guess = input.nextLine();

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

        // Get digits from guess
        int guessDigit1 = guess.charAt(0);
        int guessDigit2 = guess.charAt(1);

        System.out.println("The lottery number is " + lottery);

        // Check the guess
        if (guess.equals(lottery))
            System.out.println("Exact match: you win $10,000");
        else if (guessDigit2 == lotteryDigit1 && guessDigit1 == lotteryDigit2)
            System.out.println("Match all digits: you win $3,000");
        else if (guessDigit1 == lotteryDigit1 || guessDigit1 == lotteryDigit2
            || guessDigit2 == lotteryDigit1 || guessDigit2 == lotteryDigit2)
            System.out.println("Match one digit: you win $1,000");
        else
            System.out.println("Sorry, no match");
    }
}
```
<table>
<thead>
<tr>
<th>Enter your lottery pick (two digits):</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>The lottery number is 00</td>
<td></td>
</tr>
<tr>
<td>Exact match: you win $10,000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enter your lottery pick (two digits):</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>The lottery number is 54</td>
<td></td>
</tr>
<tr>
<td>Match all digits: you win $3,000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enter your lottery pick:</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>The lottery number is 34</td>
<td></td>
</tr>
<tr>
<td>Match one digit: you win $1,000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enter your lottery pick:</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>The lottery number is 14</td>
<td></td>
</tr>
<tr>
<td>Sorry: no match</td>
<td></td>
</tr>
</tbody>
</table>
4.6 Formatting Console Outputs

- Use the \texttt{printf} statement.

\begin{verbatim}
System.out.printf(format, items);
\end{verbatim}

- Where \texttt{format} is a string that may consist of substrings and format specifiers.

- A format \texttt{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.

\begin{table}[h]
\centering
\caption{Frequently Used Format Specifiers}
\begin{tabular}{|l|l|l|}
\hline
\textbf{Specifier} & \textbf{Output} & \textbf{Example} \\
\hline
\%b & a boolean value & true or false \\
\%c & a character & \texttt{'}a\texttt{'} \\
\%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" \\
\hline
\end{tabular}
\end{table}

\begin{verbatim}
int count = 5; 
double amount = 45.56; 
System.out.printf("count is \%d and amount is \%f", count, amount); 
\end{verbatim}

\begin{verbatim}
\hline
display            count is 5 and amount is 45.560000 
\hline
\end{verbatim}

\begin{table}[h]
\centering
\caption{Examples of Specifying Width and Precision}
\begin{tabular}{|l|l|}
\hline
\textbf{Example} & \textbf{Output} \\
\hline
\%5c & Output the character and add four spaces before the character item, because the width is 5. \\
\%6b & Output the Boolean value and add one space before the false value and two spaces before the true value. \\
\%5d & Output the integer item with width at least 5. If the number of digits in the item is 6 5, add spaces before the number. If the number of digits in the item is 7 5, the width is automatically increased. \\
\%10.2f & Output the floating-point item with width at least 10 including a decimal point and two digits after the point. Thus, there are 7 digits allocated before the decimal point. If the number of digits before the decimal point in the item is 6 7, add spaces before the number. If the number of digits before the decimal point in the item is 7 7, the width is automatically increased. \\
\%10.2e & Output the floating-point item with width at least 10 including a decimal point, two digits after the point and the exponent part. If the displayed number in scientific notation has width less than 10, add spaces before the number. \\
\%12s & Output the string with width at least 12 characters. If the string item has fewer than 12 characters, add spaces before the string. If the string item has more than 12 characters, the width is automatically increased. \\
\hline
\end{tabular}
\end{table}
- Listing 4.6 FormatDemo.java

```java
public class FormatDemo {
    public static void main(String[] args) {
        // Display the header of the table
        System.out.printf("%-10s%-10s%-10s%-10s\n", "Degrees", "Radians", "Sine", "Cosine", "Tangent");

        // Display values for 30 degrees
        int degrees = 30;
        double radians = Math.toRadians(degrees);
        System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees, radians, Math.sin(radians), Math.cos(radians), Math.tan(radians));

        // Display values for 60 degrees
        degrees = 60;
        radians = Math.toRadians(degrees);
        System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees, radians, Math.sin(radians), Math.cos(radians), Math.tan(radians));
    }
}
```

<table>
<thead>
<tr>
<th>Degrees</th>
<th>Radians</th>
<th>Sine</th>
<th>Cosine</th>
<th>Tangent</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0.5236</td>
<td>0.5000</td>
<td>0.8660</td>
<td>0.5774</td>
</tr>
<tr>
<td>60</td>
<td>1.0472</td>
<td>0.8660</td>
<td>0.5000</td>
<td>1.7321</td>
</tr>
</tbody>
</table>

- The column names are strings. Each string is displayed using the specifier %s, which left-justifies the string.
- The degrees as an integer and four float values. The integer is displayed using the specifier %d and each float is displayed using the specifier %.4f, which specifies four digits after the decimal point.