# Chapter 4: Mathematical Functions, Characters, and Strings

## 4.1. Introduction

## Objectives

- This chapter introduces mathematical functions, characters, and strings for program development.
- You will learn how to use mathematical functions to perform calculations, how to work with characters and strings, and how to format console output.
- You will also learn how to use the `Math` class to perform mathematical operations, how to work with characters and strings, and how to format console output.
- You will also learn how to use the `Scanner` class to read input from the console.

## 4.2. Common Mathematical Functions

## Trigonometric Methods

| Method | Description |
|---|---|
| `sin(radians)` | Returns the trigonometric sine of an angle in radians. |
| `cos(radians)` | Returns the trigonometric cosine of an angle in radians. |
| `tan(radians)` | Returns the trigonometric tangent of an angle in radians. |
| `toRadians(degree)` | Returns the angle in radians for the angle in degrees. |
| `toDegrees(radians)` | Returns the angle in degrees for the angle in radians. |
| `asin(a)` | Returns the angle in radians for the inverse of sine. |
| `acos(a)` | Returns the angle in radians for the inverse of cosine. |
| `atan(a)` | Returns the angle in radians for the inverse of tangent. |

## Example:

```java
double radians = Math.toRadians(45);
double sinValue = Math.sin(radians);
double cosValue = Math.cos(radians);
double tanValue = Math.tan(radians);
```

## Explanation:

- The `toRadians` method converts an angle in degrees to radians.
- The `sin` , `cos` , and `tan` methods return the sine, cosine, and tangent of an angle in radians.
- The `asin` , `acos` , and `atan` methods return the inverse sine, cosine, and tangent of a value.

## Exponent Methods

| Method | Description |
|---|---|
| `exp(x)` | Returns $e$ raised to power of $x$, $e^x$. |
| `log(x)` | Returns the natural logarithm of $x$, $\ln(x)$. |
| `log10(x)` | Returns the base 10 logarithm of $x$, $\log_{10}(x)$. |
| `pow(a, b)` | Returns $a$ raised to the power of $b$, $a^b$. |
| `sqrt(x)` | Returns the square root of $x$, $\sqrt{x}$. |

## **Example:** Exponent Methods

```
double e = Math.exp(1.0);
double ln = Math.log(e);
double log10 = Math.log10(1000);
double pow = Math.pow(2, 3);
double sqrt = Math.sqrt(25);
```

## The Rounding Methods

| Method | Description |
|---|---|
| `ceil(x)` | `x` is rounded up to its nearest integer (returned as a double value). |
| `floor(x)` | `x` is rounded down to its nearest integer (returned as a double value). |
| `rint(x)` | `x` is rounded to its nearest integer (even one if equally close to two integers). |
| `round(x)` | Returns `(int)Math.floor(x + 0.5)` if `x` is a float, `(long)Math.floor(x + 0.5)` if `x` is a double. |

## **Example:** Rounding Methods

```java
double ceil = Math.ceil(2.1);
double floor = Math.floor(2.9);
double rint = Math.rint(2.5);
double round = Math.round(2.5);
```

## The min, max, and abs Methods

| Method | Description |
|---|---|
| `max(a, b)` | Returns the larger of two values. |
| `min(a, b)` | Returns the smaller of two values. |
| `abs(x)` | Returns the absolute value of (x). |

**Example:** min, max, and abs Methods

```
int max = Math.max(2, 3);
int min = Math.min(2, 3);
double abs = Math.abs(-2.5);
```

### Explanation:

- The `max` and `min` methods return the larger and smaller of two values, respectively.
- The `abs` method returns the absolute value of a number.

## The random Method

- The `random()` method returns a random number between 0.0 and 1.0.

**Example:** Random Method

```java
double random = Math.random();
```

**Note:**

- To generate a random number between 0 and 100, multiply the result by 100.
- To generate a random number between 1 and 100, multiply the result by 100 and add 1.
- To generate a random number between 10 and 20, multiply the result by 10 and add 10.

## 4.3. Character Data Type and Operations

## The char Data Type

- The `char` data type is a 16-bit unsigned integer that represents a Unicode character.
- Unicode is a 16-bit character encoding standard that can represent all characters in most languages, including Khmer, Chinese, Japanese, and Korean.

**Example:** Declaring a Character Variable

```java
char letter = 'A';
char numChar = '4';
```

**Note:**

- 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.

## ASCII and Unicode

- **ASCII** (American Standard Code for Information Interchange) is a 7-bit character encoding standard that can represent 128 characters, including uppercase and lowercase letters, digits, and special characters.
- **Unicode** is a 16-bit character encoding standard that can represent all characters in most languages, including Chinese, Japanese, and Korean.

**Example:** ASCII for the character 'A'

```
char ch = 'A';
```

**Example:** Unicode for the character 'ក'

```
char ch = 'ក';
```

## Escape Sequences for Special Characters

| Escape Sequence | Description |
|---|---|
| `\'` | Single quote |
| `\"` | Double quote |
| `\\` | Backslash |
| `\n` | Newline |
| `\r` | Carriage return |
| `\t` | Tab |
| `\b` | Backspace |
| `\f` | Formfeed |
| `\ddd` | Octal character |
| `\uxxxx` | Unicode character |

## **Example:** Escape Sequence

```java
char ch = '\n';     // Newline
char ch = '\t';     // Tab
char ch = '\'';     // Single quote
char ch = '\"';     // Double quote
char ch = '\\';     // Backslash
char ch = '\101';   // 'A'
char ch = '\u0041'; // 'A'
```

## Casting between char and Numeric Types

- Characters are represented as integers in Java. You can cast a character to an integer and vice versa.

**Example:** Casting between char and Numeric Types

```
char ch = (char)65;         // ch is character 'A'
int i = (int)'A';           // i is 65
```

**Note:** Characters can also be compared using relational operators, similar to numbers, based on their Unicode values.

## Comparing and Testing Characters

- Java provides methods in the `Character` class for testing characters.

**Example:** Comparing and Testing Characters

```
Character.isDigit('a');      // false
Character.isLetter('a');     // true
Character.isLowerCase('a');  // true
Character.isUpperCase('a');  // false
Character.toLowerCase('T');  // 't'
Character.toUpperCase('q');  // 'Q'
```

## 4.4. The String Type

## The String Type:

- A string is a sequence of characters.
- In Java, strings are objects of the `String` class.
- You can create a string by enclosing characters in double quotation marks (`"`).

**Example:** Declaring a String Variable

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

## String Methods:

- **length()**: Returns the number of characters in the string.
- **charAt(index)**: Returns the character at the specified index in the string.
- **concat(s)**: Concatenates the specified string to the end of this string.
- **toUpperCase()**: Converts all characters in the string to uppercase.
- **toLowerCase()**: Converts all characters in the string to lowercase.
- **trim()**: Removes whitespace from both ends of the string.

## **Example:** String Methods

```java
String message = "Welcome to Java";
int length = message.length(); // 15
char ch = message.charAt(0);   // 'W'
String newMessage = message.concat(" Programming"); // "Welcome to Java Programming"
String upper = message.toUpperCase(); // "WELCOME TO JAVA"
String lower = message.toLowerCase(); // "welcome to java"
String trimmed = message.trim(); // "Welcome to Java"
```

## Reading a String from the Console:

- To read a string from the console, you can use the `next()` or `nextLine()` methods of the `Scanner` class:

**Example:** Reading a String from the Console

```java
Scanner input = new Scanner(System.in);
System.out.print("Enter a string: ");
String s = input.nextLine();
System.out.println("You entered: " + s);
```

## Reading a Character from the Console

- To read a character from the console, read a string and then get the character at the desired position:

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

## 4.5. Case Studies

Practice.

## 4.6. Formatting Console Output

## The `printf` Method

You can use the `System.out.printf` method to display formatted output on the console.

**Syntax:**

```
System.out.printf(format, item1, item2, ..., itemN);
```

**Explanation:**

- The `format` is a string that specifies how the items are formatted.
- The `item1`, `item2`, ..., `itemN` are the items to be displayed.

## **Example:** Displaying None Formatted Output

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

## **Example:** Displaying Formatted Output

```
double amount = 12618.98;
double interestRate = 0.0013;
double interest = amount * interestRate;
System.out.printf("Interest is $%4.2f", interest);
```

The `f` in the `printf` stands for formatted, implying that the method prints an item in some format.

## Format Specifiers

| Format Specifier | Output |
|---|---|
| `%b` | A Boolean value |
| `%c` | A character |
| `%d` | A decimal integer |
| `%f` | A floating-point number |
| `%e` | A number in standard scientific notation |
| `%s` | A string |

## **Example:** Format Specifiers

```
System.out.printf("%5c", 'a');
// Output: '    a' (Character with four spaces before it)
System.out.printf("%6b", false);
// Output: ' false' (Boolean value with one space before it)
System.out.printf("%5d", 123);
// Output: '  123' (Integer with two spaces before it)
System.out.printf("%10.2f", 123.45);
// Output: '    123.45' (Floating-point with width 10 and 2 digits after the decimal)
System.out.printf("%12s", "Hello");
// Output: '       Hello' (String with seven spaces before it)
System.out.printf("%08d", 123);
// Output: '00000123' (Integer padded with leading zeros)
System.out.printf("%,8d", 12345678);
// Output: '12,345,678' (Integer with thousand separators)
System.out.printf("%-8d", 123);
// Output: '123     ' (Integer left-justified)
```