Math Methods and Characters

Common Mathematical Functions

- Java provides many useful methods in the Math class for performing common mathematical functions.
- In addition to methods, the Math class provides two useful double constants, PI and E (the base of natural logarithms).
- You can use these constants as Math.PI and Math.E in any program.

Trigonometric Methods

TABLE 4.1 Trigonometric Methods in the Math Class

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 degree.
`toDegree(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.

Trigonometric Methods

```
Math.toDegrees(Math.PI / 2) returns 90.0
Math.toRadians(30) returns 0.5236 (same as π/6)
Math.sin(0) returns 0.0
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
Math.cos(Math.PI / 6) returns 0.866
Math.cos(Math.PI / 2) returns 0
Math.asin(0.5) returns 0.523598333 (same as π/6)
Math.acos(0.5) returns 1.0472 (same as π/3)
Math.atan(1.0) returns 0.785398 (same as π/4)
```

Exponent Methods

- There are five methods related to exponents in the Math class.

TABLE 4.2 Exponent Methods in the Math Class

Method	*Description*
`exp(x)`	Returns e raised to power of x (e^x).
`log(x)`	Returns the natural logarithm of x ($\ln(x) = \log_e(x)$).
`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 ($\sqrt{x}$) for x >= 0.

Exponent Methods

```
Math.exp(1) returns 2.71828
Math.log(Math.E) returns 1.0
Math.log10(10) returns 1.0
Math.pow(2, 3) returns 8.0
Math.pow(3, 2) returns 9.0
Math.pow(4.5, 2.5) returns 22.91765
Math.sqrt(4) returns 2.0
Math.sqrt(10.5) returns 4.24
```

Rounding Methods

- The Math class contains five rounding methods.

TABLE 4.3 Rounding Methods in the Math Class

Method	*Description*
`ceil(x)`	x is rounded up to its nearest integer.
`floor(x)`	x is rounded down to its nearest integer.
`rint(x)`	x is rounded to its integer.
`round(x)`	Returns Math.floor(x + 0.5).

Rounding Methods

```
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.round(2.6f) returns 3 // Returns int
Math.round(2.0) returns 2 // Returns long
Math.round(-2.0f) returns -2 // Returns int
Math.round(-2.6) returns -3 // Returns long
Math.round(-2.4) returns -2 // Returns long
```

The min, max, and abs Methods

```
Math.max(2, 3) returns 3
Math.max(2.5, 3) returns 3
Math.min(2.5, 4.6) returns 2.5
Math.abs(-2) returns 2
Math.abs(-2.1) returns 2.1
```

The random Method

- This method generates a random double value greater than or equal to 0.0 and less than 1.0 (0 <= Math.random() < 1.0).
- Example: The next code:
- `System.out.println(Math.random());`
- `System.out.println(Math.random());`
- `System.out.println(Math.random());`
- Generates:
- `0.03434630657639481`
- `0.8164034994234213`
- `0.4968215610214594`

The random Method

- You can use it to write a simple expression to generate random numbers in any range.

```
(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.

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.

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

Escape Sequences for Special Characters

- The backslash \ is called an escape character.
- It is a special character.
- It is used to display escape sequences (\n, \t, …) and special characters (\\, \", …)

Escape Sequences for Special Characters

TABLE 4.5 Escape Sequences

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

Escape Sequences for Special Characters

- System.out.println("He said \nJava is fun");
  displays:

He said // new line

Java is fun

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

He said "Java is fun " // quotation marks

Casting between char and Numeric Types

- A char can be cast into any numeric type, and vice versa.

```
char ch = (char)65.25;
// Decimal 65 is assigned to ch
System.out.println(ch); // ch is character A
System.out.println(++ch); // ch is character B
```

Casting between char and Numeric Types

- When a char is cast into a numeric type, the character's Unicode is cast into the specified numeric type.

```
int i = (int)'A';
// The Unicode of character A is assigned to i
System.out.println(i); // i is 65
```

Comparing and Testing Characters

- Two characters can be compared using the relational operators just like comparing two numbers.
  - '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).

Comparing and Testing Characters

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

TABLE 4.6 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.

Example

```
System.out.println("isDigit('a') is " + Character.isDigit('a'));
System.out.println("isLetter('a') is " + Character.isLetter('a'));
System.out.println("isLowerCase('a') is "
  + Character.isLowerCase('a'));
System.out.println("isUpperCase('a') is "
  + Character.isUpperCase('a'));
System.out.println("toLowerCase('T') is "
  + Character.toLowerCase('T'));
System.out.println("toUpperCase('q') is "
  + Character.toUpperCase('q'));
```

- Output

```
isDigit('a') is false
isLetter('a') is true
isLowerCase('a') is true
isUpperCase('a') is false
toLowerCase('T') is t
toUpperCase('q') is Q
```