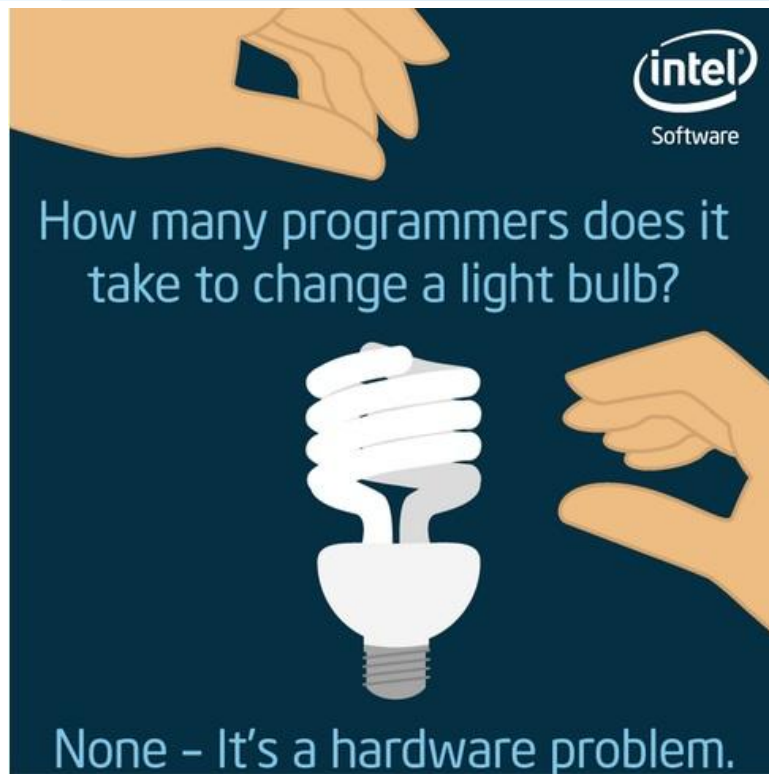


Math Functions and Characters



CS2011: Introduction to Programming I

Math Functions

Trigonometric Methods

Method	Description
<code>Math.sin(radians)</code>	Returns the sine of an angle in radians.
<code>Math.cos(radians)</code>	Returns the cosine of an angle in radians.
<code>Math.tan(radians)</code>	Returns the tangent of an angle in radians.
<code>Math.toRadians(degree)</code>	Returns a radian value for the given degree value.
<code>Math.toDegree(radians)</code>	Returns a degree value for the given radian value.
<code>Math.asin(a)</code>	Returns the angle in radians for the inverse of sine.
<code>Math.acos(a)</code>	Returns the angle in radians for the inverse of cosine.
<code>Math.atan(a)</code>	Returns the angle in radians for the inverse of tangent.

Exponent Methods

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

Rounding Methods

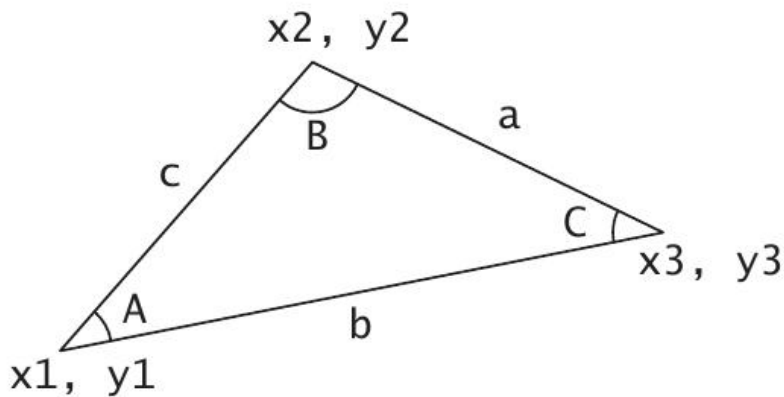
Method	Description
<code>Math.ceil(x)</code>	x is rounded up to its nearest integer. Returns a double.
<code>Math.floor(x)</code>	x is rounded down to its nearest integer. Returns a double.
<code>Math rint(x)</code>	x is rounded up to its nearest integer. If x is equally close to two integers, the even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

Min, Max and Absolute Value

- ▶ `Math.min(a, b)`
 - returns the minimum of a and b.
- ▶ `Math.max(a, b)`
 - returns the maximum of a and b.
- ▶ `Math.abs(x)`
 - returns the absolute value of x.

Example: Compute Triangle Angles

- ▶ Given the three sides of a triangle we can calculate the angles by using the formulas in the following diagram.



$$\begin{aligned} A &= \arccos((a * a - b * b - c * c) / (-2 * b * c)) \\ B &= \arccos((b * b - a * a - c * c) / (-2 * a * c)) \\ C &= \arccos((c * c - b * b - a * a) / (-2 * a * b)) \end{aligned}$$

- ▶ See Code: `ComputeAngles.java`

Character Data Type

The char Data Type

- ▶ The **char** data type represents a single character.
- ▶ A character literal is enclosed in single quotation marks ' '.
- ▶ Example:

```
char letter = 'A';  
char numChar = '4';
```
- ▶ NOTE: The increment and decrement operators can also be used on char variables to get the next or preceding character. For example, the following statements display character 'b'.

```
char ch = 'a';  
System.out.println(++ch);
```

Unicode and ASCII Code

- ▶ Remember that computers use binary numbers internally.
- ▶ How then, is a character stored?
 - as a sequence of 0s and 1s
 - each character is mapped to a binary representation ***encoding***
 - characters can be encoded in different ways and how they are encoded is defined by an ***encoding scheme***

Unicode and ASCII Code

- ▶ Java supports Unicode,
 - encoding scheme established by the Unicode Consortium
 - supports the interchange, processing, and display of written texts in many of the world's languages
 - originally designed as a 16-bit character encoding.
- ▶ the **char** data type was originally designed to support character encoding
 - 16 available bits means we can represent 65,536 different types of characters
- ▶ Unicode was developed to extend the encoding to include 1,112,064 unique characters.
 - characters beyond the original 16-bit limit are called supplementary characters.

Unicode and ASCII Code

- ▶ A 16-bit Unicode takes two bytes, preceded by `\u`, expressed in four hexadecimal digits that run from `\u0000` to `\uFFFF`.
- ▶ Example:

```
char letter = '\u0041'; (Unicode)  
char numChar = '\u0034'; (Unicode)
```

Unicode and ASCII Code

- ▶ Most computers use ASCII (American Standard Code for Information Interchange) to represent the characters.
- ▶ 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. (So the ASCII characters are a subset of the Unicode standard)

Unicode and ASCII Code

- ▶ You can use ASCII characters such as 'X', '1', and '\$' or you can use the Unicode values.

- ▶ Example:

```
char letter = 'A';  
char letter = '\u0041';
```

Escape Sequences

- ▶ How do you print a message with quotation marks in the output?

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

- The statement has a syntax error
 - The compiler thinks the 2nd quotation character is the end of the string.
-
- ▶ Java defines escape sequences to represent special characters
 - An escape sequence begins with '\'

Escape Sequences

- ▶ `\b` Backspace
- ▶ `\t` Tab
- ▶ `\n` New Line
- ▶ `\\` Backslash
- ▶ `\"` Double Quote
- ▶ `\'` Single Quote

Escape Sequences

▶ Example:

- Wrong:

- ◆ `System.out.println("He said "Java is fun");`

- Correct:

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

▶ Note: \ and " together are considered to be one character \"

Casting between char and Numeric Types

- ▶ A char can be cast into any numeric type and vice versa.
- ▶ When a char is cast into a numeric type, the character's unicode is cast into the specified numeric type
- ▶ Example:
 - `int i = (int)'A'; //i is now equal to 65`
 - `char ch = (char)65.25; //ch is now 'A'`

Operations on Characters

- ▶ The math operators can be used on characters
 - The math operation is applied to the unicode values of each character.
 - Example:
 - ♦ `int i = '2' + '3'`
 - ♦ '2' is 50 and '3' is 51 so $50 + 51 = 101$
- ▶ You can also compare characters using the relational operators.
 - The comparison is applied to the unicode values of each character.
 - Example:
 - ♦ `'a' < 'b'` is true
 - ♦ 'a' is 97 and 'b' is 98

The Character Class

- ▶ The Character class also provides some methods for testing characters.
- ▶ `Character.isDigit(ch)` : Returns true if the given character is a digit.
- ▶ `Character.isLetter(ch)` : Returns true if the given character is a letter.
- ▶ `Character.isLetterOrDigit(ch)` : Returns true if the given character is a letter or digit.
- ▶ `Character.isLowerCase(ch)` : Returns true if the given character is a lowercase letter.
- ▶ `Character.isUpperCase(ch)` : Returns true if the given character is an uppercase letter.
- ▶ `Character.toLowerCase(ch)` : Returns the lowercase of the character.
- ▶ `Character.toUpperCase(ch)` : Returns the uppercase of the character.

References

- ▶ Liang, Chapter 04: Math Functions, char Datatype and Strings.