CS1150
Principles of Computer Science
Math Functions, Characters and Strings

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

Mathematical Functions

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

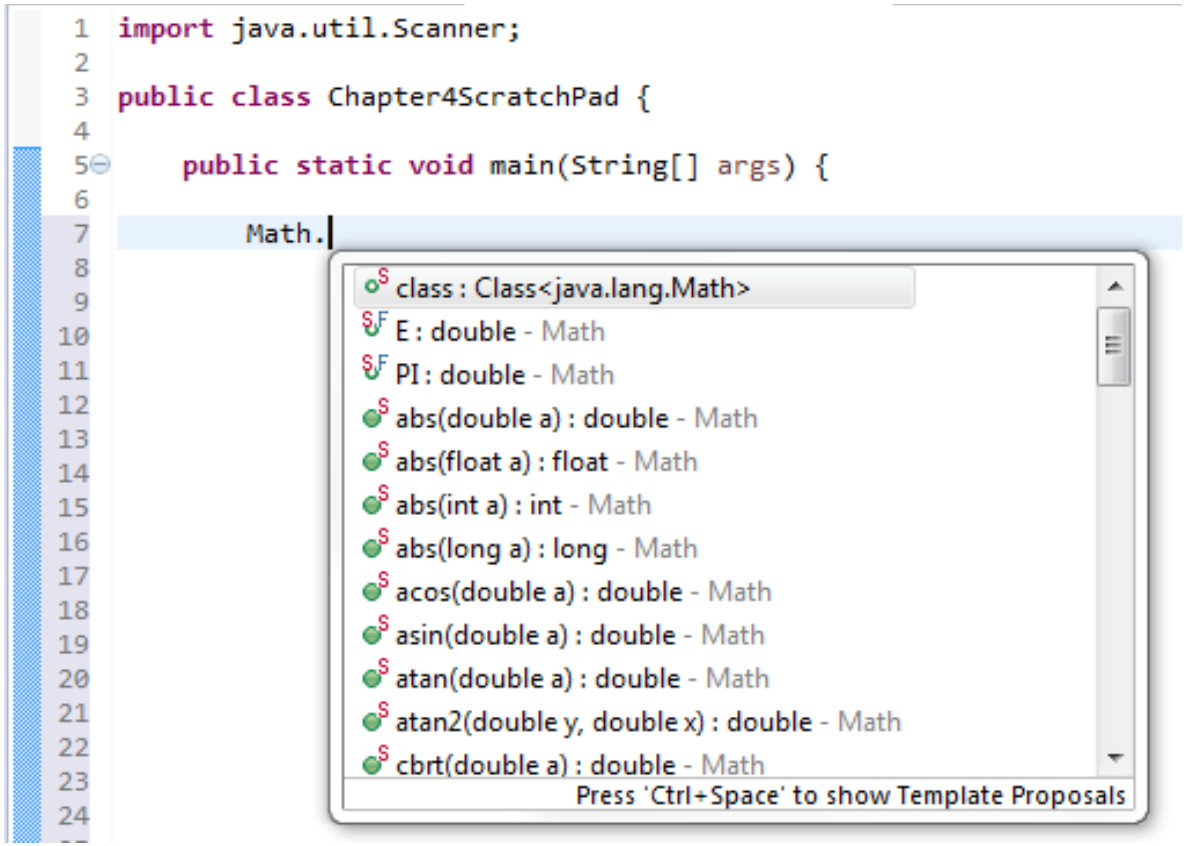
(we have used Math.sqrt(), Math.pow(), Math.random())

The `Math` Class

- Math class is different from Scanner class
  - Don’t need to create a Math instance
    - *No need* for Math myMath = new Math();
  - We call Math class methods without creating an instance
  - Use Math.methodName() directly
    - double squareRoot = **Math.sqrt**(25);

The Math Class

- Type Math.
  - You can see constants and methods



The `Math` Class

- Class constants:
  - PI (3.14159…)
  - E (2.71828…base of natual log)
- Class methods:
  - Trigonometric Methods
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods

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).

Numbers1.java

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
```

Numbers3.java

The random Method

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.

Previous example

- Math.random()
  - How to generate a random integer between [lower, upper)?
    - Example: int lower=100, upper=120;
    - randomDouble = Math.random(); // [0.0, 1.0)
    - randomDouble = randomDouble * (upper-lower); // [0.0, 20.0)
    - randomDouble = lower + randomDouble; // [100.0, 120.0)
    - randomInt = (int) randomDouble; // cast double → int
  - Or in one step
    - randomInt = (int) (lower + Math.random() * (upper-lower));

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).

Rounding Methods 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
```

Rounding Methods Examples

- Math.round(x) returns int or a long (depending on if the argument is a float or a double)

```
int x =  Math.round (81.7);          // This won't work?  Why?
                    // 81.7 is a double so need to store result in long

int intResult = Math.round (81.7f);          // Returns 82 - an int
long longResult = Math.round(81.7);          // Returns 82 - a long
```

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
```

Numbers2.java

Trigonometric Methods

- sin(double a)
- cos(double a)
- tan(double a)
- acos(double a)
- asin(double a)
- atan(double a)

Radians

toRadians(90)

toDegrees()

```
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 0.0
```

Trigonometric Methods

- Provide an angle in radians

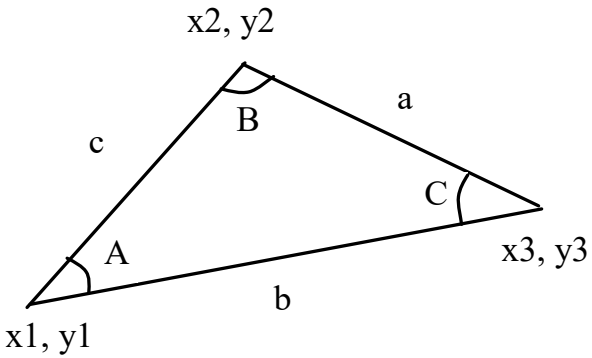
```
double sinOfZero = Math.sin(0);  // 0 is in radian
System.out.println ("Math.sin(0) = " + sinOfZero);  // Displays 0.0
```

- Examples with a value in degrees

```
double angleInRadians = Math.toRadians(60);  // 60 is degree
System.out.println("Sixty degrees = " + angleInRadians + " radians");
double sinOfAngle = Math.sin(angleInRadians);
System.out.println ("Math.sin(60) = " + sinOfAngle);
```

Numbers3.java

Case Study: Computing Angles of a Triangle




```
A = acos((a * a - b * b - c * c) / (-2 * b * c))
B = acos((b * b - a * a - c * c) / (-2 * a * c))
C = acos((c * c - b * b - a * a) / (-2 * a * b))
```

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.

Let's see ComputeAngle.java example.

Character Data Type

- Values: one single character
  - Use single quote ‘x’ to represent a character (double quotes “xxx” are for Strings)
    - char middleInitial = 'M';
    - char numCharacter = '4'; // Assigns digit character 4 to numCharacter
    - System.out.println(numCharacter); // Displays 4
  - Placing a character in “” it is no longer a char: it is a String (with one char in it)
    - char middleInitial = "M"; // Error - cannot convert String to char

Example: Char1.java

Base 10, base 2 and base 16

- The Decimal Number System is also called "Base 10"
  - There are 10 symbols (0,1,2,3,4,5,6,7,8 and 9)
  - There is no symbol for "ten". "10" is actually two symbols put together, a "1" and a "0”
  - How to get a number in base 10? Example: 23, 123
  - But don't have to use 10 as a "Base". Could use 2 ("Binary"), 16 ("Hexadecimal"), or any number

Base 10, base 2 and base 16

- Binary (base 2)
  - 000, 001, 010, 011, 100, …
  - Computers can only recognize 0/1's
- Hexadecimal (base 16)
  - Decimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
  - Hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Example: 3BD

Character Data Type

Four hexadecimal digits.

```
char letter = 'A'; (ASCII)
char numChar = '4'; (ASCII)
char letter = '\u0041'; (Unicode)
char numChar = '\u0034'; (Unicode)
```

NOTE: The increment and decrement operators can also be used on char variables to get the next or preceding ASCII/Unicode character. For example, the following statements display character b.

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

What's ASCII

- A computer cannot store characters
  - The only thing it can store and work with are **bits**
  - A bit can only have two values: 1 or 0 (an "actual" bit is a blip of electricity that either is or isn't there)
- American Standard Code for Information Interchange (ASCII): 8-bit character scheme
  - Provides encoding for 128 characters (0 to 127)
  - Based on ordering of English alphabet

ASCII Code for Commonly Used Characters

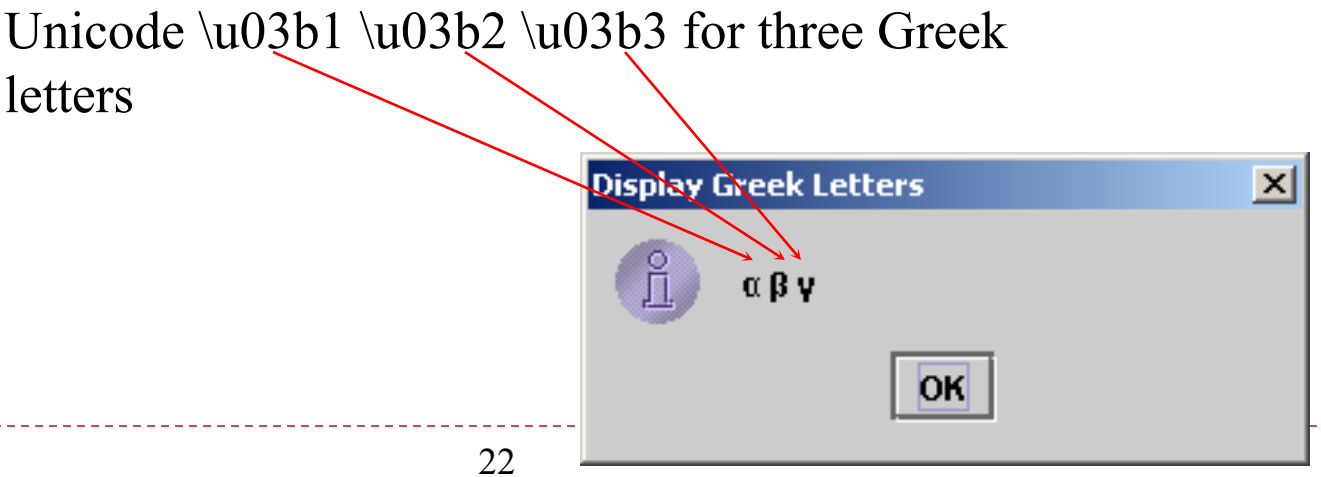
Characters	Code Value in Decimal	Unicode Value
`'0'` to `'9'`	48 to 57	\u0030 to \u0039
`'A'` to `'Z'`	65 to 90	\u0041 to \u005A
`'a'` to `'z'`	97 to 122	\u0061 to \u007A

More information: http://kunststube.net/encoding/

Unicode Format

Java characters use *Unicode*, a 16-bit encoding scheme to support texts in the world's diverse languages.

Unicode takes two bytes, preceded by \u, expressed in four hexadecimal numbers: from '\u0000' to '\uFFFF'. So, Unicode can represent `65535 + 1 characters`.



Escape Sequences for Special Characters (Considered a Single character)

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

Casting between char and Numeric Types

```
int i = 'a'; // Same as int i = (int)'a';

System.out.println (”i = " + i); // i = 97
```

all numeric operators can be applied to the char operands

```
char c = 97; // Same as char c = (char)97;

System.out.println ("c = " + c);  // c = a
```

Increment and decrement can be used on char variables to get the next or preceding ASCII/Unicode character.

```
char ch = 'a';
System.out.println(++ch); //shows character b
```

Comparing and Testing 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");
```

all numeric operators can be applied to the char operands

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.

Like the Math class - you don't create an instance of this class

Methods in the Character Class

```
if (Character.isUpperCase(c)) {
        System.out.println(”c is an uppercase letter");
        System.out.println ("Its lowercase version is: " +
                                    Character.toLowerCase(c));
}
```

Example: Char2.java