Java Basics:
The Math Class

Math functions

- In high school trigonometry class, you often wrote equations like:

```
    sin(x)/cos(x)
```

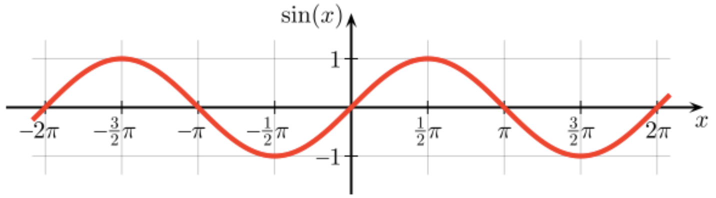
- In computer science, we would often want to save the value of such an expression in a variable:

```
    double tangent = sin(x)/cos(x);
```

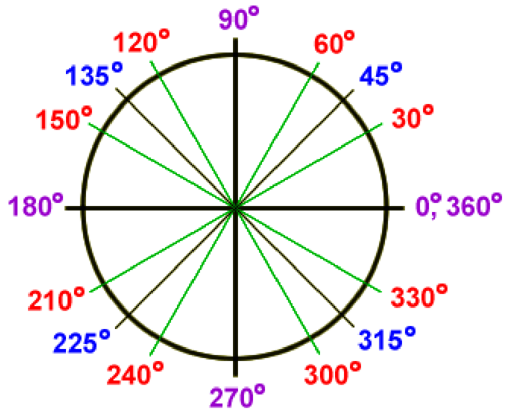
- In these cases, `sin(x)` is a function which takes a parameter and returns a value that can be used in a larger expression

The `Math` Class

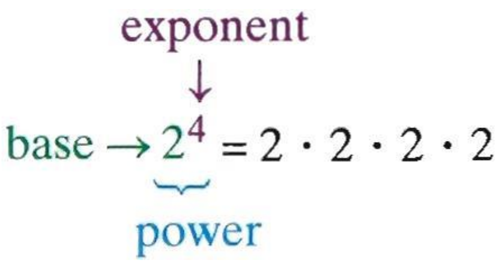
- Java provides many system-supplied functions
- The `Math` class provides many standard mathematical functions



$$\sqrt{x}$$



$$|x|$$



The `Math` Class

- Java provides many system-supplied functions
- The `Math` class provides many standard mathematical functions
  - In Java terminology, we refer to such functions as *methods*
  - All of these methods return values
  - These methods are called by using the class name (`Math`) followed by a dot and the method name:

```
Return_Value = Math.Method_Name(Parameters);
```

  - The Math class is provided automatically (no import needed)

Calling `Math` methods

- Most `Math` methods require that you provide parameters
  - These are the values you want the method to operate on
  - The parameters are specified in the parentheses that follow the method name
  - If a method accepts more than one parameter, they are comma separated
  - If a method does not accept any parameters, you still need to use the parentheses – only they will be empty in this case

- Each method produces ("*returns*") a numeric result
  - The result may be used as an expression (printed, stored, etc.)

Calling `Math` methods

- Examples:

```
double squareRoot = Math.sqrt(625.0);
out.println(squareRoot);                // 25.0

int absoluteValue = Math.abs(-25);
out.println(absoluteValue);             // 25

int a=7, b=4;
out.println(Math.min(a, b) + 5);        // 9
```

Java's `Math` class

Method name	**Description**
`Math.abs(`*value*`)`	absolute value
`Math.round(`*value*`)`	nearest whole number
`Math.ceil(`*value*`)`	rounds up
`Math.floor(`*value*`)`	rounds down
`Math.log10(`*value*`)`	logarithm, base 10
`Math.max(`*value1*`, `*value2*`)`	larger of two values
`Math.min(`*value1*`, `*value2*`)`	smaller of two values

Java's `Math` class

Method name	**Description**
`Math.pow(`*base*`, `*exp*`)`	*base* to the *exp* power
`Math.sqrt(`*value*`)`	square root

`Math.sin(`*value*`)`
`Math.cos(`*value*`)`
`Math.tan(`*value*`)`
	sine/cosine/tangent of
	an angle in radians

`Math.toDegrees(`*value*`)`
`Math.toRadians(`*value*`)`
	convert between degrees
	and radians

`Math.random()`	random `double` between 0 and 1

Java's `Math` class

Constant	Description
`Math.E`	2.7182818...
`Math.PI`	3.1415926...

- Example

```
area = Math.PI * radius * radius;
```

The `Math` Class

- Method `round` returns to the nearest whole number
  - If its argument is of type `double`, it returns a whole number of type `long`
- Method `floor` (`ceil`) returns the largest (smallest) whole number that is less (greater) than or equal to its argument
  - I.e., they always round down (up) rather than to the nearest whole number
  - If the argument is of type `double`, they return a whole number of type `double`

The `Math` Class

- The functions `round`, `floor` or `ceil` always return whole numbers
- But to store a value returned by them in a variable of type `int`, a cast must be used.

```
double val = 8.53;
int nearest =     (int)Math.round(val);  // 9
int lowerBound = (int)Math.floor(val);  // 8
int upperBound = (int)Math.ceil(val);   // 9
```