

# Unit 2: Using Objects

## Math And Wrapper Classes

Adapted from:

- 1) Building Java Programs: A Back to Basics Approach  
by Stuart Reges and Marty Stepp
- 2) Runestone CSAwesome Curriculum

This work is licensed under the  
[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

<https://longbaonguyen.github.io>

# Static Methods

The Math class has many useful **static** methods. The class is part of the **java.lang package**(group of classes) that is available by default(no need to import to use). To call these, use the syntax:

```
Math.methodName (parameters) ;
```

```
double answer = Math.sqrt(9.2) ;
```

```
int b = Math.round(5.6755) ;
```

# Java's Math class

Method name	Description
<code>int abs(int x)</code> <code>double abs(double x)</code>	returns the absolute value of a int or double value (overloaded method)
<code>double pow(double base, double exponent)</code>	Returns the value of the first parameter raised to the power of the second parameter
<code>double sqrt(double x)</code>	Returns the positive square root of a double value
<code>double random()</code>	Returns a random double value <b>greater than or equal</b> to 0.0 and <b>less than</b> 1.0

Constant	Description
<code>Math.E</code>	2.7182818...
<code>Math.PI</code>	3.1415926...

# Calling Math methods

- Examples:

```
double squareRoot = Math.sqrt(121.0);  
System.out.println(squareRoot);           // 11.0
```

```
int absoluteValue = Math.abs(-50);  
System.out.println(absoluteValue);        // 50
```

```
System.out.println(Math.min(3, 7) + 2);    // 5
```

- The `Math` methods do not print to the console.
  - Each method produces ("returns") a numeric result.
  - Remember to store, print or use the result in some expression

# Math questions

- Evaluate the following expressions:
  - `Math.abs(-1.23)`
  - `Math.pow(3, 2)`
  - `Math.pow(10, -2)`
  - `Math.sqrt(121.0) - Math.sqrt(256.0)`

# Math questions

Write a method `withinHalf` which takes two double parameters and return true if they are within .5 of each other and false otherwise.

```
withinHalf(4, 5.1) // returns false  
withinHalf(3.4, 3.9) // returns true  
withinHalf(3.9, 3.4) // returns true  
withinHalf(-1.2, -1.1) // returns true
```

```
public static boolean withinHalf(double x, double y)  
{  
    return Math.abs(x - y) <= .5;  
}
```

# Quirks of real numbers

- Some Math methods return double or other non-int types.

```
int x = Math.pow(10, 3);    // ERROR: incompat. types
```

- Some double values print poorly (too many digits).

```
double result = 1.0 / 3.0;  
System.out.println(result);    // 0.3333333333333333
```

- The computer represents doubles in an imprecise way.

```
System.out.println(0.1 + 0.2);
```

– Instead of 0.3, the output is 0.30000000000000004

# Random Numbers



# Random numbers

- `Math.random()` produces a number from 0(inclusive) to 1 exclusive.

- `double x = Math.random(); // 0.0 <= x < 1.0`
- `double x = 3 * Math.random(); // 0.0 <= x < 3.0`
- `double x = Math.random() + 2; // 2.0 <= x < 3.0`
- `double x = 5 * Math.random() + 4; // 4.0 <= x < 9.0`

In general, to produce a random real number in the range `[low,high)`,

- `double x = (high - low) * Math.random() + low;`

**Generate a random real value in `[7.0,15.0)`.**

```
double x = 8 * Math.random() + 7;
```

# Random Integers

How do we generate random integers? Use casting!

```
int x = (int) (100 * Math.random());  
// random integer 0 to 99 inclusive.
```

```
int y = (int) (100 * Math.random()) + 4;  
// random integer 4 to 103 inclusive.
```

```
int z = (int) (2 * Math.random());  
// random integer 0 or 1, useful for heads/tails
```

# More Examples

```
int x = (int) Math.random() * 5;  
// x = 0
```

```
int y = (int) (6 * Math.random()) - 10;  
// integer from -10 to -5 inclusive.
```

```
double z = 3 * Math.random() + 5;  
//random double in [5,8)
```

# Wrapper Classes

- A wrapper class takes an existing value of primitive type and “wraps” or “boxes” it in an object, and provides a new set of methods for that type.
- It can be used in Java container classes that requires the item to be objects. (Arraylist)

# Wrapper Classes

The wrapper class allows

1. The construction of an object from a single value(wrapping or boxing the primitive in a wrapper object.
2. The retrieval of a primitive value(unwrapping or unboxing from a wrapper object.)

# Wrapper Classes

You will need to know two wrapper classes:

- 1) Integer class
- 2) Double class

# Wrapper Classes

Integer and Double are wrapper classes...not Rapper Classes.

These are Rapper Classes:

```
public class Tupac{...}
```

```
public class Biggie extends Tupac{...}
```

```
public class JayZ extends Biggie{...}
```

```
public class KendrickLamar extends Biggie{...}
```



# Integer Class

The Integer class wraps a value of type `int` in an object.

Here are two useful methods:

`Integer(int value)`: Constructs an Integer object from an `int`.

`int intValue()`: Returns the value of this Integer as an `int`.

The class also has two static variables: A Java integer uses 32 bits(0 or 1) of memory. One bit is used for the sign(+ or -). Thus:

`Integer.MIN_VALUE`— The minimum value represented by an `int` or Integer  
 $= -2^{31} = -2147483648$

`Integer.MAX_VALUE`— The maximum value represented by an `int` or Integer  
 $= 2^{31} - 1 = 2147483647$



# Double Class

The Double class wraps a value of type double in an object.

Here are two useful methods:

`Double(double value)` : Constructs an Double object from a double.

`double doubleValue()` : Returns the value of this Double as a double

# Examples

```
Integer intObj = new Integer(6); //boxes 6 in Integer object  
int j = intObj.intValue(); //unboxes 6 from Integer object
```

```
Double dObj = new Double(2.5); //boxes 2.5 in Double object  
double d = dObj.doubleValue(); //unboxes 2.5 from Double object
```

# Auto-Boxing and Unboxing

Auto-boxing is the automatic boxing of primitive types in their wrapper classes.

To retrieve the value of an Integer(or Double), the `intValue()` or `doubleValue()` method can be called(unboxing).

Auto-unboxing is the automatic conversion of a wrapper class to its corresponding primitive type. This means you don't need to explicitly call the `intValue()` or `doubleValue()`.

# Autoboxing and Auto-unboxing

```
Integer a = new Integer(5);  
int x = a.intValue(); // unboxing x = 5  
int y = a; // auto-unboxing, easier.
```

```
Integer b = new Integer(7); // boxing  
Integer c = 7; // auto-boxing  
int z = a + x; // auto-unboxing
```

```
Double d = new Double(7.5); // boxing  
double e = d.doubleValue(); // unboxing  
double f = d + 2.0; // auto-unboxing
```

# Lab

Go to the following repl on repl.it:

<https://repl.it/@LongNguyen18/Unit2MathClassLab>

Fork it and follow the comments to complete the code.

# References

1) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum:

<https://runestone.academy/runestone/books/published/csawesome/index.html>

For more tutorials/lecture notes in Java, Python, game programming, artificial intelligence with neural networks:

<https://longbaonguyen.github.io>