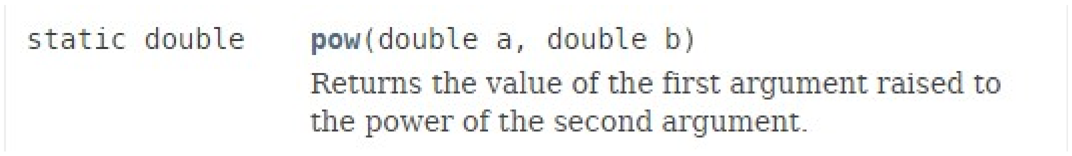
02.03 Virtual Lecture Notes

Part 1: The power of method: Math.pow( )

The Method Summary of the **pow()** method gives an overview of how to use it. The **pow()** method behaves just like the y^x key on a calculator.



```
static double    pow(double a, double b)
                 Returns the value of the first argument raised to
                 the power of the second argument.
```

We could use the following statement to find the value of 2^{15}:

```
double powValue = Math.pow(2,15);
```

The arguments (2 and 15) of the method need to be numeric values but don't have to be the actual numbers. They could just as easily be variables. For example,

```
//example of raising a number to the nth power
double number = 2;
double nthPower = 15;
double powValue = Math.pow(number, nthPower);
```

Here, 2 and 15 are assigned to the double variables number and nthPower, respectively.

- **`Math.pow()`** invokes the **`pow()`** method of the Math class.
- Two arguments, the values assigned to number and nthPower, are passed to the **`pow()`** method.
- The **`pow()`** method computes $\text{number}^{\text{nthpower}}$ and assigns the calculated value of 2^{15} to the variable powValue.

Examine the section of code in the MathMethodsDemo class that deals with the pow() method. Run the program and observe the output. Modify the program using values for number and nthPower of your choice. Try using doubles in place of the integers.

Part 2: The square root method: Math.sqrt( )

The Method Summary of the **`sqrt()`** method tells the essential information needed to use it. The

`sqrt()` method behaves just like the √X key on a calculator.

static double	**sqrt**(double a) Returns the correctly rounded positive square root of a double value.

The **`sqrt()`** method takes a single parameter, so to find the square root of a number, just supply a double argument. For example, √17.05 would be,

```
//example of finding the square root of a value
double someNumber = 17.5;
double squareRoot = Math.sqrt(someNumber);
```

In this example, 17.5 is the double argument assigned to the someNumber variable.

- **`Math.sqrt()`** invokes the **`sqrt()`** method of the Math class.
- The argument, the value assigned to someNumber, is passed to the **`sqrt()`** method.
- The **`sqrt()`** method computes √17.5 and assigns the value to the variable squareRoot.

Uncomment the code with the square root example in the MathMethodsDemo program. Run the program and observe the output. Modify the program to calculate a few square roots of your choice.

To use the **`sqrt()`** method in an arithmetic expression, you might do something like this:

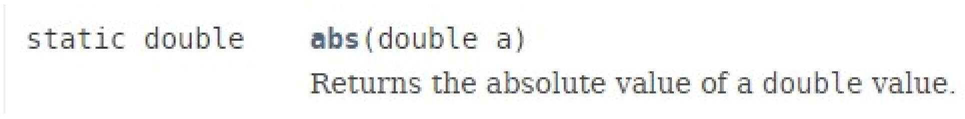
```
double value1 = 88.3;
double answer2 =  Math.sqrt(value1) * Math.pow(value1, 0.5);
```

Use a calculator to find the answer to this expression, then add the code to the program and print the result to verify your answer. How is it possible for the answer to be the same as the argument?

Part 3: The absolute value method: Math.abs( )

The Method Summary of the **`abs()`** method tells us everything we need to know to use it. The **`abs()`** method behaves just like the |x| key on a calculator. However, there are two versions of this method. Study the two Method Summaries below and see if you can spot the difference between the two methods.

static int	**abs**(int a) Returns the absolute value of an int value.



static double abs(double a)
Returns the absolute value of a double value.

It may seem unusual to have two methods with the same name, but this is common in Java. As long as the parameter list of methods with the same name is different, the compiler will invoke the correct method. In this case, one version of the **`abs()`** method takes an int parameter while the other takes a double. Re-using the name of a method like this is called **overloading**.

To find the absolute value of a number, just supply the **`abs()`** method with the appropriate argument. For example, |-34| would be,

```
//example of finding the absolute value of an integer
int integerNumber = −34;
int intAbsValue = Math.abs(integerNumber);
```

In this example, −34 is the argument assigned to the int variable `integerNumber`.

- **`Math.abs()`** invokes the version of the **`abs()`** method that takes an int parameter.
- The argument, the value assigned to integerNumber, is passed to the **`abs()`** method.
- The **`abs()`** method computes |−34| and assigns the value to the variable intAbsValue.

Uncomment the section of code with the example. Run the program and observe the output. Modify the program to calculate and print a few absolute values of your choice.

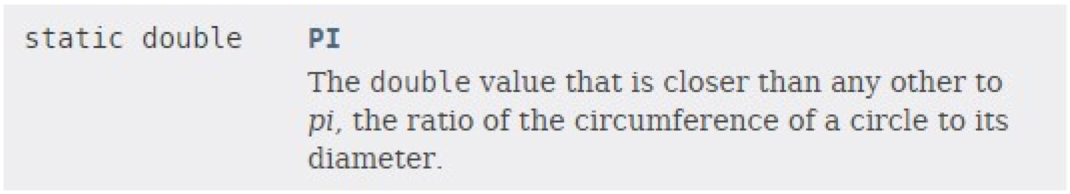
The abs() method could be used in an arithmetic expression as follows:

```
double myNumber = -23.75;
double answer3 = Math.abs(myNumber) + myNumber;
```

Before moving on, examine the Method Details for the **`abs()`** method. Why do you think the **`abs()`** method is so overloaded?

Part 4: The pi constant: Math.PI

In addition to methods, the Math class contains two constants: **`PI`** and **`E`**. You only need to be concerned about **`PI`** for the AP Computer Science exam. The Field Summary for **`PI`** provides an overview of this important mathematical constant.



static double	**PI** The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

The **`PI`** constant works just like the pi key on a calculator. To use **`PI`** in an arithmetic expression, do the following.

```
//example of using PI to calculate a circumference
double myRadius = 3.5;
double myCircumference = 2 * Math.PI * myRadius;
```

In this example, 3.5 is assigned to the double variable myRadius.

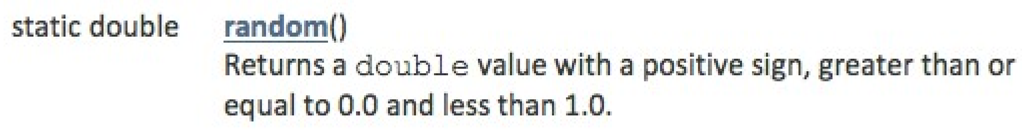
- **`Math.PI()`** invokes the **`PI`** constant of the Math class.
- The calculation is carried out using the value assigned to myRadius and the constant value assigned to PI by the Math class.

Uncomment the code segment with the PI example. Run the program and observe the output.

Modify the program to calculate and print a few calculations of your choice using **`PI`**. Before moving on, examine the Method Details for PI. You might want to also print the value assigned to PI and compare it to the value of pi used by your calculator.

Part 5: The random number method: Math.random( )

The Method Summary of the **`random()`** method explains how to use it. The **`random()`** method generates a value from 0–1 (exclusive).



static double	random() Returns a `double` value with a positive sign, greater than or equal to 0.0 and less than 1.0.

We could use the following statement generate a random value between 0 and 10 (exclusive):

```
(int)(Math.random() * 10);
```

`Math.random()` returns a double value from 0.0 to .99999, so if we want whole numbers values, we need to multiply first by our desired maximum value, plus 1 and then cast to int. For example,

```
//example of generating a value from 0-4 (inclusive)
int randNumber = (int)(Math.random() * 5);
```

In the above example, a random value is generated between 0.0 and .99999 and then it's multiplied by 5. Which makes the value generated between 0.0 and 4.99999. To truncate the value to an integer, we cast with int(). Once casted, the range becomes 0 to 4 (inclusive).

```
//example of generating a value from 4-11 (inclusive)
int randNumber2 = (int)(Math.random() * 8) + 4;
```

In the above example, a random value is generated between 0.0 and .99999 and then it's multiplied by 8. Which makes the value generated between 0.0 and 7.99999. To truncate the value to an integer, we cast with int() which makes the range 0 to 7. When we add 4, the range is adjusted to 4 to 11 (inclusive).

Uncomment the code segment with the **`random()`** methods. Run the program and observe the output. Modify the values to adjust the range used with the **`random()`** method.

A lot of information about Math class has been covered in this lesson. If you need information about other members of the Math class, your new familiarity with the structure and organization of the Java API should help you find what you need.

Print