Programming 1

WEEK 03 – Math Class

Advanced Math Operations

- The Math class in Java contains a variety of methods that perform mathematical functions
- All methods in Math class are static
- You can’t extend or create an instance of the Math class
- We discuss static methods further in this course

Advanced Math Operations

- All advanced operations are defined in the Math class.
- For example:
    - `Math.pow(a, b)` calculate a^b
        - Can be passes a combination of two ints, doubles, floats, or longs
        - Will return a double
        - Example: Math.pow(3,2.0) returns 9.0
    - `Math.sqrt(a)` calculate $\sqrt{a}$
        - Will return a double
        - Example: Math.sqrt(9) returns 3.0

Advanced Math Operations

• `Math.min(a, b)`	find the minimum value between a and b
• `Math.max(a, b)`	find the maximum value between a and b
• `Math.abs(a)`	calculate the absolute value of a: |a| - return the same data type

- These are the basic methods in that Math class that we usually use.

Advanced Math Operations

- Assume you want to calculate the standard deviation of 3 numbers, num1 = 3, num2 = 5, num3 = 10, we can use the formula:

$$\mathrm{SD} = \sqrt{\frac{\sum |x - \bar{x}|^2}{n}}$$

  - Where x is a number, $\bar{x}$ is the mean (average) of the three, n is 3.

```
int num1 = 3;
int num2 = 5;
int num3 = 10;
double mean = (num1 + num2 + num3)/3;
double std = Math.sqrt(Math.abs((Math.pow(num1 - mean, 2) + Math.pow(num2 - mean, 2)
            + Math.pow(num3 - mean, 2))) / 3);
```

- You have to be careful about the () when the formula gets long.

Advanced Math Operations

- If you want to round the value, you can also use methods in the Math class:

• `Math.round(a)`	round the value a:	3.1 -> 3.0	2.9 -> 3.0
• `Math.ceil(a)`	always bring the value up	3.1 -> 4.0	2.9 -> 3.0
• `Math.floor(a)`	always bring the value down	3.1 -> 3.0	2.9 -> 2.0

- For example, if we want to pay for the parking, we need to know how long was the car parked. And if you parked 2 hours and 2 mins, you have to pay for 3 hours, this is Math.ceil().

- If we want to check how old is one person, if the age is 27.9, which means the birthday is coming very soon. But as far as the birthday has not arrived, the age will still be 27, this is Math.floor().

Advanced Math Operations

- Math class also contains methods Trigonometric function:
  - `Math.sin(a)`
  - `Math.cos(a)`
  - `Math.tan(a)`
- Remember these methods receive an angle in radians instead of in degrees.
- For example
  - `Math.sin (30)` will not return 0.5, cause 30 is in degrees.
  - `Math.sin(Math.toRadians(30) )` will return 0.5, because `Math.toRadians()` receives a degree and convert it to radians.

Advanced Math Operations

- The same as Math.toRadians(a), there is also Math.toDegrees(a) which receives an angle in radians and convert it to degrees.
- PI is also defined in the Math class. If you want to calculate the area of a circle, we can use: Math.PI * radius * radius
- Attention: PI is a variable instead of a method, so there is no () after it.
- Further reading: Math Class in Java:

https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Demo..

Determines the roots of a quadratic equation: **`ax`**^{2} **`+ bx + c`**

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```java
public class Quadratic
{
   //-----------------------------------------------------------------
   //  Determines the roots of a quadratic equation.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      int a, b, c;  // ax^2 + bx + c
      double discriminant, root1, root2;

      Scanner scan = new Scanner(System.in);

      System.out.print("Enter the coefficient of x squared: ");
      a = scan.nextInt();

      System.out.print("Enter the coefficient of x: ");
      b = scan.nextInt();

      System.out.print("Enter the constant: ");
      c = scan.nextInt();

      // Use the quadratic formula to compute the roots.
      // Assumes a positive discriminant.

      discriminant = Math.pow(b, 2) - (4 * a * c);
      root1 = ((-1 * b) + Math.sqrt(discriminant)) / (2 * a);
      root2 = ((-1 * b) - Math.sqrt(discriminant)) / (2 * a);

      System.out.println("Root #1: " + root1);
      System.out.println("Root #2: " + root2);
   }
}
```

Sample Run

```
Enter the coefficient of x squared: 3
Enter the coefficient of x: 8
Enter the constant: 4
Root #1: -0.6666666666666666
Root #2: -2.0
```

Java Math.random()

Some applications, such as games and simulations, require the use of randomly generated numbers.

For example:

- You want to flip a coin, so the result could be two situations, head or tail
- You want to roll a dice, so the result could be 6 situations: 1 to 6
- You want to have hide a secret number between 1 to 10, and ask the user to guess it, so the result could be 10 situations: 1 to 10 For all these tasks we have to generate something random.

Java Math.random()

- It is a built-in method that belongs to java’s java.lang.Math class that is used to generate a random value of double data type.
- The random() method returns a random value that is greater than or equal to 0.0 and less than 1.0.

```
System.out.println(Math.random());  // 0.45950063688194265
// second random value
System.out.println(Math.random());  // 0.3388581014886102
// third random value
System.out.println(Math.random());  // 0.8002849308960158
```

How to get Specific Range of Values using Math.random()?

- Let’s assume we want to generate a random integer between 0 and 10, can we do this?
- **Generate Random Number Between 0 and 10**

```
int randval = (int) (Math.random() * 11);
```

- Every time we execute this program, we will get a different random number between the specified range. In the above example, we have seen that we can specify the range of values, but the initial value is zero.

All random number generation follows these basic steps:

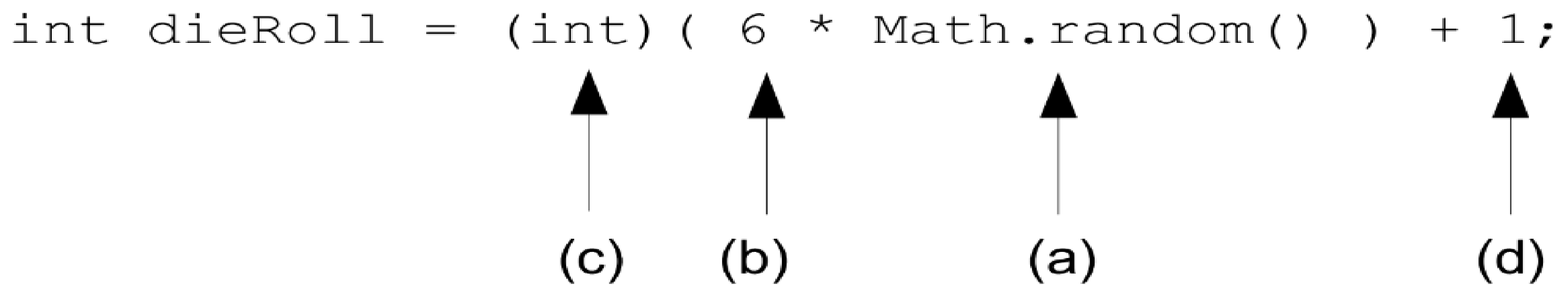
(a) generate a random number between 0.00000 and 0.99999,

(b) multiply by the number of possible results we desire(possible outcomes),

(c) optional: cast the result as an integer (if we want integer values)

(d) add (or subtract) some number to shift our random numbers to the desired range.

To simulate the throwing of a fair die, which is a six-sided die with an equal probability of getting a 1, 2, 3, 4, 5, or 6.



```
int dieRoll = (int)( 6 * Math.random() ) + 1;
```

Create a program that generates random numbers between 200 to 400

```
// The Math.random() method will generate a random number in the range
between the initial value as (initial value + (final value-1)).
    int min = 200;
    int max = 400;
//Generate random double value from 200 to 400
   System.out.println("Random value of type double between "+min+" to "+max+ ":");
   double a = Math.random()*(max-min+1)+min;
   System.out.println(a);
//Generate random int value from 200 to 400
   System.out.println("Random value of type int between "+min+" to "+max+ ":");
   int b = (int)(Math.random()*(max-min+1)+min);
   System.out.println(b);
}
```

Hands on

- Write a java program to calculate the volume of a sphere. Prompt the user to enter the diameter of a sphere. The diameter is twice as long as the radius, so calculate and store the radius in an appropriately named variable.
- The formula for the volume of a sphere is: $V = \frac{4}{3}\pi r^3$
- Convert the formula to Java code and add a line which calculates and stores the value of volume in an appropriately named variable. Use Math.PI for π and Math.pow to cube the radius.

```
public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        //prompt the user to insert a diameter
        System.out.print("Enter the diameter of a sphere: ");
        double diameter = input.nextDouble();
        // calculate the radius
        double radius = diameter/2;
        // calculate the volume
        double volume = 4/3 * Math.PI * Math.pow(radius, 3);
        System.out.printf("The radius of a sphere is: %.2f%nand the
                        volume is: %.2f%n ", radius, volume);


    }
```