# The Wrapper and Math Classes

The example file uses two types of Wrapper classes, the Integer class and the Double class. Both of these wrapper classes can be used to “wrap” a value of the respective type (integer or double) in an object.

Just as was done with the Object class and String class, a comparison can be made between the equals, “==”, operator and the Object class’ equals() method.

In the Integer and Double classes, the equals() method returns true if the respective int or double values are the same as that in the object.

1. **Explain what you would expect the output of the first two println() statements to be if the integer variable “distance” was equal to 10 centimeters.**

The random() method, part of the Math class, returns a double between 0 and 1. In the example file, because distance data is in integer form, it is cast to an integer.

1. **The range of the random() method in the example file is scaled to be from 0 to 20. How would you make the values range from 2 to 28? Write the code to achieve this.**

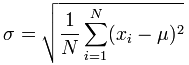
Other methods available in the Math class include the absolute value method, abs(), the square root method, sqrt(), and the power method, pow().

Add logic to the program, before the random number is picked, to get two distance readings with a wait time of 1 second in between the reads. The waitTime() method takes as input milliseconds. There are 1000 milliseconds in 1 second.

1. **Store the difference between the two values in a new integer variable and ensure the difference is the magnitude of the difference.**

In statistics, the standard deviation is a measurement of how spread out the numbers are. We can implement this in Java with the help of the Math class.

The equation for the standard deviation is:



Where sigma is the standard deviation, N is the number of samples, xi is the ith value from the set of values, and mu is the arithmetic mean of the set

Start by commenting out the while loop with a multi-line comment enclosing it in /\* and \*/.

/\*

while (distance < x)

{

distance = robot.getUSDistanceCMFiltered();

control.waitTime(100);

}

\*/

Underneath, add three double variables distance1, distance2, and distance3. On the line of each variable definition, add equals to robot.getUSDistanceCMFiltered(); like in the example file. Even though the distance readings are integers, because we will be performing calculations with these variables involving floating point precision, we will have them as doubles.

Between each reading add a 1 second delay using the waitTime() method, which is part of the “control” object as can be seen in the example.

Next, calculate the average by summing the three distance reading variables and dividing the sum by 3.

1. **Now implement a one line form of the standard deviation equation provided above using the Math class, the three distance variables, the number of samples (3), and the average distance.**

Lastly, you can use a println() method to display the end result and run the program. There are data structures called arrays that you will learn about in later lessons that would have made this process easier.