



Computer Science Virtual Learning

HS Computer Science A

April 17th, 2020



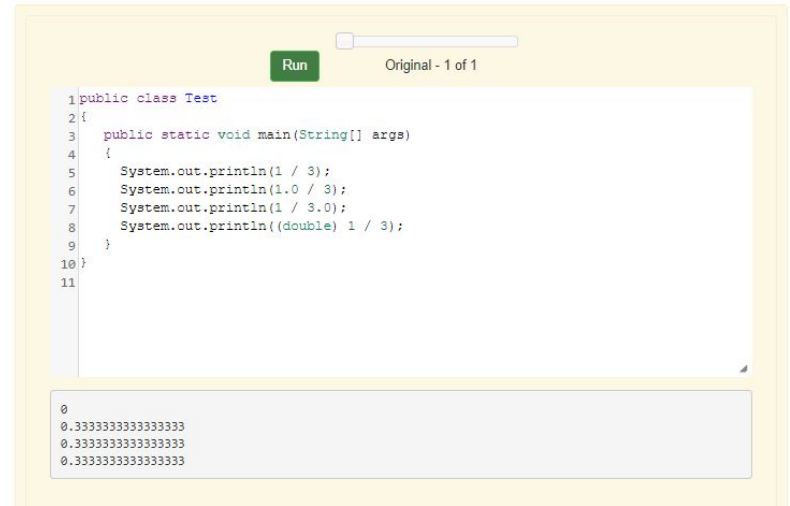
Lesson: Casting Variables

Objective/Learning Target:

Understanding how and why we cast different variables in
Java

Casting Variables

Java assumes that if you are doing division with integers that you want an integer result and it will throw away any fractional part (part after the decimal point). But, if you use a mixture of integers (int) and floating point (double) numbers Java will assume that you want a floating point result. If you have integers and you want a floating point result from some mathematical operation **cast** one of the integers to a double using (double) as shown above. By **casting** we don't mean something to do with fishing, but it is a similar idea to casting a pot in clay. In Java when you cast you are changing the "shape" (or type) of the variable to the right of the cast to the specified type.

A screenshot of a Java IDE window titled "Original - 1 of 1". It contains a code editor with the following code:

```
1 public class Test
2 {
3     public static void main(String[] args)
4     {
5         System.out.println(1 / 3);
6         System.out.println(1.0 / 3);
7         System.out.println(1 / 3.0);
8         System.out.println((double) 1 / 3);
9     }
10 }
11
```

Below the code editor is an output window showing the results of the program execution:

```
0
0.3333333333333333
0.3333333333333333
0.3333333333333333
```

Is the result of 1.0 divided by 3 what you expected? Java limits the number of digits you can save for any `double` number to about 14-15 digits. You should be aware that the accuracy of any calculation on a computer is limited by the fact that computers can only hold a limited number of digits.



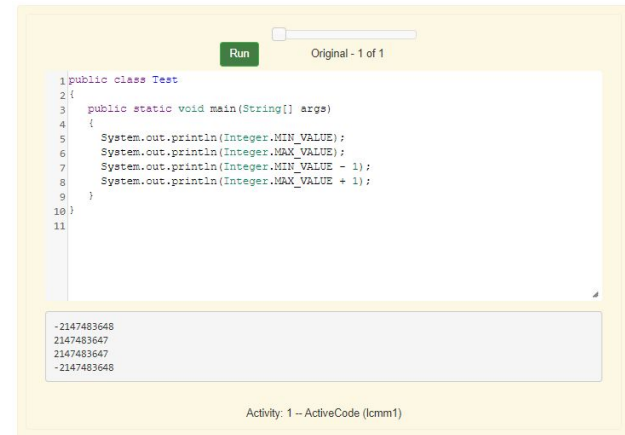
Check Your Understanding

Answer the following questions to check your understanding of the material in the previous slides

1. True or False: Java rounds up automatically when you do integer division
 - a. True
 - b. False
2. True or false: Casting Always results in a double type
 - a. True
 - b. False
3. Which of the following returns the correct average when 3 values had been added to an integer total?
 - a. `(double) (total / 3);`
 - b. `total / 3;`
 - c. `(double) total / 3;`

Integer Min and Max

The `int` type in Java can be used to represent any whole number from -2147483648 to 2147483647. Why those numbers? Integers in Java are represented in 2's complement binary and each integer gets 32 bits of space. In 32 bits of space with one bit used to represent the sign you can represent that many values. Why is there one more negative number than positive number? It is because 0 is considered a positive number.

A screenshot of a Java IDE window titled "Original - 1 of 1". It shows a code editor with the following code:

```
1 public class Test
2 {
3     public static void main(String[] args)
4     {
5         System.out.println(Integer.MIN_VALUE);
6         System.out.println(Integer.MAX_VALUE);
7         System.out.println(Integer.MIN_VALUE - 1);
8         System.out.println(Integer.MAX_VALUE + 1);
9     }
10 }
11
```

Below the code editor is an output window showing the results of the program execution:

```
-2147483648
2147483647
2147483647
-2147483648
```

At the bottom of the IDE window, it says "Activity: 1 -- ActiveCode (lcm11)".

What do the last two lines print out? Did this surprise you? Java will actually return the maximum integer value if you try to subtract one from the minimum value. This is called **underflow**. And, Java will return the minimum integer value if you try to add one to the maximum. This is called **overflow**. It is similar to how odometers work. When would you ever use `Integer.MIN_VALUE` or `Integer.MAX_VALUE`? They are handy if you want to initialize a variable to the smallest possible value and then search a sequence of values for a larger value.



Random Numbers

Games would be boring if the same thing happened each time you played the game. Games often use random numbers to generate different possibilities. You need to know how to use the `Math.random()` method to generate a random number. There are lots of mathematical methods that you might want to use in your programs like `Math.abs` (absolute value). These methods are in the `Math` class and are **static (class)** methods so that you can call them by just using `ClassName.methodName`.

****Note:** **Class** or **static** methods are in the object that defines the class (an object of a class named `Class`) and can be accessed directly from the class. You do not need to create an object of the class to use them.

Random Numbers

The `Math.random()` method returns a number greater than or equal to 0.0, and less than 1.0. Try out the following code. Run it several times to see what it prints each time.

A screenshot of a Java IDE window. At the top, there is a green "Run" button and a progress indicator showing "Original - 1 of 1". Below this is a code editor with the following Java code:

```
1 public class Test3
2 {
3     public static void main(String[] args)
4     {
5         System.out.println(Math.random());
6         System.out.println(Math.random());
7     }
8 }
9
```

At the bottom of the IDE, there is an output window displaying the results of the code execution:

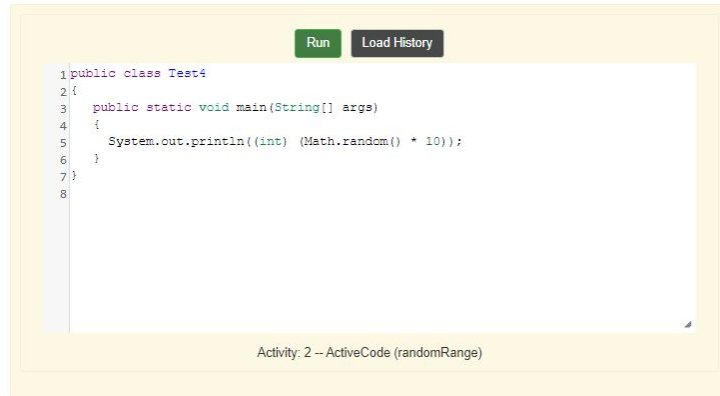
```
0.24075916580974477
0.6858132203655835
```

****Note:** Remember that a casting a double value to integer (`int`) will throw away any values after the decimal point.

You can use `Math.random` and a cast to integer to return a random number between some starting and ending value. The code below will return a random number from 0 to 9.

Check Your Understanding

1. Go to: <https://runestone.academy/runestone/books/published/apcsareview/VariableBasics/randomNumbers.html>
 - a. Run the code below several times to see how the value changes each time

A screenshot of a code editor interface. At the top, there are two buttons: "Run" (green) and "Load History" (grey). Below the buttons is a text area containing the following Java code:

```
1 public class Test4
2 {
3     public static void main(String[] args)
4     {
5         System.out.println((int) (Math.random() * 10));
6     }
7 }
8
```

At the bottom of the editor, there is a status bar that reads "Activity: 2 -- ActiveCode (randomRange)".

- b. How could you change the code above to return a random number from 1 to 10? Modify the code above and see if your answer is correct.



Check Your Understanding

2. Which of the following would be true about 40% of the time?

- a. `Math.random() < 0.4`
- b. `Math.random() > 0.4`
- c. `Math.random() == 0.4`

3. Which of the following would return a random number from 1 to 5 inclusive?

- a. `((int) (Math.random() * 5))`
- b. `((int) (Math.random() * 6))`
- c. `((int) Math.random() * 5) + 1`

4. Which of the following would return a random number from 0 to 10 inclusive?

- a. `((int) (Math.random() * 10))`
- b. `((int) (Math.random() * 11))`
- c. `((int) Math.random() * 10) + 1`

5. Which of the following would be true about 75% of the time?

- a. `Math.random() < 0.25`
- b. `Math.random() > 0.25`
- c. `Math.random() == 0.25`



For More Resources and to Check Answers

Go to: <https://runestone.academy/runestone/books/published/apcsareview/VariableBasics/casting.html>

<https://runestone.academy/runestone/books/published/apcsareview/VariableBasics/minAndMax.html>

<https://runestone.academy/runestone/books/published/apcsareview/VariableBasics/randomNumbers.html>