ACTIVITY **3.4.3**

Computations in Java

INTRODUCTION

Recall from algebra that a polynomial, such as $x^2 + 4x + 4$, has one or more roots, or x-intercepts. If a polynomial does not have integer roots, or if it is a higher order polynomial, the roots are difficult to find. Using a computer science algorithm, you will determine whether an estimate of the real roots of a polynomial are "close enough." You will also learn an algorithm for rounding integers, how to represent very small or very large numbers, and how to represent integers in different bases.

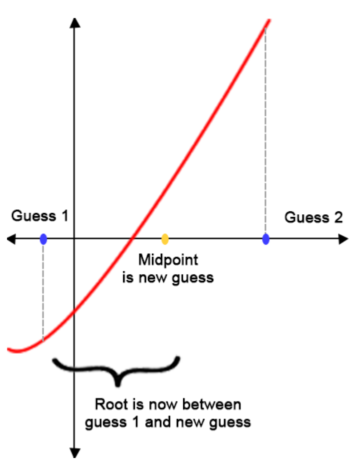
Materials

- Computer with Android™ Studio
- Android™ tablet and a USB cable, or device emulator

Procedure

Part I: Estimating with Epsilon

1. Open *`3.4.3Computations`* as directed by your teacher.
2. Open *WhiteboardAppComputationsActivity.java* and become familiar with the algorithm in the `bisectingPolynomials(…)` method that finds a root (the x-intercept where y is zero) of a polynomial using the bisection method. The visual representation of the first step of the algorithm is shown below.



Subsequent steps determine guesses that are closer and closer to the root.

3. In `bisectingPolynomials(…)`, find the statements for calculating the y values of the polynomial and write the equations in algebraic form, similar to: $y = x^2 + 4x + 4$

4. Find the statement containing `(Math.abs(y1 - y2))`. It compares this value to an **epsilon value**.

   What does the `while` statement accomplish with the epsilon value?

Floating Point Rounding Errors

- Computers cannot accurately store many numbers (such as 1/3)
- Round-off errors
- Use **epsilon**: a “close enough” value
- Needs of the application determine precision

Computer Science A

Algorithm Using Epsilon

- Epsilon is a very small amount
- Used to compare two values such as 2.734 and 2.735
- Considered “equal” with epsilon value of .01
- In pseudocode:

  if |value1 – value2| <= epsilon

  values are equal

Computer Science A

In mathematics, epsilon is a term representing a very small quantity. In computer science, it represents the difference between two values and how accurate you want an equality to be. For example, Pi can be represented as 3.14159. If you choose accuracy to 4 decimal places, the epsilon value would be .0001 and the values 3.14159 and 3.1416 would be considered close enough to be equal: The difference between them is .00001 which is less than .0001.

As another example, with an epsilon value of .01, the numbers 2.734 and 2.735 can be considered “equal.” Notice, rounding does work to determine this type of equality.

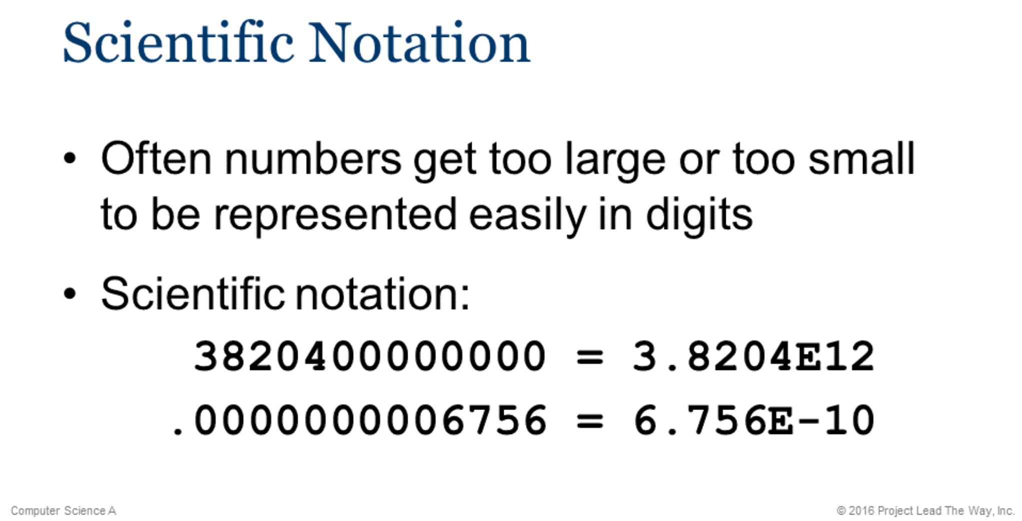
In general, you compare the “real” difference between two values to the epsilon value. If the real difference is less than or equal to epsilon, they can be considered equal. Or, in other words, if the real difference is greater than epsilon, they are not equal.

5. Add output to display `(Math.abs(y1 - y2))` at each iteration and when the root is found. Run the app.

   What is the difference between `y1` and `y2` when the root is found:

   a. Using scientific notation to three decimal places?

   b. Using decimal notation to eight decimal places?



6. What value of epsilon would you use to determine whether two numbers were equal to within four decimal places? Write your answer in decimal and scientific notation.

Part II: Another Epsilon

7. Comment out the call to the `bisectingPolynomials(…)` method and uncomment the call to `squareRootSquared(…)`.

8. Run the app. Explain why the square root of 2 squared does not calculate to 2.

9. Use an epsilon value to validate that the numbers are equal to four decimal places and test your app. What value of epsilon did you use?

   The numbers are “close enough“ to be equal, but they do not *appear* equal.

Rounding Algorithm

- Cast a floating point or double value to an integer to get a truncated integer value:
  - `(int) (1.3)` results in a value of 1
  - `(int) (2.9)` results in a value of 2
- A **rounding algorithm** uses a cast to determine nearest integer
- Add .5 to each number and truncate

```
34.06 + .5 = 34.56 → (int)34.56 = 34
1.875 + 2.5 = 2.375 → (int)2.375 = 2
```

Computer Science A

10. To improve the output, implement the `roundToInt(double d)` method and use it in the `textOut.append` statement. Test your app.
11. Create an overloaded method to do the same rounding algorithm for floating point numbers.
12. In your own words, explain how the algorithms in `roundtoInt` work. Use examples to support your explanation.

Part III: Unplugged Hex

Computers represent all information using the **binary number system**, and computer scientists use the **hexadecimal number system** to represent values precisely.

binary
The base-2 number system consisting of digits 0 and 1.

hexadecimal
The base-16 number system consisting of digits 0–9 and letters A, B, C, D, E, and F.

Binary and Hexadecimal (Hex) Representation

- Computers represent all data using on/off state of electricity
- 0, 1: digits of the binary number system
  - For example, decimal 7 is 00110111 in binary
- Difficult/impossible for humans
- Hexadecimal (base-16) easier to represent "native" value in a computer
- Digits: `0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F`

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Computers are built around the concept of electricity, specifically, the on/off state of electrical flow. Therefore, all data and information stored in a computer is represented in the binary number system, a base-2 number system consisting of 0s (no electrical flow) and 1s (electrical flow).

Representing information in binary, such as 00110111 for the number 7, is extremely time consuming and probably impossible for humans to quickly write/interpret.

Computer scientists use base-16 representation called hexadecimal to represent values precisely. Base-16 means that instead of using the base-10 digits 1 through 9, place values in base-16 are represented using 16 different "digits". More on these 16 hexadecimal digits in a few slides...

Review: Decimal Representation

- Write 7408 in decimal place values (base-10):

Place	Thousands (10^3)	Hundreds (10^2)	Tens (10^1)	Ones (10^0)
Digit	7	4	0	8
Exponential form	$7*10^3$	$4*10^2$	$0*10^1$	$8*10^0$
Calc. powers	7*1000	4*100	0*10	8*1
Do the Math	7000	400	0	8
	7000 + 400 + 0 + 8 = 7408			

To understand the base-16 number system, recall how the base-10 number system works:

- The digit 8 is in the ones place, also represented as the 100 place. The 0 is in the tens place, or the 101 place, the 4 is in the 102 place, and the 7 is in the 103 place.
- To calculate the value, multiply each place value by the corresponding 10n.
- Do the math and sum up to get your value.
- To summarize, a digit in an nth place represents the value of the digit * 10n.

7*1003 = 7000
4*102 = 400

0*101 = 0

8*100 = 8

= 7408

Converting from Hex to Decimal

- Similar to previous algorithm:
  - Write hex digits in exponential form
  - (Extra step) Convert hex digits to decimal
  - Calculate the powers
  - Sum to get decimal value

- Practice with hex value B61F

The general algorithm is similar to the previous steps of decimal representation. Examples on the following slides will help solidify the algorithm.

Convert $B61F_{16}$ to decimal

Place	(16^3)	(16^2)	(16^1)	(16^0)
Digit	B	6	1	F
Exponential form	$B*16^3$	$6*16^2$	$1*16^1$	$F*16^0$
Use hex values	$11*16^3$	$6*16^2$	$1* 16^1$	$1*16^0$
Calc. powers	11*4096	6*256	1*16	15*1
Do the Math	45056 +	1536 +	16 +	15 = 46623

Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

$B61F_{16} = 46623_{10}$

Example: convert B61F16 to decimal.

- Write digits in base-16 place values.
- Write the exponential equation:
- B*163 + 6*162 + 1*161 + F*160
- Replace hex digits with decimal value:
- 11*163 + 6*162 + 1*161 + 15*160
- Calculate powers:

11*4096 + 6*256 + 1*16 + 15*1

Do the math:
45056 + 1536 + 16 + 15 = 46623

13 Review the slides and follow the steps to convert the following three hexadecimal (hex) values to their decimal equivalent.

Hex	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

a. Convert `9B`$_{16}$ to decimal.

Hex	Instruction	Work			
9B	Write number in hex place values	16^2 (256)	16^1 (16)	16^0 (1)	16^2 (256)
	Write exponential equation				
	Replace hex digits with decimal value				
	Do the Math				
	Decimal value:				

b. Convert `BAD`$_{16}$ to decimal.

Hex	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Hex	Instruction	Work			
BAD	Write number in hex place values	16^3	16^2 (256)	16^1 (16)	16^0 (1)
	Write exponential equation				
	Replace hex digits with decimal value				
	Do the Math				
	Decimal value:				

c. Convert $770F_{16}$ to decimal.

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Hex	Instruction	Work
770F	Write number in hex place values	16^3 (4096) | 16^2 (256) | 16^1 (16) | 16^0 (1)
	Write exponential equation	
	Replace hex digits with decimal value	
	Do the Math	
	Decimal value:	

Part IV: Unplugged Decimals

14 Review the slides and follow the steps to convert the following decimal values to hexadecimal.

a. Convert 42_{10} to hex.

Decimal	Instruction	Work
42	Divide by *largest* 16^n; write quotient in appropriate 16^n place in sub-table at the bottom of the Work column	
	State the remainder; use in next iteration until < 16	
	Fill in the table each time you iterate:	16^2 (256) | 16^1 (16) | 16^0 (1)
Rewrite with hex digits for **hex value**:		

b.

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Convert 961_{10} to hex.

Decimal	Instruction	Work
961	Divide by largest 16^n; write quotient in appropriate 16^n place in sub-table at the bottom of the Work column	
	State the remainder; use in next iteration until < 16	
	Divide by largest 16^n; write quotient in 16^n place	
	State the remainder; use in next iteration until < 16	
	Fill in the table each time you iterate:	16^2 (256) | 16^1 (16) | 16^0 (1)
Rewrite with hex digits for **hex value**:		

c.

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Convert decimal 3589_{10} to hex. (If a remainder is too small to divide by the next 16^n, a 0 should go in the corresponding n^{th} place.)

Decimal	Instruction	Work
3589	Divide by largest 16^n; write quotient in appropriate 16^n place in sub-table at the bottom of the Work column	
	State the remainder; use in next iteration until < 16	
	Divide by largest 16^n; write quotient in 16^n place	
	State the remainder; use in next iteration until < 16	
	Fill in the table each time you iterate until complete	16^2 (256) | 16^1 (16) | 16^0 (1)
Rewrite with hex digits for **hex value**:		

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Reverse: Converting Decimal to Hex

- General algorithm: Repeatedly divide by the largest 16^n
  - Record quotients in 16^n place value
  - Divide with remainders until you get a value less than 16
- Use hex digits in the final result
- Practice with decimal value 942

Computer Science A

Convert 942_{10} to Hex

Repeatedly divide value by largest 16^n		16^2	16^1	16^0
Divide value by 16^2	942 / 256 =	3		
	(remainder 167)			
Divide remainder by 16^1	167/16 =		10	
	(remainder 7)			
Remainder 7 is < 16, write in ones place, done dividing				7
Replace decimals with hex digits		3	10	7
		3	A	7

Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

$942_{10} = 3A7_{16}$

To convert from decimal to hexadecimal:

- Use long division to divide value by largest 16n.
- 924 / 256 = 3 remainder 167
- Write quotient 3 in correct hex place value.
- Use remainder for next division of 16n.

167 / 16 = 10 remainder 7

- Write quotient 10 in correct hex place value.
- Use remainder, but since 7 < 16, you are done dividing.
- Write remainder in "ones" place value (160).
- Replace decimals with hex digits.

Part V: Finite Bounds

15 Back in *`WhiteboardAppComputationsActivity.java`*, comment out the call to `squareRootSquared()` and uncomment `finiteBounds()`.

16 Run the app and explain why adding 1 to `i` results in the value you see.

Finite Bounds of Integers

- A single digit or character is stored in a 32-bit register
- For example, the number **`7`** is represented in 32-bit register as:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1

- The **number** of signed integers is 2^n-1, or **`4,294,967,294`**
- And ~~T~~that translates to integer **values** from **`-2,147,483,648`** to **`2,147,483,647`**
- Larger or smaller than this: **integer overflow**

Finite Bounds of Integers

- Different languages handle **integer overflow** differently
- In Java (and C), the behavior is undefined
- Therefore, different compilers may show different results when an integer overflows

Finite Bounds of Integers

- One common behavior: the value *rolls over*
- In binary, most significant bit represents sign (+ or –)
  - `2,147,483,647` is:

```
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

  - `-2,147,483,648` is:

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- Add 1 to `2,147,483,647`

```
01111111111111111111111111111111
                              +1
--------------------------------
10000000000000000000000000000000
```

…and the rollover occurs, resulting in -2,147,483,648

Part VI: Computational Risks

17 The search and sort algorithms you learned are powerful tools, and on large computer systems, they can provide enormous functionality. But computational speed and powerful computers can give criminals the upper hand when trying to break into secure computer systems. This is especially true when security measures that should protect those systems are weak, passwords are easy to guess, and data is unencrypted.

a. Visit **World's Biggest Data Breaches**. Near the upper right-hand corner of the data visualization, filter **Method of Leak** by checking **hacked**.

b. Select one of the larger breaches, and the bubble will expand. Click on the **Read a bit more** link and summarize, in your own words, how the breach occurred.

c. Share your findings with another team (one who reported on a different breach) and describe their findings.

CONCLUSION

1. Explain how an epsilon value could be used in the following scenarios and provide an estimate for a good epsilon value:
    a. To determine that `Math.PI` "equals" 3.14.
    b. To build a fence around a dog run measuring 8 feet by 24 feet.
2. Describe a situation where the finite bounds of an integer could pose a problem for an app.