Unit 10 → Recursion

Recursion

- Definition
  - An iterative proccess where a method calls itself
  - Doing something repeatedly until forced out
- What does Recursion do?
  - Breaks a problem down into simpler sub-problems of the same for until it becomes easier to solve
  - Example
    - sum(10) = 10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0
      - sum(10) = 10 + sum(9)
- Base Case
  - A point where something cannot be simplified anymore
  - A way to exit a loop; get out
  - Not having a base case causes an infinite loop
  - Example

```
if (n==0)
      return 0;
```

or

```
if (n < 0)
      return 0;
```

- Formula
  - sum(n) = n + sum(n - 1)
    - Could be used as a return line
      - Returning a line that calls itself will cause the method to loop until it hits the base case
    - Similar to the arithmetic recursive formula
      - $a_n = a_{n-1} + d$
- Properties
  - Each recursive call has its own set of variables and parameters

- Implementations and traversing
  - Strings
  - Arrays
  - ArrayLists

Recursive Searching

- Review (from Unit 7)
    - Linear search checks each value in order until it reaches the end or desired value
- Binary Searches
    - Binary searches look for a midpoint and checks if value is greater or less than
    - ARRAY MUST BE SORTED
    - Very efficient because it eliminates half of the values
    - Walkthrough
        1) Test midpoint
        2) Eliminate half of population
        3) Find midpoint of remainder
        4) Repeat until target is found
    - Linear vs. Binary
        - Linear
            - Not efficient
            - Sorted or unsorted arrays
            - Easy to code
        - Binary
            - Very efficient
            - Sorted arrays ONLY
            - Slightly harder to code
        - Why?
            - Both offer the same efficiency from a memory and speed perspective
            - Personal preference and money
    - Formula
        - $number\ of\ iterations = log_2 array\ size$
            - $number\ of\ iterations = \frac{log(array\ length)}{log(2)}$
            - $array\ size = 2^{number\ of\ iterations}$
        - Maximum number of iterations needed for a binary search
- Encapsulation → trapped inside of itself

Recursive Sorting

- Merge Sort
    - Divides a list into 2 pars: left side and right side, then repeats this process until there is one number on each side. Finally it recombines them in order
    - Uses a recursive algorithm
    - More efficient than insertion and selection sort
        - Takes less time
        - Can be somewhat complex
    - Does *NOT* have to be in order
        - If there is an odd number of elements, right array would have more
    - Base case

```
if (length < 2)
      return;
```

    - Helper method → a private method used to simplify you code