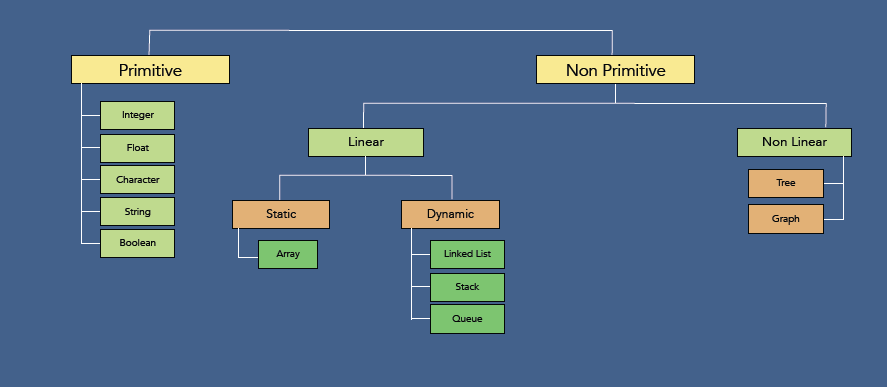
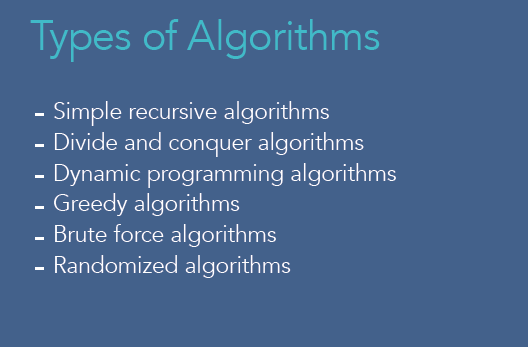
1. ****Types of Data Structures****

* **Primitive Data Structures are built in with in programming language**
* **Non Primitive Data Structures are defined by the developers**
* **Non Primitive data structures can be linear and non linear**
* **Linear data structures are divided as Static and Dynamic**

****

1. ****Types of Algorithms:****

* **Algorithms classified based on the problems they are trying to solve such as Sorting, searching algorithms etc or based on the problem solving approach.**
* **Here types of algorithms are classified based on the problem solving approach**

****

****Simple Recursive Algorithms:****

* **They are iterative algorithms**

****Divide and Conquer Algorithms:****

* **Divide the problem into smaller subproblems of same type and solve these problems recursively**
* **Combine the solutions to the subproblems into a solution to the original problem**

**Eg: Quick sort and Merge Sort**

****Dynamic Programming Algorithms:****

* **They work based on memorization, which means that algorithms remember the past results and use them to find new result**
* **Generally used for optimization problems**
* **Goal of these algorithms is to find the best solutions amongst multiple solutions**

****Greedy Algorithms:****

* **To find best solution and work well for optimization problems**
* **Works in phases**
* **At each phase, we take the best we can without worrying about future consequences**
* **Algorithm hopes that by choosing a local optimum solution at each step, we will end up with global optimal solution**

****Brute Force Algorithms:****

* **It Simply tries all possibilities until a satisfactory solution is found.**

****Randomized Algorithms:****

* **Algorithm uses a random number atleast once during the computation to make a decision**

**Eg: Quick sort algorithm**

1. ****Recursion****

****What is Recursion:****

* **It is a way of solving a problem where a function calls itself**
* **Performing same operation multiple times with different inputs**
* **In every step, we try smaller inputs to make the problem smaller**
* **Base condition is needed to stop the recursion, otherwise infinite loop will occur**
* **Base condition is the condition after which we wont do recursion.**

****Why do we need Recursion:****

* 1. **Recursive thinking is really important in programming and it helps you break down big problems into smaller ones and easier to use**
  2. **When to choose recursion?**

**‣ If you can divide the problem into similar sub problems**

**‣ Design an algorithm to compute nth…**

**‣ Write code to list the n…**

**‣ Implement a method to compute all.**

**‣ Practice**

* 1. **The prominent usage of recursion in data structures like trees and graphs.**
  2. **It is used in many algorithms (divide and conquer, greedy and dynamic programming)**

****Logic behind Recursion:****

* **Method calls itself**
* **Exit from infinite loop**

***static string recursionMethod(String[] parameters) {***

***if (exit from condition satisfied) {***

***return some value;***

***} else {***

***recursionMethod(modified parameters);***

***}***

***}***

****Stack and Methods:****

* **Stack works on LIFO principle, where last entered will be removed first**
* **In Stack push method is for insertion and pop method is there for removal**

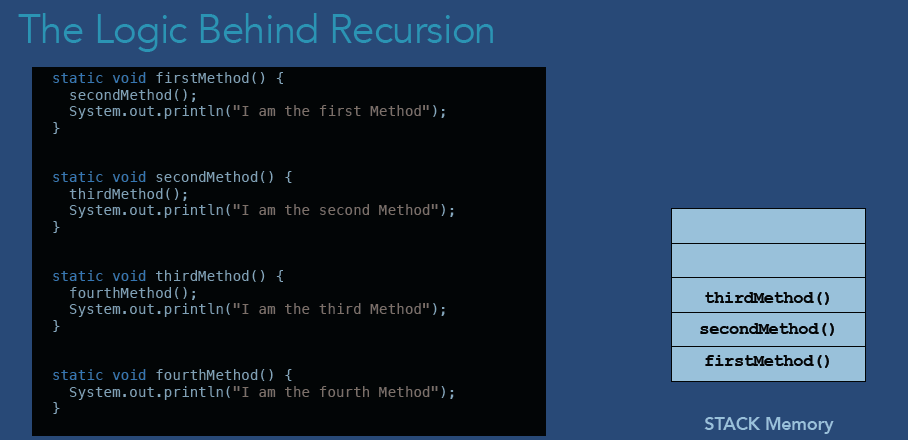
****How Normal Method Execution stored in Stack memory****

* **First method calls Second and second calls third and third calls fourth method**
* **Lets see what’s happening in stack when you call those methods**
* **Stack memory is maintained by the system for method invocation**

**To remember name of first method , system pushes first method name to stack memory**

**Similarly second and third method name will be stored in stack**

* **Since fourthMethod() is executing directly one line, without calling any method, System does not need to store method name for calling it back. Hence it will not be inserted in stack memory**
* **After executing fourth method, how does the system know that it needs to execute third method? Answer is: it will take it from stack memory. Thus Stacks LIFO principle ensures that methods are called in order. Once particular method is called, that method name will be removed from stack.**

****

****Recursion Method in Stack Memory:****