Stacks & Queues

Week 3 AI Inspire Fall 2019 @ PRC

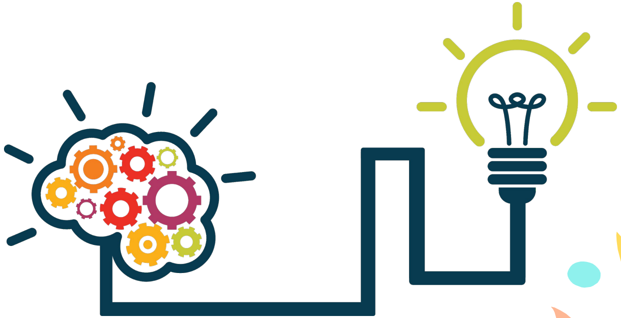
1.

Recursion

Intro to recursion, examples

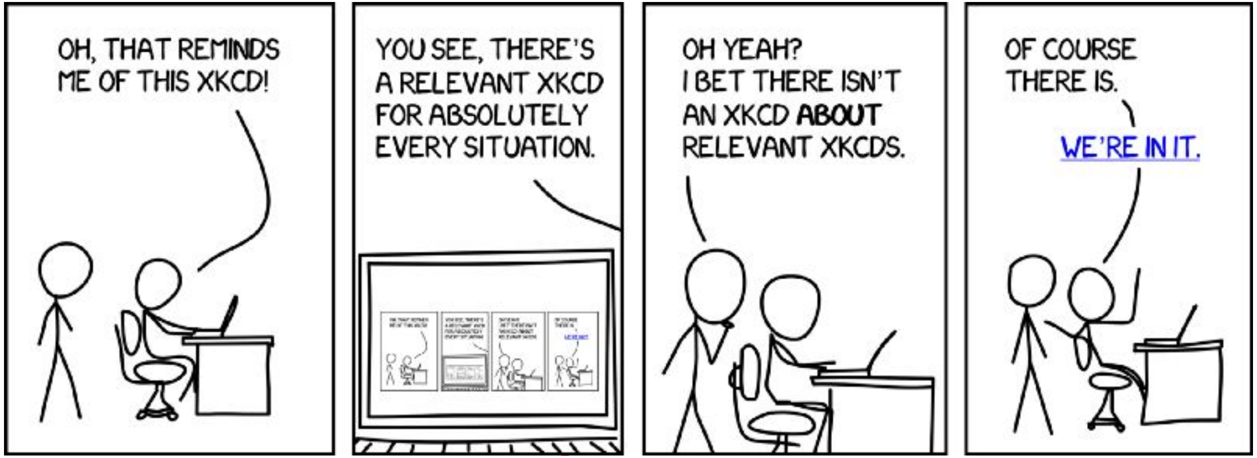
Recursion - What is it?

- Popular method of solving problems in theoretical CS
  - Used in many algos
- Method/function calls itself
- Function is defined in terms of itself
- Repetition
- More condensed way

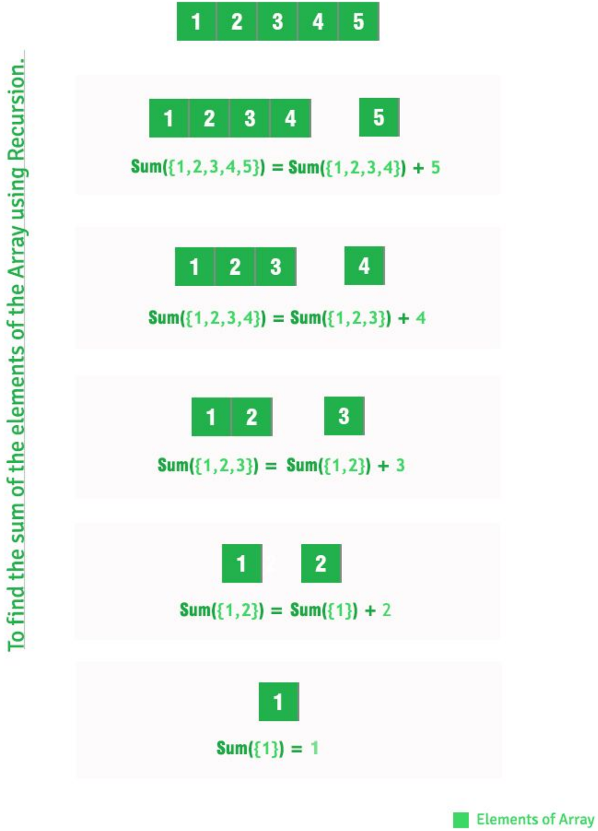


Recursion - Pseudocode

- Need to make sure there is a base case which stops the execution of algo
- Have recursive step
    - Recursion implemented


OH, THAT REMINDS ME OF THIS XKCD!
YOU SEE, THERE'S A RELEVANT XKCD FOR ABSOLUTELY EVERY SITUATION.
OH YEAH? I BET THERE ISN'T AN XKCD ABOUT RELEVANT XKCDS.
OF COURSE THERE IS.
WE'RE IN IT.

Example #1 - Sum of first n terms



To find the sum of the elements of the Array using Recursion.

Sum({1,2,3,4,5}) = Sum({1,2,3,4}) + 5

Sum({1,2,3,4}) = Sum({1,2,3}) + 4

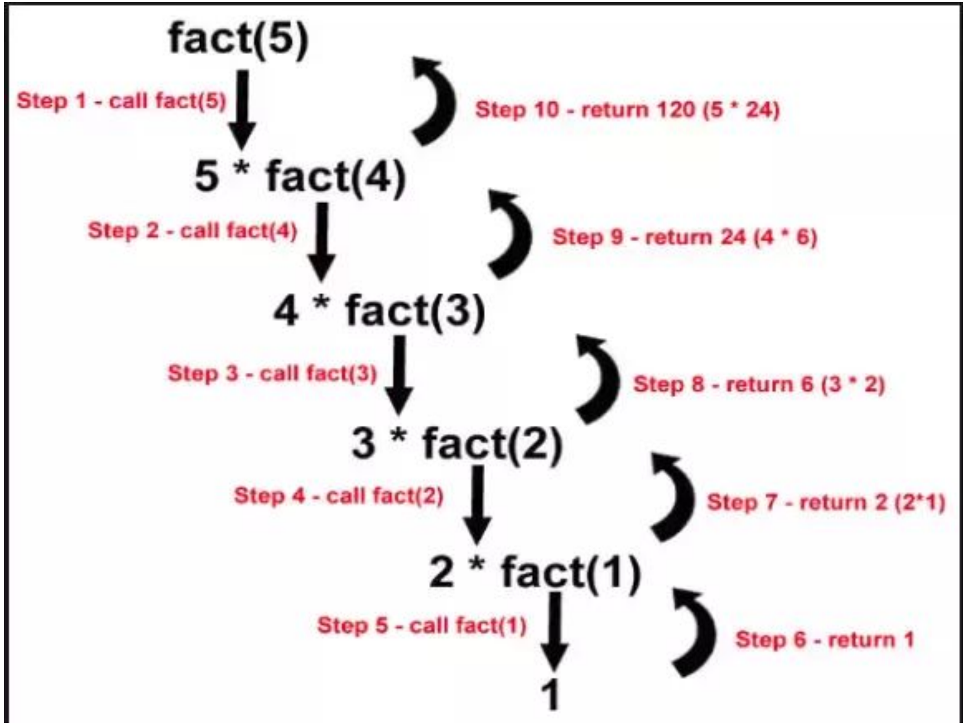
Sum({1,2,3}) = Sum({1,2}) + 3

Sum({1,2}) = Sum({1}) + 2

Sum({1}) = 1

🟩 Elements of Array

Example #2 - Factorial



fact(5)

Step 1 - call fact(5)

Step 10 - return 120 (5 * 24)

5 * fact(4)

Step 2 - call fact(4)

Step 9 - return 24 (4 * 6)

4 * fact(3)

Step 3 - call fact(3)

Step 8 - return 6 (3 * 2)

3 * fact(2)

Step 4 - call fact(2)

Step 7 - return 2 (2*1)

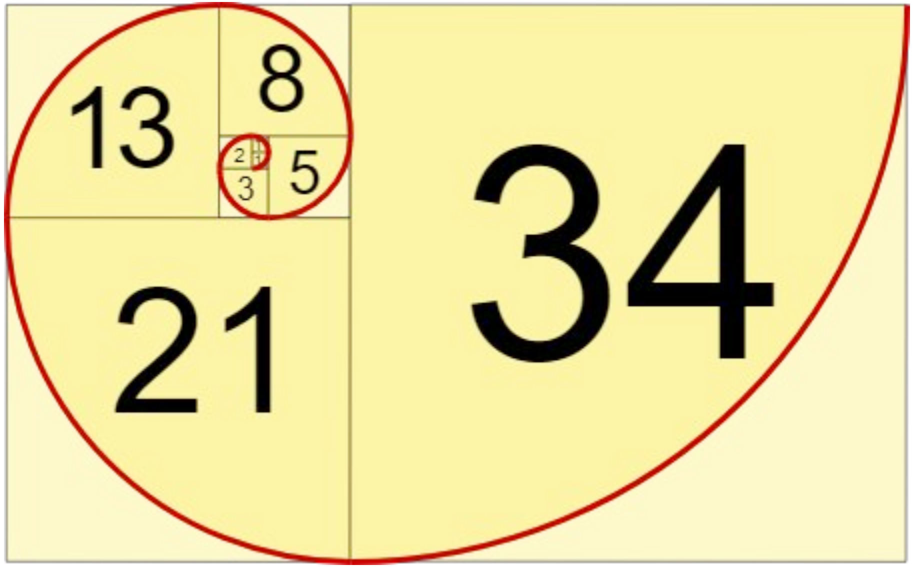
2 * fact(1)

Step 5 - call fact(1)

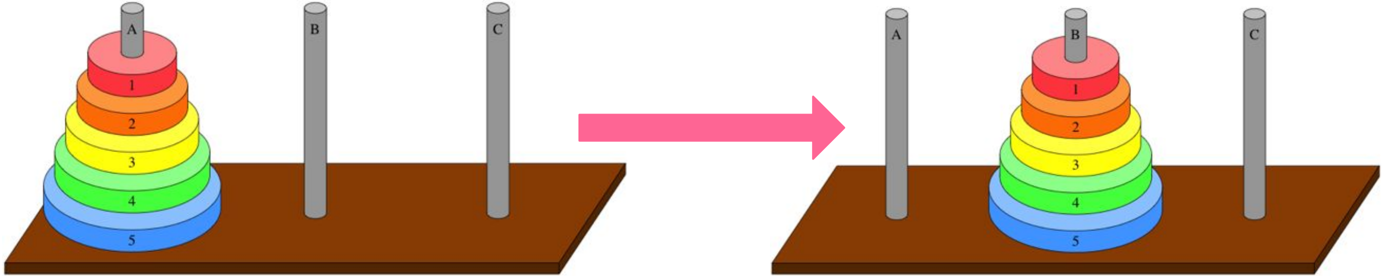
Step 6 - return 1

1

Example #3 - Fibonacci



Differences between Recursion and Iteration - Towers of Hanoi problem



Rules -

1. Move only 1 disk at a time (only top one)
2. No bigger disk can be on top of a smaller disk

Notation

Peg A = Source, Peg B = Destination, Peg C = spare

HW Task #1-

1) Figure out how to solve Towers of Hanoi problem
2) Develop recursive solution
3) Develop iterative solution

Difference between Recursion and Iteration?

Recursion -

- Conditional “if statement” decides termination of statement
- Infinite recursion = computer crashes
- Overhead of method calls
- Code is more compact

Iteration -

- Control variable’s value decides termination of statement (“for loop”) except for while loop
- Infinite iteration = CPU cycle consumed
- No overhead of method calls
- Code is larger

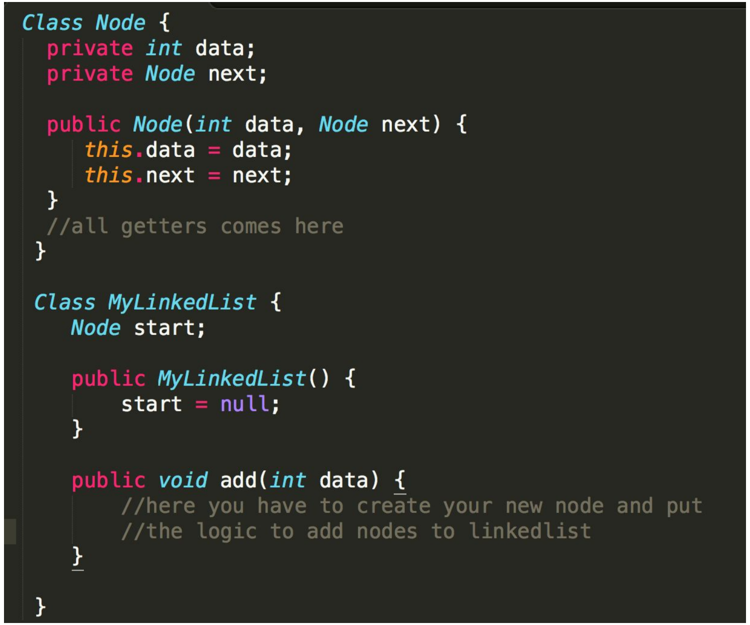
2.
Node Class

Intro to Node class, applications to Stacks & Queues, other algos

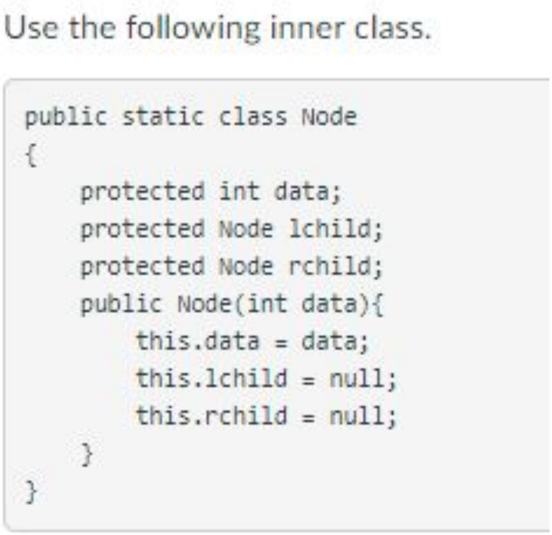
What is an inner class?

- Nested class
  - Declared inside of another bigger class
- Private inner class purpose
  - Cannot be accessed by other outside classes (other than class it resides in)
  - Outer class can easily use its methods
- Can declare node class as separate private class, but many times declared as private inner class

Node Inner Class



```
Class Node {
  private int data;
  private Node next;

  public Node(int data, Node next) {
     this.data = data;
     this.next = next;
  }
  //all getters comes here
}

 Class MyLinkedList {
    Node start;

    public MyLinkedList() {
        start = null;
    }

    public void add(int data) {
        //here you have to create your new node and put
        //the logic to add nodes to linkedlist
    }

 }
```

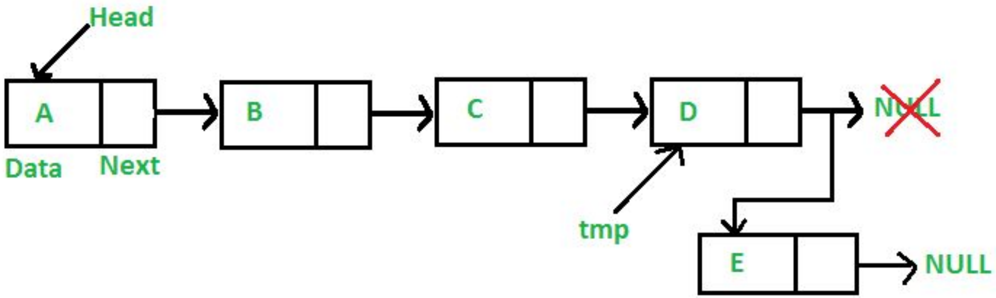


Use the following inner class.

```
public static class Node
{
     protected int data;
     protected Node lchild;
     protected Node rchild;
     public Node(int data){
          this.data = data;
          this.lchild = null;
          this.rchild = null;
     }
}
```

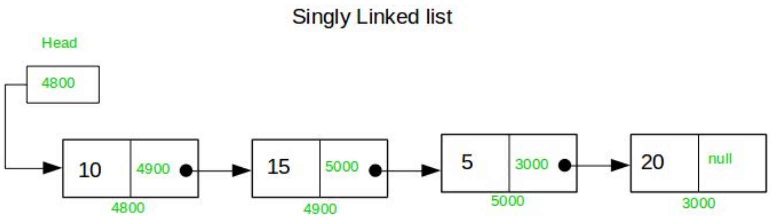
What is a node?

- Basic data structure unit
- Private inner class node
  - Data
  - Reference to next node
- Used in several data structures
- NOTE - Inner class is RECURSIVE



Where are nodes used?

- Linked lists
  - Singly linked, doubly linked, etc.
- Trees
- Graphs
- Implementing stacks
- And many more data structures + algos + applications


Singly Linked list

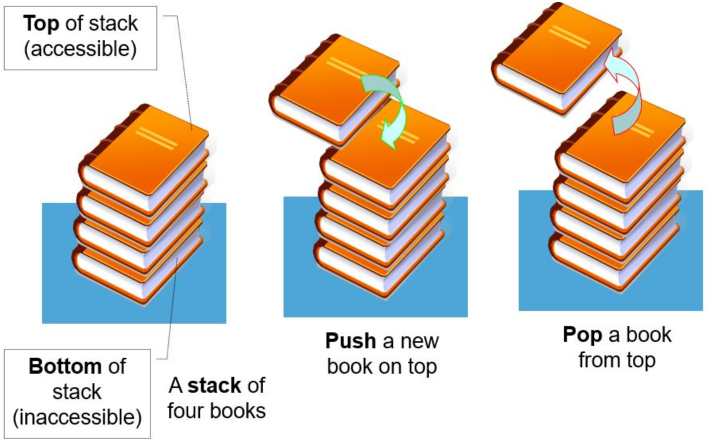
2.
Stacks

Intro to stacks, applications, implementation

What is a Stack?

- Key = LIFO
  - **Last In First Out**
- Operations
- Push, pop, peek
  - Java implementation
- Examples

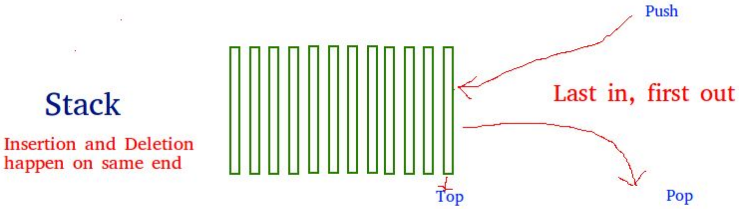
Stack of books


Top of stack (accessible)
Bottom of stack (inaccessible)
A **stack** of four books
Push a new book on top
Pop a book from top

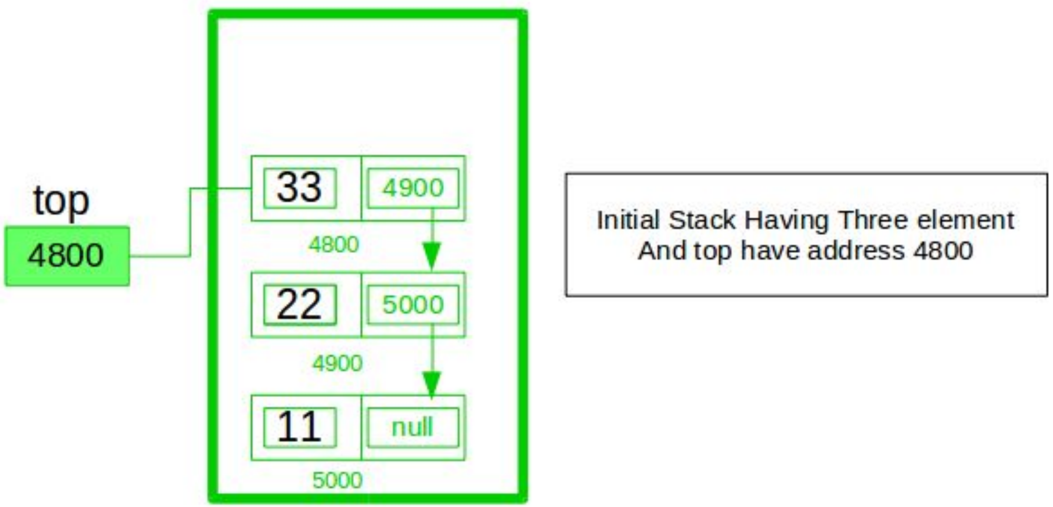
Stack Java API Operations

Operations

- Push()
  - Add new element to top
- Pop()
  - Return top element
- Peek()
  - Return top element
- Display()
  - Print out all elements



Stack Implementation with Linked List



Initial Stack Having Three element
And top have address 4800

Implement these 4 operations using what you've learned so far!!!

3.

Queues

Intro to queues, applications, implementation

What is a Queue?

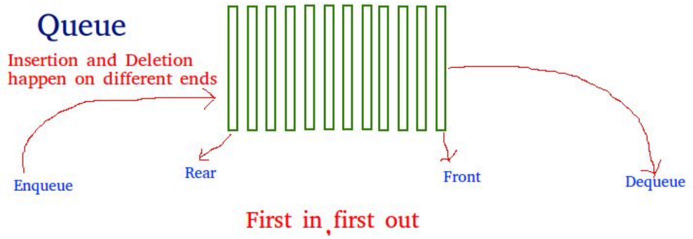
- Key = FIFO
  - **First In First Out**
- Operations
- Enqueue, dequeue, pee, display
  - Java implementation
- Examples

Queue of people in line

Queue Java API Operations

Operations

- Enqueue()
  - Add item to back
- Dequeue()
  - Remove item from front
- Peek()
  - Look at item from front
- Display()
  - Print all elements



HW Task #2 - Implement Queue operations using Linked List

HW Task #3 - Readings on more interesting data structure called deque (similar to stacks and queues but implementation is more advanced)

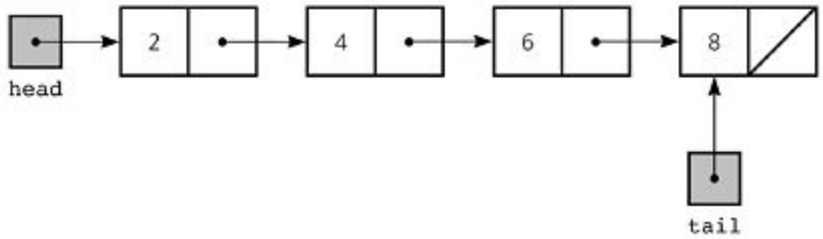
Brief Summary of concepts (not Java Implementation)- https://www.youtube.com/watch?v=6QS_Cup1YoI

4.
Iterators

Intro to iterators, applications, implementation

What are Iterators?

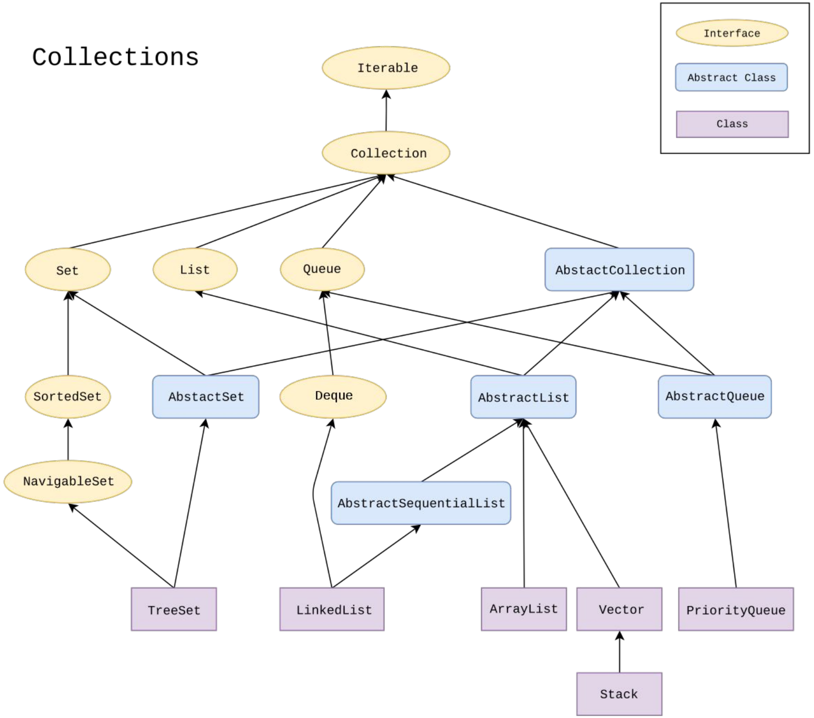
- Helps with traversing or iterating through list
  - Used in linked lists to display elements all the time



Iterator Java API

- Part of Java collections framework
  - Java collections
    - Set of classes and interfaces which implement collection data structures
      - Similar to a library
        - Interfaces define collections and classes implement them
- Java API for Iterator - https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html
  - Interface
    - Diff between interface and class in Java

Java Collections



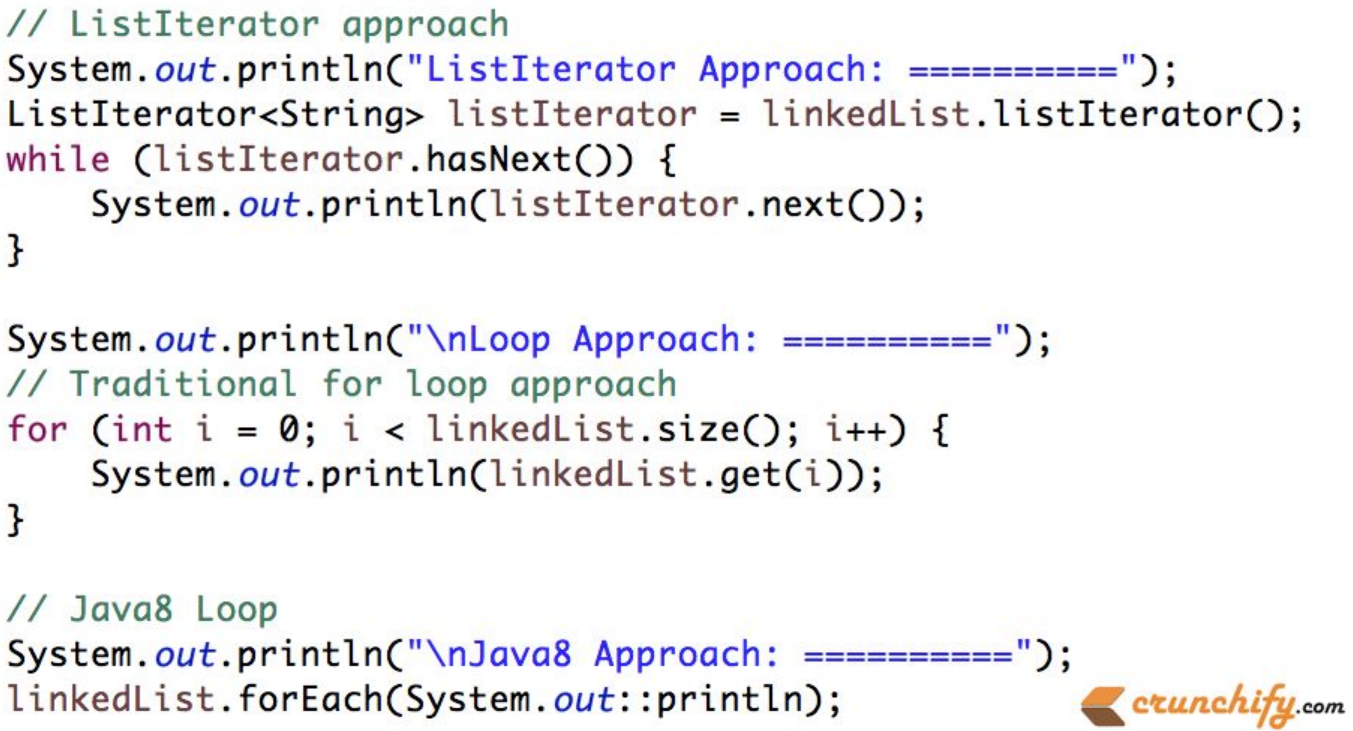
ListIterator Java API Interface

- Similar to iterator interface but has more functions
  - Can add elements
  - Has back link
- https://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html

Linked List Built in List Iterator

https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

- .listIterator(int index) method returns a list-iterator of elements in list at specified index
  - Able to traverse through elements



```
// ListIterator approach
System.out.println("ListIterator Approach: ==========");
ListIterator<String> listIterator = linkedList.listIterator();
while (listIterator.hasNext()) {
    System.out.println(listIterator.next());
}

System.out.println("\nLoop Approach: ==========");
// Traditional for loop approach
for (int i = 0; i < linkedList.size(); i++) {
    System.out.println(linkedList.get(i));
}

// Java8 Loop
System.out.println("\nJava8 Approach: ==========");
linkedList.forEach(System.out::println);
```

Hope you enjoyed this topic!