-----------------------------------------------------------------------------------------------------------------------------

**Design Notebook**

-----------------------------------------------------------------------------------------------------------------------------

**Max Santomauro**

**Step 1: Problem Statement**

This project involves implementing a binary search tree and a level-order traversal of the tree. The tree will be filled with parrots a when a level order traversal is performed the parrots will sing a song. The program has a Parrot class, a Binary Tree class, and a Tree Node class within the Binary Tree class.

Files used have the following value shown below.

**parroatsTest.txt**

15 Mojo Oh

38 Pollymorphic for

8 Charlie beautiful

12 Love skies

2 Tweety spacious

**parrots.txt**

93 Blue School's

200 Moe for

275 Charlie pencils,

45 Elvis out

150 Molly more

72 Kiwi No

81 Grace no

310 Hawk notebooks!

28 Izzy summer!

168 Jam crazy

15 Kookie no

3 Molly Have

125 Hula more

230 Ruko design

35 Boss more

130 Nutty see

400 Tweety in

42 Coco summer,

94 Yakky we'll

31 Pollymorphic great

401 Yeti Fall!

19 Love a

64 Cutie books,

235 Rocky you

Format of each line:

A close-up of words

AI-generated content may be incorrect.

**Step 2: Design Sketch**

**A diagram of a method

AI-generated content may be incorrect.**

**Step 3: Pseudocode**

* **Within the Main method**
  + Create two file reference variables that refer to the **parrotsTest.txt** and **parrots.txt** files
  + Open both files for reading with Scanner
  + Create instance object of BinaryTree class called **parrotsTest** to store parrots in
  + Read parrots from the parrotsTest.txt file
    - Use a while loop with a **hasNext** method as the condition
    - Use insert method from the **BinaryTree class**
  + Call the **levelOrder** method to display the parrot songs
  + Call the **visitLeaves** method to display the parrot names
  + Repeat the process for the parrots.txt file as was done with the parrotsTest.txt file
  + Make sure to close the scanners for both files
* **Outside the class where Main method resides** 
  + Create a **Parrot class** that represents one parrot and implements **Comparable<Parrot>**
    - In the private data field, create an int variable for **id** and two String variables for **name** and **songWord**.
    - Create a constructor and initialize all private data fields with incoming values read from the file.
    - Create getters for name and songWord
    - Create a **compareTo method** that compares parrots with their id's
      * Use -1, 1, and 0 as return values
  + Create a **BinaryTree class** to store elements in hierarchy
    - In the private data field, create a private instance variable called **root** of type TreeNode
    - Create a constructor and set private field variable root to null
    - Create boolean method called **insert** that takes in a Parrot object for adding new nodes (data values) to the tree while maintaining the tree's structure
      * Use if/else conditions with a while loop to transverse the tree to find the insertion point
      * Create an instance of TreeNode called **newNode** and insert a new Node
      * Make use of the compareTo method
    - Create a void method called **levelOrder** to do a level-order traversal of the tree
      * Create Queue called **queue** to keep track of the nodes in the next level down and use the **add** method to add the root
      * Use while loop with isEmpty() as the condition to add tempNode’s left and right children to the queue
        + Create an instance of TreeNode called tempNodeB and equal it to the queue array using the remove method.
        + Print out the song using the **sing** method
    - Create a public void method called **visitLeaves** that calls the private recursive method visitLeaves
      * Used to call into main since main because main does not have access to the root
      * Call private recursive method of visitLeaves
    - Create a private void method called **visitLeaves** for visiting leaf nodes in left to right order using recursive method
      * Use if condition to check if the current node is a leaf and print parrot name
      * Recursively visit the left subtree and then right subtree
    - Create a private inner class tree node called **TreeNode** that represents one node in the binary search tree.
      * Create a instance of Parrot called **value** and instances of the TreeNode called **left** and **right**
      * Create a constructor and initialize value variable to itself and the left and right variables to null