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

**Design Notebook**

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

**Max Santomauro**

**Step 1: Problem Statement**

This project involves the implementation of Java's predefined JCF stack class and a created generic class along with its own methods. The first part involves replacing all zeroes of array of integer values and the second part involves manually sorting two unsorted stacks and merging them into one sorted stacked based on values from files.

**The files are:**

|  |  |  |  |
| --- | --- | --- | --- |
| **integers1.txt** | **integers2.txt** | **strings1.txt** | **strings2.txt** |
| 26  81  22  45 | 20  9  33  21  30  48  68 | Dopey  Sleepy  Grumpy  Bashful  Happy  Sneezy  Doc | Saturn  Earth  Venus  Mars |

**Step 2: Design Sketch**

**A diagram of a process

AI-generated content may be incorrect.**

**Step 3: Pseudocode**

* **Within the Main method**
  + Create an integer array called **numbers** with values 0, 0, 4, 3, 0, 0, 2, 1, 0, 0.
  + Create a stack based on Java’s available JCF called **stack.**
    - **import java.util.Stack;**
  + Fill **stack** with the array elements
  + Replace the 0 values with 10 values by calling **replaceZerosWithTen** method and running stack through it.
  + Call the none-generic **printStock** method to display original stack with the 0 values replaced with 10
  + Create new File objects that represent files named "integers1.txt", "integers2.txt", "strings1.txt", and "strings2.txt".
  + Create a scanner for each file
  + Create seperate **GenerickStack** objects that hold the objects from the files.
  + Fill the GenericStack objects with the elements.
    - Use While loops with **hasNext** method as condition.
  + Call the generic **printStack** method, run the integers1 and integers2 GenericStack objects through it, and display the values for both.
  + Call the **sortStack** method to sort both integers1 and integers2
  + Call the generic **printStack** method, run the integers1 and integers2 GenericStack objects through it, and display the sort values in descending order for both.
  + Assign the **mergeStacks** method to merge both integer1 and integer2 to a GenerickStack object that holds both integers1 and integers2 objects.
  + Call the generic **printStack** method, run the integers1 and integers2 GenericStack objects through it, and display the merged values in ascending order.
  + Call the generic printStack method, run the strings1 and strings2 GenericStack objects through it. and display the values for both.
  + Call the sortStack method to sort both strings1 and strings2
  + Call the generic **printStack** method, run the strings1 and strings2 GenericStack objects through it, and display the sort values in descending order for both.
  + Assign the mergeStacks method to merge both string1 and string2 to a GenerickStack object that holds both strings1 and strings2 objects
  + Call the generic printStack method, run the strings1 and strings2 GenericStack objects through it, and display the merged values in ascending order.
  + Make sure to close the files
* **Outside the Main method within the same class**
  + Create a public static method that replaces the 0 values in the stack with the value 10.
    - Create a stack based on Java’s available JCF called **stacks.**
    - Can only use one temporary stack of type **Stack<Integer>**
    - Process each element in the original stack using while and/or if/else
      * Replace the 0 values with 10 using **push** and **pop**
    - Restore to the original stack
  + Create a non-generic method called **printStack** that displays stack in original state
    - Create a stack based on Java’s available JCF called **stacks.**
    - Print the original stack with 0s replaced with 10s
  + Create a generic method called **printStack** for displaying the file values
    - Can only use one temporary stack of type **GenericStack<E>**
    - Process each element in the original stack.
    - Push the current element to the number stack.
    - Restore to original stack and print.
  + Create a generic method called **sortStack** that extends to the **Comparable** class for the purpose of sorting the file values from largest to smallest.
    - Can only use one temporary stack of type **GenericStack<E>**
    - Process each element in the original stack.
    - Push the current element to the number stack
    - Place values in descending order.
    - Restore to original stack
  + Create a generic method called **mergeStacks** that extends to the **Comparable** class for the purpose of merging the file values
    - Can only use one temporary stack of type **GenericStack<E>**
    - The incoming stacks do not need to be preserved
    - Process each element in the original stack.
    - Push the current element to the number stack
    - Return the values
* **Outside the class where Main method resides** 
  + Create a generic stack class called **GenericStack** that will have a LIFO behavior.
    - In the private data field, create an **ArrayList** as a storage container and behaves like a stack.
    - Create a constructor.
      * Create an empty stack by allocating memory to ArrayList
    - Create a public method to indicate if the stack is currently empty.
    - Create a public method that returns the number of objects on the stack.
    - Create a public method that returns the object at the top of the stack, but doesn't remove it.
    - Create a public method that removes the top of the stack.
    - Create a public method that adds an object to the top of the stack.