Programming Assignments #4 and 5
CS 202 Programming Systems

For This Program

With both programs #4 and #5 you will be implementing your solutions using Java. Your goal must be to develop an object-oriented solution but this time implement it in Java. You may use an IDE and you should use the string class! Make sure that your OO Design is not centered around your data structures – your data structures support the design but shouldn't be the primary emphasis of your design.

Your Java programs must follow these rules:

- **No public or friendly fields (data members) *** NONE!!!*****
- No friendly methods (member functions); all members must use the public, private, or protected keywords
- Yes, you SHOULD use the string class!
- Limit your use of static methods – these should be restricted to just utility functions and main
- Use an inheritance hierarchy using "extends"; there must be a minimum of 5 classes with 3 of them in a hierarchy. *These should not be isolated to just your data structures.*
- Create at least one abstract base class
- Implement at least one constructor with arguments
- Implement at least two functions using function overloading between classes and experiment with the way function overloading works in Java. ***Write about this.
- Implement dynamic binding and experiment with how it works in Java. Prove to yourself that the functions are being overridden versus overloaded. ***Write about this
- Use the super keyword in invoking a base class' constructor. *This is what we use instead of an initialization list.*
- The data structures required in the assignment (below) are to be fully implemented by you

For each of the above that you experiment with, write up information about it in your efficiency write-up

What papers do you need to write?

1. There is ONLY ONE design writeup for the combined programs 4-5. This means, you are required to turn in ONE paper on how this solution will be object oriented (your design).
2. With EACH program 4 and program 5, you are required to write two papers:
    a. New Syntax – 400 words about the new syntax that you learned about (how you convinced yourself that the overriding, overloading, or hiding was taking place) instead of the efficiency writeup.
    b. IDE – 200 words about how you used the IDE to create your solution (instead of the debugger writeup)

Data Structures

In these last two programs, you must implement two data structures:

1. Program 4: A binary search tree or 2-3-4 tree
    a. In program 4, implement the other tree that you did not implement in program 3. If you have already implemented a 2-3-4 tree in program 3, then implement a BST in program 4, implementing insert, display, retrieve, retrieve all related items, remove an individual item, and remove all; the algorithms must be implemented recursively.
    b. If you have not yet implemented a 2-3-4 tree yet, then this will need to be implemented with program 4. Implement the insert, display, retrieve, retrieve all related items and remove_all (no remove individual items).
2. Program 5: A linear linked list where each node has an array of data

The required data structures specified in the assignment must be your own implementation: as in BSTs (or balanced tree) and the LLL of arrays. Once you meet the basic requirements of the assignment, you are allowed to use libraries for any subsequent data structures.

Program Requirements

There is a trend towards playing VR games. As a game designer, my daughter often talks about the decisions that players must make as they work through their game in virtual reality. For each decision there are consequences. Designers use decision analysis such as with a decision tree to help guide the design of these kinds of games. At each point in the tree, we can have a decision, chance, or end the game. Think of them like and if/else sequence. Based on some condition, we either go left for result 1 or go right for result 2 which in turn are additional decisions, chance happenings, or a dead end.

This program is divided into two parts. The first part is to build the decision tree. When you are working on this part, think of yourself as the "game" designer. This is program #4 with the BST or balanced tree as your data structure. Each node can be one of four different types of objects (consider dynamic binding for this!). Because we want to use hierarchies, create at least four different categories of types of decisions (Decision, Chance, End, and one more of your own selection).

Then, once the data exists, with program #5 we can have users work through the decision tree and keep in the secondary data structure the history of their journey through the "game" of decisions and their ultimate ramifications.

Try to have some fun this with!!

Your job is to come up with a design of an OO framework that will support this type of application. The key is to make sure to solve this problem using Object Oriented methodologies **with dynamic binding** and function overloading. *The use of external data file(s) are necessary!*