How Does Our Family Tree Works

Intro for user:

Our program helps to identify the title of your family member so that you won't experience the awkward situation in family reunion in Chinese New Year when you don't know how to address your relative! All you need to do is to input your relatives information. Our program stores that information and creates a family tree automatically. Just click "get title", if you want to know how to call your selected relative. You can also keep track of all your family members through in the Tree Panel.

Intro for programmer:

We use a weighted graph to represent the family tree. Arcs of weight one represents a parent-to-child relationship; arcs of weight 0 represent sibling-to-sibling relationship; Arcs go from older generation to younger generation; edges go between people in the same generation. With the weight we are able to calculate the generation gap between two people, which will be a factor determining how you supposed to call your relative. Other factors include gender (isFemale in Family Class), age (isOlder in FamilyMember), whether the relative is from the mother's side or father's side (isMaternal in Family) and spousal relationship (isSpouse).

TECHNICAL REPORT

ADT

- Graph: A weighted graph used to represents all the family member. LinkedList: used within the Graph to hold all the vertices, arcs and weights
- LinkedStack: used in the getPath method in order to backtrack

Classes

- FamilyMember
    - Represents each individual
    - Stores information about each person (birthday, name, gender, image filename, spouse)
    - Methods
        - Getters and setters:
            - getSpouse, setSpouse
            - getGender, setGender
            - getBDay, getBMonth, getBYear
            - getTitle, setTitle
            - getName
        - isOlder(FamilyMember Person)
            - A boolean method that returns true if a family member is older
        - toString
            - Return a string representation of the class

- AdjListsWeightedGraph<T>

- an adaptation of the AdjListsGraph<T> Class that creates a graph with weights and include method that could allow us to find a path from the start vertex to a destination
- Implements the graph interface
- Methods
  - private void addArc(int n1, int n2, int weight)
    - Helper method.Inserts an arc with its weight between two vertices of the graph. If arc already exists, ignores the addition.
  - public void addArc (T v1, T v2, int weight)
    - Inserts an arc with its weight between two vertices of the graph.
  - public void addEdge (T vertex1, T vertex2, int weight)
    - Inserts an edge between two vertices of the graph. If one or both vertices do not exist, ignores the addition.
  - public void removeVertex(T v)
    - Remove a vertex from the graph. Ignores the removal if the vertex does not exist.
  - public void removeArc (int index1, int index2)
    - Remove an arc and its weight from the graph. Ignores the removal if the index is not valid.
  - public int getWeight(T start, T destination)
    - Get the weight of an arc. Return a invalid maximum integer value if there is no arc exist between two vertices
  - public String toString()
    - Returns a string representation of the graph.
  - public getPath(T vertex, T target)
    - Get a path between two vertices
  - public LinkedStack<T> getPathStack(T vertex, T target)
    - Helper. Returns a stack representation of the path

- Family
  - Represents the family, is a collection of family members
  - uses the AdjListsWeightedGraph<T>
  - Methods
    - Add Methods: adding a FamilyMember to the graph:
      - public void addParent(FamilyMember current, FamilyMember parent)
      - public void addSpouse(FamilyMember current, FamilyMember spouse)
      - public void addSibling(FamilyMember current, FamilyMember sib)
      - public void addChild(FamilyMember current, FamilyMember child)
    - Determine the identity of a FamilyMember:
      - public boolean isFather(FamilyMember current, FamilyMember person)

        - public boolean isSpouse(FamilyMember current, FamilyMember person)
        - public boolean isSib(FamilyMember current, FamilyMember person)
        - public boolean isMother(FamilyMember current, FamilyMember person)
        - public boolean isFemale(FamilyMember person)
    - Get the children of a FamilyMember:
        - LinkedList<FamilyMember> getChildren(FamilyMember person)
    - Return true if the family member is from the mother side
        - public boolean isMaternal(FamilyMember member)
    - Get the generation gap between two family member
        - public int getGeneration(FamilyMember younger, FamilyMember older)
    - Get the title of a FamilyMember
        - public String getTitle(FamilyMember person)
    - Return a string representation of the FamilyMember
        - public String toString()
    - Get a string array representation of all members
        - public String[] getMemberArray()
    - Get a Linkedlist representation of all members
        - public Linkedlist<FamilyMember> getMembers()
    - Get a specific family member
        - public FamilyMember getMember()
    - Getters and Setters
        - public void setUser(FamilyMember user)
        - public FamilyMember getUser()
        - public AdjListsWeightedGraph<FamilyMember> getFamilyTree()

GUI

- TreePanel
    - Display the family tree

- InfoPanel
    - The initial interface

- AddMemberPanel
    - Add a member to the family tree
    - Input family member's personal information
- FamiilyTree

Known Bugs/Deficiencies

- Display of the graph
    - Arrows between couples

    - No images in the graph component; Instead we have it show up in the Add Member tab with the Get Title button

Potential Improvements (beyond fixing bugs/deficiencies)

- Using a decision tree for the getTitle method may be more elegant and less convoluted
    - Switch statement may also work better
- More flexibility beyond three generations