Lab Assignment 2

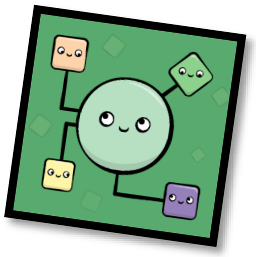
Classes & Objects, UML Diagrams, The Object Type

For all assignments, it is important that you start as early as possible and ask questions before the deadline. Also, make use of the resources provided on blackboard (Videos, Source Code, etc.) which will always pertain to the assignment topic.

NOTE: The length of instructions may at first, be daunting. BUT, if you take this assignment step by step (line by line) you will realize that detailed here for you, is the answer! This is the pseudo code for your finished assignment. Step, by, Step.

Video Demo / Explanation Playlist:
https://www.youtube.com/watch?v=MFhJmwRCMJw&list=PLDLzQoPnlK2YEMkuFNi4UP75UVV6KB6mM&index=1

Due Date: 02 / 20 (Monday)



For this assignment you will create & submit 3 classes:

1. **Family**
2. **Friend**
3. **FriendsAndFamily (Main Class)**

Begin by creating a class named **Family** *(This class will not have a main method)*
Below is the UML diagram for the class.

Family
- name : String - relation : String
+ Family(String n, String r) + toString ( ) : String

In the Family class:

Data fields:

- **name,** is a global (instance) variable of type String
- **relation**, is a global (instance) variable of type String

Constructor:

- There is no default constructor for this class
- The constructor will take two arguments (String n and String r)
- Use the "private = public" for each variable within the constructor
    - i.e. name = n

Method:

- **toString ( )**
    - *Returns a neatly formatted String as seen in the images on the last page of this document (For FAMILY)*

See next page for Friend class information & instructions

Next, create a class named **Friend** *(This class <u>will not</u> have a main method)*
Below is the UML diagram for the class.

Friend
- name : String - timeKnown : int
+ Friend(String n, int tk) + validateTimeKnown (int tk) : int + toString ( ) : String

In the Friend class:

Data fields:

- **name,** is a global (instance) variable of type String
- **timeKnown**, is a global (instance) variable of type int

Constructor:

- There is no default constructor for this class
- The constructor will take two arguments (String n and int tk)
- Use the "private = public" for the name variable within the constructor
- <u>Assign the timeKnown variable to the validateTimeKnown method, passing tk</u>

Methods:

<u>Remember, refer to the UML diagram for arguments, public/private and return types</u>

- **validateTimeKnown ( )**
    - *Returns an integer value for timeKnown variable based on tk argument passed*
    - *If the users defined argument is < 0 we assign timeKnown to 0*
    - *Otherwise, timeKnown is what the user passes to the method call*

- **toString ( )**
    - *Returns a neatly formatted String as seen in the images on the last page of this document (For FRIEND)*

See next page for FriendsAndFamily class information & instructions

Now, create a main class named **FriendsAndFamily** *(This class* ***<u>will</u>*** *have a main method)*

Below is the UML diagram for the class.
— Note: The instructions, variables and general information pertaining to the main method is not presented in the UML but will be below.

FriendsAndFamily
+ getSize(Scanner sc) : Object[] + friendOrFamily(Scanner sc) : int + getName (Scanner sc, int t) : String + getRelation(Scanner sc, String n) : String + getTimeKnown(Scanner sc, String n) : int + display(Object[] faf) : void ***<u>As these methods are part of the main class, you can assume all of the above methods are static</u>***

In the FriendsAndFamily class:

1. In the **main method**, create the following variables:
   - **A. Scanner**
   - B. **Object Array** named **friendsAndFamily** (Declare but do not assign)
   - **C. Friend object** (Declare but do not assign) *<u>i.e. -> Friend fri;</u>*
   - **D. Family object** (Declare but do not assign)
   - **E. String name**
   - **F. String relation**
   - **G. int timeKnown**
   - **H. int type**

2. Write the following functionality for each of the methods:
   <u>Remember to check the UML for return type and parameters</u>

   A. getSize ( )
   - **I.** Create integer size
   - **II.** Use a validation loop to ensure size is positive
   - **III.** Create an object array based on user-defined size variable
   - **IV.** Return the array

   B. friendOrFamily ( )
   - I. Ask user if they are entering a friend or family member (String)
   - II. Determine which String is entered (case does not matter) this information will be used with the **type** variable created earlier:
     - If “friend” return 0
     - If “family” return 1
     - Otherwise, return 2

- **C. getName ( )**
    - **I.** Using the **t** variable passed, use a control structure to determine:
        - ❖ If t is 0 prompt for friend input
        - ❖ If t is 1 prompt for family input
        - ❖ Otherwise, end the program with a System.exit(0)
    - **II.** Read in the given name using .next()
    - **III.** Clear the buffer if needed (.nextLine())
        - ❖ **Further info on buffer issues:** **https://www.youtube.com/watch?v=hVoYpa_ryl0&list=PLDLzQoPnlK2Z-ZqfkJXyiv7GjgNdocmyO&index=11**
    - **IV.** Return the given name

- **D. getRelation ( )**
    - **I.** Prompt user for their relation to given name
    - **II.** Return relation variable (String)

- **E. getTimeKnown ( )**
    - **I.** Prompt user the length they have known their given friend
    - **II.** Return integer variable for time known

- **F. display ( )**
    - I. Using an enhanced for loop, loop through the given object array printing the toString for each iteration

3. Back in the main method, use a for loop pertaining to the length of the object array, **<u>within this loop</u>**:

- A. Assign type to the method **friendOrFamily**

- B. **<u>If</u>** type is 0 then:
    - I. Assign name to getName passing the appropriate variables
    - II. Assign timeKnown to getTimeKnown passing the appropriate variables
    - III. Assign **fri** to a new Friend object passing the appropriate variables
    - IV. Assign the ith element of the array to fri

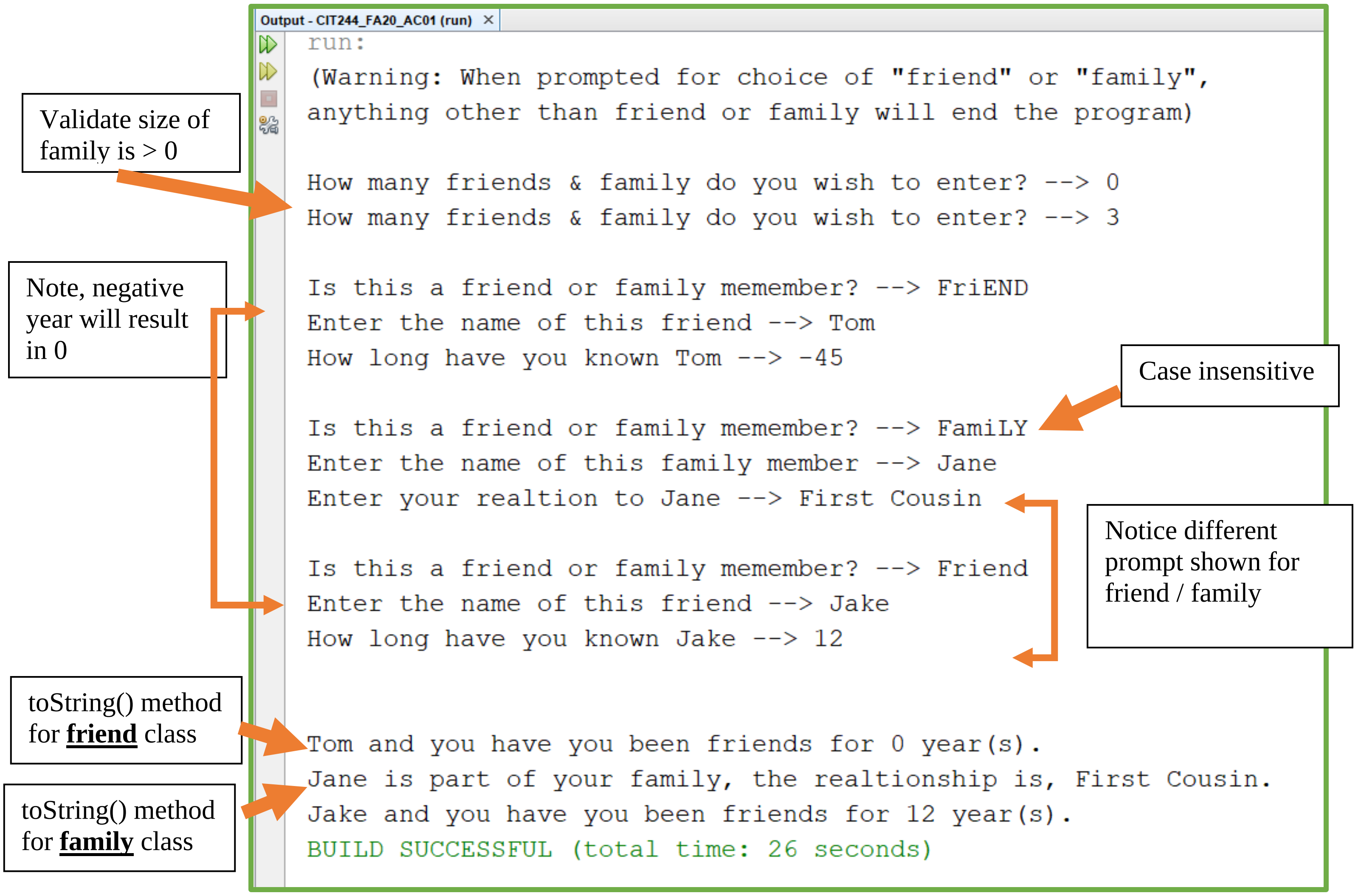
- C. If type is 1 then:
    - I. Repeat the above, but instead of getTimeKnown call the relation method.
    - II. Assign **fam** to a new Family object (not fri)
    - III. Assign the ith element of the array to fam

- D. If type is NOT 0 or 1 then state you are ending the program due to invalid input and end using a System.exit(0)

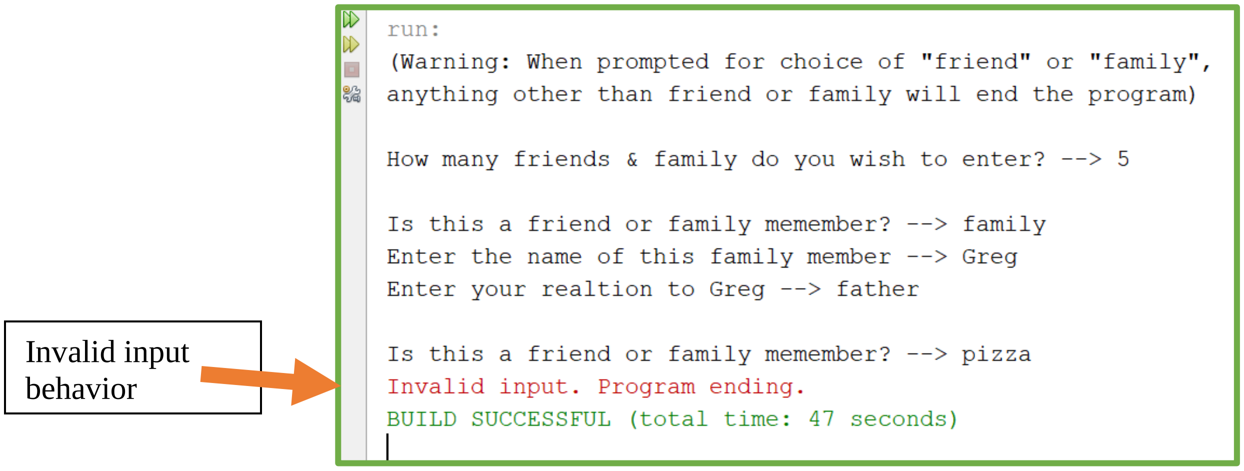
4. Outside of the loop call the display method passing the object array.

Images of programming running on next page

Example of programming running … Feel free to get creative. You do not have to have the EXACT output as shown.


Output - CIT244_FA20_AC01 (run)
Validate size of family is > 0
Note, negative year will result in 0
Case insensitive
Notice different prompt shown for friend / family
toString() method for **friend** class
toString() method for **family** class

```
run:
(Warning: When prompted for choice of "friend" or "family",
anything other than friend or family will end the program)

How many friends & family do you wish to enter? --> 0
How many friends & family do you wish to enter? --> 3

Is this a friend or family memember? --> FriEND
Enter the name of this friend --> Tom
How long have you known Tom --> -45

Is this a friend or family memember? --> FamiLY
Enter the name of this family member --> Jane
Enter your realtion to Jane --> First Cousin

Is this a friend or family memember? --> Friend
Enter the name of this friend --> Jake
How long have you known Jake --> 12


Tom and you have you been friends for 0 year(s).
Jane is part of your family, the realtionship is, First Cousin.
Jake and you have you been friends for 12 year(s).
BUILD SUCCESSFUL (total time: 26 seconds)
```



```
run:
(Warning: When prompted for choice of "friend" or "family",
anything other than friend or family will end the program)

How many friends & family do you wish to enter? --> 5

Is this a friend or family memember? --> family
Enter the name of this family member --> Greg
Enter your realtion to Greg --> father

Is this a friend or family memember? --> pizza
Invalid input. Program ending.
BUILD SUCCESSFUL (total time: 47 seconds)
```

If you have any questions, please ask for a demonstration of the program executing.