

Exercise 1 <<30 points>>

We have been asked to develop a software solution for Nimbus Airlines' Frequent Flyer department. Nimbus Airlines has provided the following explanation of how its Frequent Flyer program works:

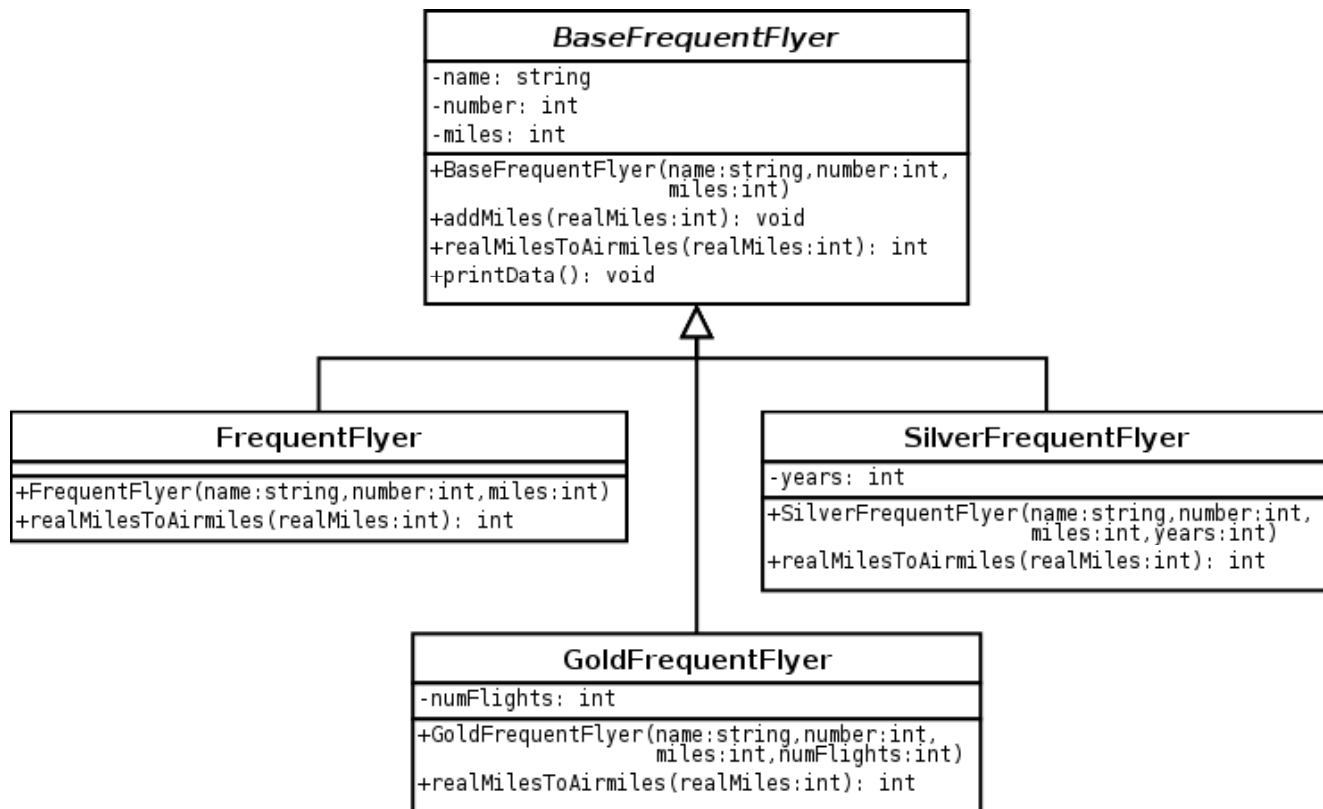
In the Frequent Flyer (FF) department, we need to keep track of all the clients who have enrolled in the FF program. When a client signs up, he/she must provide his/her *name*, and is assigned an 8-digit *number*. The client's record also reflects the number of *miles* accumulated, which can later on be used to obtain free flights or upgrades.

However, the number of miles accumulated in a client's FF account does not necessarily correspond with the actual number of miles traveled by the client. Depending on the client's category within the FF program, he/she might accumulate more miles than actually traveled. We usually use the terms *real miles* and *air miles* to distinguish both types of miles (the former being the real number of miles traveled, and the latter being the number of miles accumulated in the client's account).

Our FF program has three types of frequent flyers:

- The ordinary *frequent flyer* is the lowest category. Clients in this category receive as many air miles as real miles.
- The *silver frequent flyer* rewards clients that have been with us for a long time. When a silver frequent flyer adds miles to his/her account, the number of air miles is equal to the number of real miles times the number of years, plus one, that client has been enrolled in the program. For example, if a silver frequent flyer takes an 800 mile flight, and he/she has been enrolled for three years, the number of air miles is $800 * (3+1) = 3200$.
- The *gold frequent flyer* rewards clients that fly with us frequently. When a gold frequent flyer adds miles to his/her account, the number of air miles is equal to the number of real miles times one tenth (rounded down) of the number of times he/she has flied with us (plus one). For example, if a gold frequent flyer takes an 800 mile flight, and he/she has previously been on 97 flights, the number of air miles is $800 * (97/10 + 1) = 800 * (9 + 1) = 8000$.

Our software design department has come up with the following object-oriented model for the stated problem:



The design department has also provided the following comments to the diagram:

- `printData()` prints the client's name, number, and miles.
- `realMilesToAirmiles()` converts a number of real miles to air miles, taking into account the client's category.
- `addMiles()` adds a number of real miles to a user's account. Internally, this function must use the `realMilesToAirmiles()` function to determine what amount of airmiles must be added to the *miles* member variable.

The diagram does not reflect what functions are virtual.

Implement the four classes described in the UML class diagram shown above. The points in this exercise are divided as follows:

- 15 points for implementing the four classes, correctly using the inheritance features in C++.
- 15 points for correctly implementing `addMiles()` and `realMilesToAirmiles()` using the polymorphism features in C++.

To test your implementation, a `main.cpp` file is provided in the homework files that should yield the following output:

Cornelius Doe	#23166841	200mi
Lucius Doe	#94565432	700mi
Rufus Doe	#32155994	1200mi

No header files are provided, so the `main.cpp` file will only work if you use exactly the same class and function names shown in the UML diagram.

Exercise 2 <<Bonus points: 20 points>>

You are provided with the same Tree implementation provided in the previous homework. Implement the breadth function:

```
void breadth(Tree &t);
```

This function does a breadth-first traversal of the tree. This algorithm has not been explained in class, so you will have to look it up elsewhere. Note that most sources show the algorithm for traversing cyclic graphs. The algorithm for breadth-first traversal of a tree is slightly simpler. Hint: You will not need to make any modifications to the `TreeNode` struct. Hint #2: You will need to use a queue for this traversal. An implementation of a Queue ADT is provided for you.

A `main.cpp` file is provided for you to test your implementation. If correctly implemented, the program should print out the following:

BREADTH-FIRST TRAVERSAL

15 10 23 4 13 17 29 12 25