Homework #3 – Class SavingsAccount

In this assignment you are asked to implement a `SavingsAccount` class used to represent a savings account at a bank. This class will track the account's current balance as well as the annual interest rate that applies to all accounts. The following UML class diagram shows the attributes and behaviors of class `SavingsAccount`. *Note: UML type Float will map to type* `double` *when implemented in C++.*

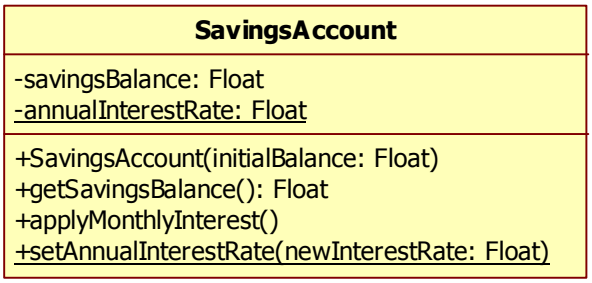

Figure 1. UML class diagram for class SavingsAccount

1. **(1 point)** Create files named "SavingsAccount.h" and "SavingsAccount.cpp" to hold your implementation of class `SavingsAccount`.
2. **(1 point)** Define a namespace based on your first and last name (e.g. "RayMitchell") in which you will define class `SavingsAccount`.
3. **(1 point)** Define class `SavingsAccount` in the files and namespace created in the previous steps. The class should have two data members:
   a. `savingsBalance` (type `double`) – an instance data member representing the current balance for the account
   b. `annualInterestRate` (type `double`) – a static data member representing the annual interest rate for all accounts
4. **(1 point)** Define a constructor that takes a single parameter used to initialize the account's balance. The constructor should ensure that the initial balance is non-negative; if the initial balance is negative an error message should be output and the balance should be set to zero.
5. **(1 point)** Define member function `getSavingsBalance` that returns the account's current balance.
6. **(1 point)** Define `static` member function `setAnnualInterestRate` that sets the annual interest rate used by all accounts. This function should ensure that the annual interest rate is non-negative; if the interest rate is negative an error message should be output and the interest rate should be set to zero.
7. **(1 point)** Define member function `applyMonthlyInterest` that calculates the monthly interest earned for the account (1/12 of the annual interest) and adds the interest to the account's balance.
8. **(1 point)** Make `SavingsAccount`'s member functions `const` where appropriate.

9. **(1 point)** Write a test program that demonstrates class `SavingsAccount` capabilities. Your test program should demonstrate all constructors and public member functions. It should also verify all error conditions are handled properly. Place your test program in a file named "hw3.cpp".
10. **(1 point)** Make sure your source code is well-commented, consistently formatted, uses no magic numbers/values, follows a consistent style, and is ANSI-compliant.

Place all source code and a screen capture of the output produced by your program in a single Word or PDF document. Submit this document.