Eideen Mozaffari

UID: 105988436

Project 6 Report: Vacation Account Balance

Notable Obstacles/Challenges

1. BloodDonation::setID
   1. When I first read the spec, my first thought was how am I going to iterate through each digit in the ID and make sures its valid. So, I converted the passed in integer to a string, and used a loop to make sure each element of the string was a digit. After I realized this was unnecessary, because only an integer can be passed through as a parameter, I updated my conditions within my function, and it produced the same results but was less lines of code and much easier to read. Also, once we were told that leading 0s were not being tested (I’m not even sure how you would keep the leading zeroes after an integer is passed in as a parameter), the bounds for the ID became very clear.

Test Data

1. I tested using the given asserts in the spec, but I will not copy and paste those here. Instead, I will list some of the things that the spec’s asserts tested on:
   1. Creating multiple BloodDonation objects, some with valid and invalid parameters passed in for ID, Age, and Weight
   2. Creating a VacationAccount Object with a valid ID
   3. Using the getter/accessor methods in the BloodDonation class to retrieve values stored in private variables
   4. Using the getter/accessor methods in the VacationAccount class to retrieve values stored in private variables
   5. Calling the addVacationToAccount method, testing both cases for when the BloodDonation ID does and does not match the VacationAccount ID
   6. Calling the getBalance method in the VacationAccount class after adding a valid BloodDonation to the account to reflect the updated vacation hour balance
2. Now, I will copy and paste the asserts I wrote that test the code and accounted for some boundary cases:

// Created more BloodDonation objects. Some of the objects are invalid, some of the parameters for the objects are on the inclusive bounds, and some are completely valid.

* 1. BloodDonation doner5( 1234.5, -40, 285.00);
  2. BloodDonation doner6( 12345, 66, 0);
  3. BloodDonation doner7( 123456., 2.1, 280.00);
  4. BloodDonation doner8( 123456.000, 27.1, 101.00);
  5. BloodDonation doner9( 12345, 21, -150.00);
  6. BloodDonation doner10(987654, 65, 101.0000);
  7. BloodDonation doner11(123456, 21, 280.0000);
  8. BloodDonation doner12(987654, 66, 101.0000);
  9. BloodDonation doner13(987654, 64, 100.0000);

// Testing my Getter/Setter Functions through constructor and main:

1. assert(std::to\_string(doner5.getID( )) == "-1");
2. assert(std::to\_string(doner6.getID( )) == "-1");
3. assert(std::to\_string(doner7.getID( )) == "123456");
4. assert(std::to\_string(doner8.getID( )) == "123456");
5. assert(std::to\_string(doner3.getAge( )) == "65");
6. assert(std::to\_string(doner4.getAge( )) == "-1");
7. assert(std::to\_string(doner5.getAge( )) == "-1");
8. assert(std::to\_string(doner6.getAge( )) == "-1");
9. assert(std::to\_string(doner7.getAge( )) == "-1");
10. assert(std::to\_string(doner8.getAge( )) == "27");
11. assert(std::to\_string(doner9.getAge( )) == "21");
12. assert(std::to\_string(doner1.getWeight( )) == "-1.000000" );
13. assert(std::to\_string(doner5.getWeight( )) == "-1.000000" );
14. assert(std::to\_string(doner6.getWeight( )) == "-1.000000" );
15. assert(std::to\_string(doner7.getWeight( )) == "280.000000" );
16. assert(std::to\_string(doner8.getWeight( )) == "101.000000" );
17. assert(std::to\_string(doner9.getWeight( )) == "-1.000000" );

// Created more VacationAccount Objects, one has an invalid ID, and the other ones have their ID corresponding with BloodDonation objects, so I can test my addVacationToAccount function:

1. VacationAccount account(889543);
2. VacationAccount account2(543);
3. VacationAccount account3(123456);
4. VacationAccount account4(987654);

// Testing my Getter/Setter Functions through constructor and main

1. assert( account2.getBalance() == 0.000000);
2. assert( account2.getID() == -1);
3. assert(account2.addVacationToAccount(doner2) == false);
4. assert(account2.addVacationToAccount(doner1) == false);
5. assert(account2.addVacationToAccount(doner3) == false);
6. assert(account2.getBalance( ) == 0.000000 );
7. assert(account2.getID( ) == -1 );
8. assert( account3.getBalance( ) == 0.000000 );
9. assert( account3.getID( ) == 123456);
10. assert( account3.addVacationToAccount( doner11 ) == true );
11. assert( account3.addVacationToAccount( doner1 ) == false );
12. assert( account3.addVacationToAccount( doner4 ) == false );
13. assert( account3.addVacationToAccount( doner11 ) == true );
14. assert( account3.getBalance( ) == 8.000000 );
15. assert( account3.getID( ) == 123456 );

// Just for safe measure:

1. assert( account4.getBalance( ) == 0.000000 );
2. assert( account4.getID( ) == 987654);
3. assert( account4.addVacationToAccount( doner10 ) == true );
4. assert( account4.addVacationToAccount( doner12 ) == false );
5. assert( account4.addVacationToAccount( doner13 ) == false );
6. assert( account4.getBalance( ) == 4.000000 );
7. assert( account4.getID( ) == 987654 );