PseudoCode in Account.java:

1. Imports needed:
   1. Calendar to get the current date
   2. Date for the datatype tohold a date
   3. Random to get random numbers
2. Create a private global variable id to hold the id
3. Create private global variable balance to hold the balance
4. Create private global variable annualInterestRate to hold the annual interest rate
5. Create private global variable dateCreated to hold the date created
6. Blank Constructor:
   1. Create a Random object to get a random balance and ID
   2. Set the global ID and balance to the random numbers gotten from the random object
7. Constructor that accepts two variables, an int and a double
   1. Set the global ID to the int passed, and the balance to the balance passed
8. getId() method, returns an int
   1. Return the global ID
9. setId(int) method, returns nothing
   1. Set the global ID to the ID that the user passed
10. getBalance() method, returns a double
    1. Return the global balance
11. setBalance(double) method, returns nothing
    1. Set the global balance to the balance that the user passed through
12. getAnnualInterestRate() method, returns a double
    1. Returns the global annual interest rate
13. setAnnualInterestRate(double) method, returns nothing
    1. Sets the global annual interest rate to what the user passed through
14. getDateCreated() method, returns a Date
    1. Returns the global date created variable
15. getMonthlyInterestRate() method, returns double
    1. Returns the annual interest rate divided by 12 for the monthly interest rate
16. deposit(double) method, returns nothing
    1. Adds whatever the user passed through to the global balance
17. withdraw(double) method, returns nothing
    1. Check if the balance minus the argument the user passed is greater than 0
       1. If it is, subtract the money from the global balance
       2. If it’s not, print an error that says there’s not enough money and show the current unchanged balance

PseudoCode in Main.java:

1. Create a new Account object with an ID of 1122, and initial balance of 20000
2. Set the annual interest rate of the account to 4.5 for 4.5%
3. Withdraw 2500 from the account
4. Deposit 3000 to the account
5. Show the user the current balance, monthly interest rate, and the date the account was created.

Flow chart:

A screenshot of a cell phone

Description automatically generated

Test Plan:

Create more Account objects and give them random values and check by eyesight to see if there’s anything wrong with it.

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cases** | **Input** | **Expected Result** | **Actual Result** | **Did It Pass?** |
| Case 1 | Account as a blank constructor  Set annual interest rate to 2.4  Deposit 200  Withdraw 130 | Account to get a random starting balance and ID  Balance:  Initial Balance + 200 – 130  Monthly Interest Rate:  0.2 | Account to get a random starting balance and ID  Balance:  Initial Balance + 200 – 130  Monthly Interest Rate:  0.199999998 | Y |
| Case 2 | Account with ID being 200 and initial balance of 100  Annual Interest Rate:  7.4  Deposit 20  Withdraw 500 | Account with ID 200 and initial balance of 100  Error from withdrawing too much  Balance:  120  Monthly Interest Rate:  0.62 | Account with ID 200 and initial balance of 100  Error from withdrawing too much  Balance:  120  Monthly Interest Rate:  0.62 | Y |
| Case 3 | Account with ID being 500 and initial balance of 200  Annual Interest Rate:  1.8  Deposit 100000  Withdraw 20000 | Account with ID 500 and initial balance of 200  Balance:  80200  Monthly Interest Rate:  0.15 | Account with ID 500 and initial balance of 200  Balance:  80200  Monthly Interest Rate:  0.15 | Y |

Screenshots:

Case 1

A screenshot of a cell phone

Description automatically generated

Case 2

A screenshot of a cell phone

Description automatically generated

Case 3

A screenshot of a cell phone

Description automatically generated

UML Class Diagram:

A screenshot of a cell phone

Description automatically generated

Lessons learned:

I learned how to use classes to make my code easier to read and a lot easier to implement. I learned that I can now take this Accounts file in any project and use it the same way that I used it this main code. This makes it so that the same code does not exist on my machine. I also learned that the this keywork can be used in a blank constructor to call an overloaded constructor to not repeat the same code, programmers are lazy.

Checklist:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Source java files** | **Y** |  |
|  | **Compressed files:** | **Y** |  |
|  | FirstInitialLastName\_Project6\_Moss.zip | **Y** |  |
|  |  | **Y** |  |
|  | FirstInitialLastName\_Project6\_doc.zip | **Y** |  |
|  | **Program compiles** | **Y** |  |
|  | **Program runs** | **Y** |  |
|  | **Checklist is completed and included in the Documentation** | **Y** |  |
|  | **Documentation file:** | **Y** |  |
|  | **Comprehensive Test Plan** | **Y** |  |
|  | **Screenshots based on Test Plan** | **Y** |  |
|  | **UML Diagram** | **Y** |  |
|  | **Algorithms/Pseudocode** | **Y** |  |
|  | **Flowchart** | **Y** |  |
|  | **Lessons Learned** | **Y** |  |