Ryan Richardson

Foundations of Programming

Final Project

* Create an **Account class** that holds the owner’s first name, last name, ssn, pin, and social security number.
* Constructor method creates variable attributes for all the data.
* A static method called generate\_pin to create a random pin upon account creation.
* A method to set the owner’s first name and a method to get the owner’s first name which will return it..
* A method to set the owner’s last name and a method to get the owner’s last name which will return it..
* A method to set the SSN and a method to get the SSN which will return it.
* A method to set a randomly generated account number and a method to get that account number which will return it.
* A method to set the pin when changing it and a get pin method to return the pin.
* A set balance method that will start at zero and a get balance that will return it.
* A deposit method that will add the amount to the account balance and return it.
* A withdraw method that will subtract the amount from the balance and return it.
* An ATM withdrawal method that will function in the same way but only in intervals of 5 and will print out the number of 20, 10, and 5 dollar bills.
* An is valid pin method to check that the pin is 4 digits only.
* An is valid ssn method that checks that the ssn is 9 digits only.
* A \_\_str\_\_ method that prints out all the account information in a formatted string.
* Create a **Bank class** that creates the bank and has options of adding, removing, and finding accounts as well as adding interest.
* Define the maximum account to be 100 and create an empty list in the constructor to represent the bank.
* Add account to bank method will check that there are less than 100 accounts then append the new account.
* Remove account from bank will find the selected account and if it is present, remove it.
* A method findAccount that searches for the account by way of the account number and returns it.
* An add monthly interest method to calculate the imputed interest rate and add it to all accounts in the bank.
* Create a **BankUtility class** that has methods for number generation, and prompting for values that will be accepted.
* Prompt user for string method that will only take a number that is numeric. (static method)
* Prompt user for positive number method that will only accept a value that is numeric and a positive number. (static method)
* A generate random integer method that will generate a number based on the min and max values. Used when creating the account number. (static method)
* Convert from dollars to cents method that takes the balance and calculates what it would be in cents and returns it. (static method)
* A method isNumeric to check that the input is numeric.
* Create a **CoinCollector class** that creates a dictionary and assigns each letter that represents coins a value.
* A method parseChange holds the dictionary then iterates through what the input would be and adds the calculated values to another variable called total\_cents.
* Create a **BankManager class** that runs the main program and houses all the functionality around accounts in the bank.
* Constructor method creates a variable attribute to an instance of the bank class
* A method to prompt for account number and pin will be called on all actions to find the account in the bank that the action is being performed on.
* A main method will house everything inside including a print statement of all the choices available.
* Choice 1 will create an account object in the bank by asking for user input as well as random generating for the account number and pin and will add the account to the bank.
* Choice 2 will find the account then print the \_\_str\_\_ from the Account class to display the account information.
* Choice 3 will find the account then ask for your pin to change it. If the pin is valid and was entered correctly twice, it will set your account to that pin.
* Choice 4 will prompt for an amount to deposit then add that amount to the balance in the selected account.
* Choice 5 will ask for an account to transfer from then the amount you want to transfer and the account to transfer to. The amount will be withdrawn from the transfer from account and deposited into the transfer to account.
* Choice 7 will only withdraw amounts in intervals of 5 and print out the number of 20,10, and 5 dollar bills given.
* Choice 8 will use the CoinCollector class to parse through the string input and calculate the values based on the keys. Once added up that value will be deposited into the account.
* Choice 9 will remove the account selected if it is found in the bank.
* Choice 10 will ask for a yearly interest then calculate the monthly interest and add that to all accounts in the bank.
* Choice 11 will convert the balance of the selected account to cents by multiplying by 100.
* Choice 12 will end the programs by breaking it.
* Any other input besides 1-12 will be invalid.
* An instance of the BankManager class is created and main is called.
* All correlated unit tests added for each function.