Project

Task 1 (CAccount.py)

1. Create the `CAccount` class file.
2. Define a class variable for `BSBno` with a value of **1246**.
3. Create 4 instance variables as outlined in the UML diagram.
4. Write a constructor with two options:
    - **Option 1**: Initialize all 4 instance variables.
    - **Option 2**: Initialize 3 variables, with the 4th assigned a default value of **50.00**.
5. Implement getter methods for all 4 instance variables.
6. Implement a method for deposit transactions using polymorphism (algorithm is provided). (refer to Week 13).
7. Create a function to display account details in string format.

Task 2 (SAccount.py)

1. Create the `SAccount` class file.
2. Define a class variable for `BSBno` with a value of **1246**.
3. Create 4 instance variables as outlined in the UML diagram.
4. Write a constructor with two options:
    - **Option 1**: Initialize all 4 instance variables.
    - **Option 2**: Initialize 3 variables, with the 4th assigned a default value of **500.00**.
5. Implement getter methods for all 4 instance variables.
6. Implement a method for deposit transactions using polymorphism (algorithm is provided).(refer to Week 13).
7. Create a function to display account details in string format.

Task 3 (Customer.py)

1. Create the `Customer` class file.
2. Define 5 instance variables as outlined in the UML diagram.
3. Write a constructor to initialize all 5 instance variables.
4. Use aggregation to store an object from either the `CAccount` or `SAccount` class. - week 14
5. Implement getter methods for all 5 instance variables.
6. Create a function to display customer details in string format.

Task 4 (Bank.py)

1. Create the `Bank` class file.
2. Write the `main()` function to execute the entire program.
3. Create two array lists:
   - One for `CAccount` and `SAccount` objects.
   - Another for `Customer` objects.
4. Assign the corresponding `CAccount` or `SAccount` object to each `Customer` object.
5. Read data from `CAccounts.txt`, `SAccounts.txt`, and `Customers.txt` to populate the Array lists (refer to Week 12 - CarApp8).
6. Display checking and saving account details using getter methods.
7. Perform invalid deposit transactions for `CAccount` and `SAccount` (embed in code or prompt user input). (refer to Week 13).
8. Perform valid deposit transactions for `CAccount` and `SAccount` (embed in code or prompt user input).(refer to Week 13).
9. Write account details to `BankingReceipt.txt` after successful transactions (refer to Week 12 - CarApp9).