What work is left

Sunday, March 24, 2024 7:53 PM

Let's break down Revolve Bank system for better planning:

(We will later size them down during our meeting. Aim here is to be as detailed as possible and use full sentences)

1. **Creating classes to represent Bank agents**:
   - ★ Create classes that exhaustively contain all the properties, constructors and methods of the following individuals:

     a. **The Bank manager: (Devin)**

     - ▶ Comprehensive list of properties of Bank managers that will be included:

       - ◆ personal

       i. **First name**
       ii. **Last name**
       iii. **Contact number**
       iv. **Salary**
       v. **Start date**
       vi. **End date**
       vii. **Is fulltime or not?**
       viii. **Education level**
       ix. **Years of experience**
       x. **Employee ID**
       xi. **National ID.**

       - ◆ System related information**.**

       xii. **Department/Specialization** (This can be a totally new class inserted here as property)
       xiii. **Number of employees supervised**
       xiv. **Branch code/Location :** (Can be a completely new class inserted)
       xv. **Annual Bonus**:
       xvi. **Password**:

     - ▶ Comprehensive list of meaningful constructors based on the instance variables of the BANK MANAGER provided above:

       i. **Default constructor**: all properties initialized to default.
       ii. **Parameter Constructor**: will take arguments for all instance variables above.
       iii. **Copy Constructor**: will be able to essentially create a Bank manager object based on the info of another Bank manager object.
       iv. **Subset Constructor**: Can initialize a subset of the variables, if others don't exist.

       Note: Constructor overloading helps with incremental initialization; we can create objects, Bank managers in this case, with basic data and then fully get them initialized with more data later.

     - ▶ Comprehensive list of methods for the Bank manager objects depending on the on their specific functions we had agreed upon:

       i. **The invest money button.**
       ii. **The withdraw investments button.**
       iii. **Dissolve branch button.**
       iv. **Fire agent/employee button.**
       v. **Login as a teller button.**

     - 🟥 **Plus Bank teller methods:** Think in terms of inheritance.

   b. **The Bank Teller:(Amir)**

     - ▶ Comprehensive list of Bank Teller properties that will be included:
       - ◆ Personal

       i. **First name**
       ii. **Last name**
       iii. **Contact number**
       iv. **Salary**
       v. **Start date**
       vi. **End date**
       vii. **Is fulltime or not?**
       viii. **Education level**
       ix. **Years of experience**
       x. **Employee ID**
       xi. **National ID.**
       xii. **Password:**

- ◆ Organizational
  - xiii. **Teller ID**
  - xiv. **Branch ID**
  - xv. **Date of hire.**
  - xvi. **Is the teller active?**
  - xvii. **Number of customers served.**
  - xviii. **Number of transactions.**
  - xix. **Supervisor ID.** (Maybe we will need a list of supervisors etc..??)

► Comprehensive list of meaningful constructors based on the instance variables of the BANK TELLER provided above:

- i. **Default constructor**: all properties initialized to default.
- ii. **Parameter Constructor**: will take arguments for all instance variables above.
- iii. **Copy Constructor**: will be able to essentially create a Bank manager object based on the info of another Bank manager object.
- iv. **Subset Constructor**: Can initialize a subset of the variables, if others don't exist.

► Comprehensive list of methods for the Bank teller objects depending on the on their specific functions we had agreed upon:

- i. **Open customer account button**: Essentially create a new customer object.
- ii. **Close customer account button**: Essentially delete/archive a customer object.
- iii. **Approve/below above limit transaction button**:
- iv. **Generate a report for individual customer transaction button**:
- v. **Deposit**:
- vi. **Withdraw**:
- vii. **Check balance**:
- viii. **Transfer funds**:

c. The Customer:(Hien)

► Comprehensive list of customer properties that will be included:

- ◆ Personal.
  - **i. First name.**
  - **ii. Last Name.**
  - **iii. Email address**
  - **iv. Phone number of the customer**
  - **v. Date of Birth.**
  - **vi. Is Active. Is the customer active or not?**
  - **vii. Password**

► Create default constructor and custom constructor for reason already said above:

► Comprehensive list of methods for the customer:[**Should be buttons**]

- i. **Change Passwords.**
- **ii. Request Loan.**
- **iii. Pay Loan Installment.**
- **iv. Update personal Information.**
- **v. Dispute Transaction.**
- **vi. Access to bank account classes and their properties**

d. Bank Account class: (Devin)

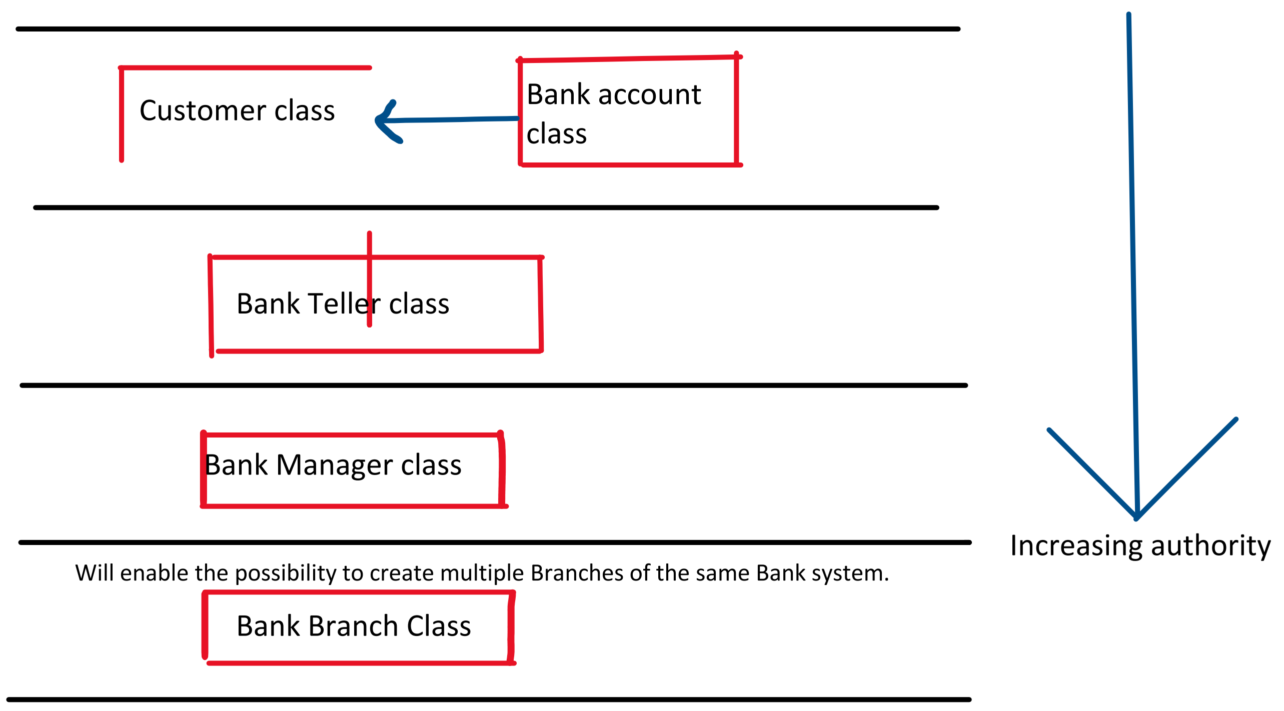
► Comprehensive list of customer properties that will be included:
- **i. Transaction History**
- **ii. Balance**
- **iii. ID Number**

► Create default constructor and custom constructor that has a beginning balance and id

► Comprehensive list of methods for the customer:
- i. **Deposit funds**:
- ii. **Withdraw funds**:
- iii. **Transfer funds**:
- iv. **Check Balance**:
- v. **View Transactions**:

a. Bank Branch class: (Lennox)

- **Background**:



Will enable the possibility to create multiple Branches of the same Bank system.

- Comprehensive List of instance variables to be created for the main Bank branch class:

  i. **Bank Manager class object (Limited to Just One)**
  ii. **Array List of objects of the Teller class**
  iii. **Array List of objects of the Customer class (This gives us access to Accounts)**
  iv. **Bank name**
  v. **Bank Location**
  Need more research.
  vi. **Employee Database**
  vii. **Customer Database**

- Comprehensive list of methods at the Bank branch Level:

  i. **Assign Manager**.
  ii. **Remove Manager**
  iii. **Total revenue**:
  iv. **Total Loans out**:
  v. **Get bank status** - Retrieves comprehensive overview of current status of the bank for example, number of Tellers, number of customers and accounts.

1. Rectangular Graphical User interfaces: **(Multiple people check Gantt Chart)**

★ Using JPanel and Jframes, (plus related utilities that you will research about), design GUIs that align with the designs agreed upon during project design milestone 3: link to PowerPoint presentation.

- Navigation system – a way to navigate back and forth through pages
- Pages for the Customer and Bank Account object
  - Buttons for transactions, loans and more
  - Balance
  - List of past transactions for an account with logging info
- Pages for Bank Teller object
  - Way to search for customers and their accounts
  - Way to add an account and customer
- Pages for Branch manager
  - Can act as bank teller
  - Able to pass a quarter

        - Get general summary information if a quarter passes
    - Pop Up systems – to aid with navigation and help with certain tasks like signatures

- **Password Hashing and Security mechanism:(Lennox)**
    - Research and come up with a clear way for handling passwords (Remember Damian proposed hashing).
    - Integrate the password handling mechanism that you came up with to the revolve BANK SYSTEM.

2. **Database to hold on the individuals accounts data from every transaction done to enabling generation of reports. (Devin)**
    - Research how to maintain a proper log of Customer transactions .
    - Research how to maintain a proper log of Customer and Bank agents personal and Organizational details.
    - Research how to maintain a log of Customer accounts: This is closely related to customer transactions already talked about.
    - Integrate these database maintenance procedures to the Revolve BANK SYSTEM: