Bank System

Create Bank system app which contains 3 modules:

1. Client Module
    a. Each client contains: int id, string name, string password, double balance.
    b. Client can login to the system using id and password.
    c. Client can deposit amount of money
    d. Client can withdraw amount of money
    e. Client can check his balance
    f. Client can transfer money to another client

2. Employee Module
    a. Each employee contains: string name, int id, string password, double salary.
    b. Employee can login to the system by id and password
    c. Employee can add new Client
    d. Employee can search for client by id
    e. Employee can list all clients
    f. Employee can edit info of client
    g. Employee can display his info

3. Admin Module
    a. Admin will be the same like Employee
    b. Admin can add new Employee
    c. Admin can search for Employee
    d. Admin can edit Employee
    e. Admin can list all employees

Hint: **you can create Person which contains name, id, password and let the client, employee inherit all data from Person, Admin could inherit from employee.**

Phase 1 -

Create the following class:

1. Client class which contains the following:
   a. Int id, string name, string password, double balance.
   b. Setter functions
      i. setName: the name must be alphabetic chars and min size 5 and max size 20
      ii. setPassword: Password must be with min size 8 and max size 20
      iii. Min balance is 1500
   c. Getter functions
   d. void deposit (double amount).
   e. void withdraw (double amount).
   f. void transferTo (double amount, Client& recipient).
   g. void checkBalance ().
   h. Display function

2. Employee Class which contains the following:
   a. Int id, string name, string password, double balance.
   b. Setter functions
      i. setName: the name must be alphabetic chars and min size 5 and max size 20
      ii. setPassword: Password must be with min size 8 and max size 20
      iii. Min Salary 5000
   c. Getter functions
   d. Display function

3. Admin Class which contains the following:
   a. Int id, string name, string password, double balance.
   b. Setter functions
      i. setName: the name must be alphabetic chars and min size 5 and max size 20
      ii. setPassword: Password must be with min size 8 and max size 20
      iii. Min Salary 5000
   c. Getter functions
   d. Display function

Hint: you can create Validation class contains all validation that you will need as static methods, and use these validation in rest of other classes.

Phase 2 –

1. Create the following text files
    a. Clients.txt to save client info
    b. Employee.txt to save employee info
    c. Admin.txt to save admin info

2. Create DataSourceInterface as abstract class contains the following
    a. Abstract void addClient(Client)
    b. Abstract void addEmployee(Employee)
    c. Abstract void addAdmin(Admin)
    d. Abstract vector<Client> getAllClients()
    e. Abstract vector<Employee>getAllEmployees()
    f. Abstract vector<Employee>getAllAdmins()
    g. Abstract void removeAllClients()
    h. Abstract void removeAllEmployees()
    i. Abstract void removeAllAdmins()

3. Create FileManager Class to implement DataSourceinterface
    a. addClient should save client info in clients.txt
    b. addEmployee should save employee info in employees.txt
    c. addAdmin should save employee info in admins.txt
    d. getAllClients(), getAllEmployees() and getAllAdmins() should retrieve data from files
    e. removeAllClients(), removeAllEmployees() and removeAllAdmins() should remove all data from the files

4. add to employee class:
    a. void addClient(Client& client).
    b. Client* searchClient(int id).
    c. void listClient().
    d. void editClient(int id, string name, string password, double balance).
5. Add to admin class:
    a. void addClient(Client& client).
    b. Client* searchClient(int id).
    c. void listClient().
    d. void editClient(int id, string name, string password, double balance).
    e. void addEmployee(Employee& employee).
    f. Employee* searchEmployee(int id).
    g. void editEmployee(int id, string name, string password, double salary).
    h. void listEmployee().

6. Create Parser class to read string line from and split this string to (id, name, password, balance or salary) contains:
    a. static vector<string> split(string line).
    b. static Client parseToClient(string line).
    c. static Employee parseToEmployee(string line).
    d. static Admin parseToAdmin(string line).

7. Create FilesHelper save and get from txt files contains:
    a. ✓ static void saveLast(string fileName, int id)
    b. ✓ static int getLast(string fileName).
    c. ✓ static void saveClient(Client c).
    d. ✓ static void saveEmployee(string fileName, string lastIdFile, Employee e).
    e. ✓ static void getClients().
    f. ✓ static void getEmployees().
    g. ✓ static void getAdmins().
    h. ✓ static void clearFile(string fileName, string lastIdFile).

Phase 3 :

1. Create ClientManger class contains:
    a. ✓ static void printClientMenu().
    b. ✓ static void updatePassword(Person* person).
    c. ✓ static Client* login(int id, string password).
    d. ✓ static bool clientOptions(Client* client).

2. Create EmployeeManager class contains:
    a. static void printClientMenu().
    b. static void newClient(Employee* employee).
    c. static void listAllClients(Employee* employee).
    d. static void searchForClient(Employee* employee).
    e. static void editClientInfo(Employee* employee)
    f. static Client* login(int id, string password).
    g. static bool employeeOptions(Client* client).

3. Create AdminManager class contains:
    a. static void printClientMenu().
    b. static Client* login(int id, string password).
    c. static bool AdminOptions(Client* client).

4. Create Screens class contains:
    a. static void bankName().
    b. static void welcome().
    c. static void loginOptions().
    d. static int loginAs().
    e. static void invalid(int c).
    f. static void logout().
    g. static void loginScreen(int c).
    h. static void runApp().