Prakash Krishnan (W200)

Project 1

Requirements Document

Basic Enterprise Application

Business Requirements-

1. This is a simple application for a business to manage employees, budget and a product list.
2. The Classes include:

Organization Class: This class is an abstraction to hold the list of companies that entity operates across.

Company Class: This class defines the different companies that operate inside the Organization. Each Company can have a list of products, employees or a list of accounts. This is a one to many relationship between Organization and Company.

Employee Class: This class defines each employee and the employees belong to a Company

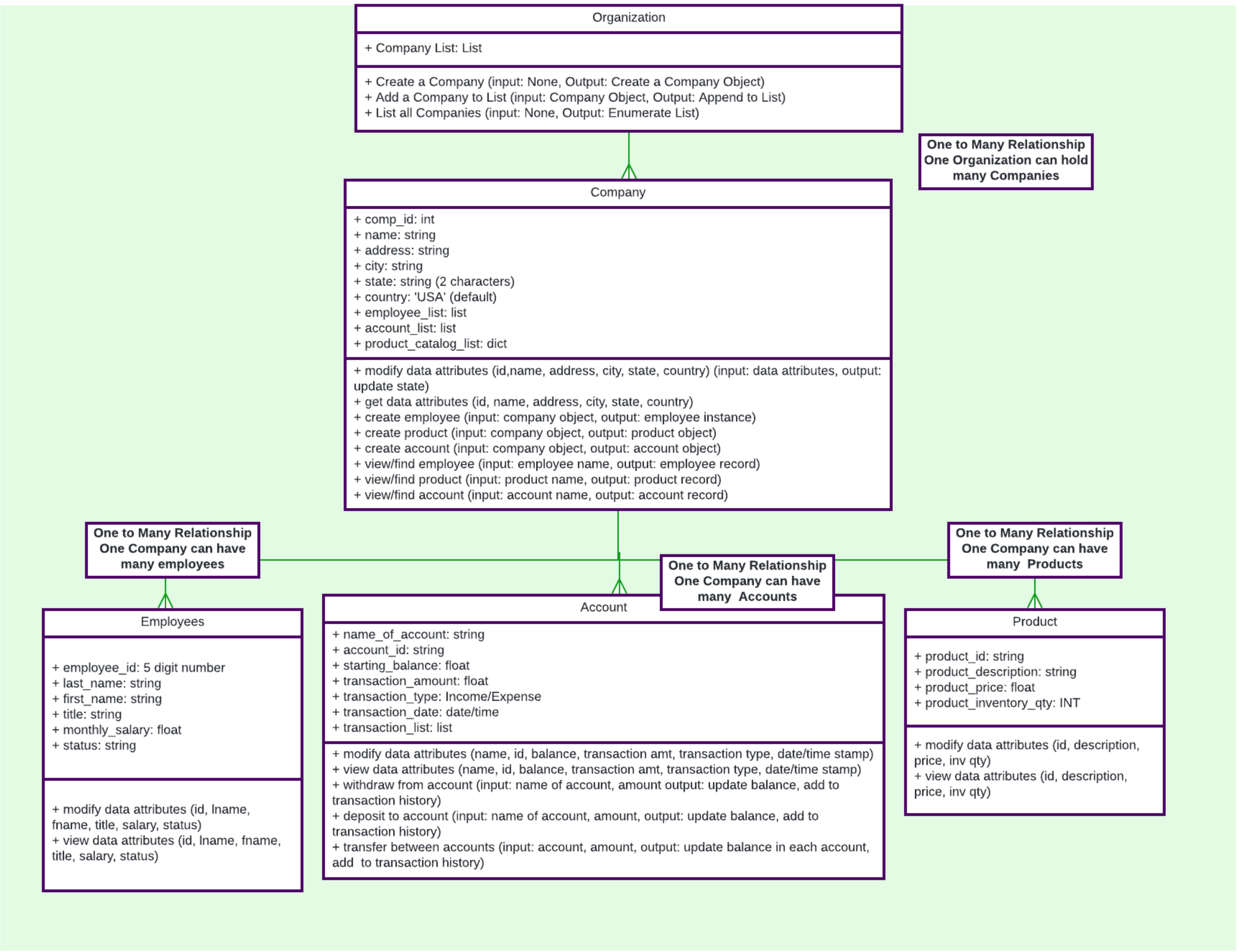
Account Class: This class defines each account that belongs to a company and contains all transactions (withdrawl, deposit and transfer) belonging to the account.

Product Class: This class defines products each Company can sell

Technical Requirements-

1. Leverage Object-Oriented-Programming Design with 5 classes or more
2. Menu driven and interactive
3. Maximum of 750 lines of Code and ideally 300 to 500 lines of code
4. Data attribute protection where appropriate
5. Basic error checking

UML Drawing-


Organization
+ Company List: List
+ Create a Company (input: None, Output: Create a Company Object)
+ Add a Company to List (input: Company Object, Output: Append to List)
+ List all Companies (input: None, Output: Enumerate List)
One to Many Relationship
One Organization can hold
many Companies
Company
+ comp_id: int
+ name: string
+ address: string
+ city: string
+ state: string (2 characters)
+ country: 'USA' (default)
+ employee_list: list
+ account_list: list
+ product_catalog_list: dict
+ modify data attributes (id,name, address, city, state, country) (input: data attributes, output: update state)
+ get data attributes (id, name, address, city, state, country)
+ create employee (input: company object, output: employee instance)
+ create product (input: company object, output: product object)
+ create account (input: company object, output: account object)
+ view/find employee (input: employee name, output: employee record)
+ view/find product (input: product name, output: product record)
+ view/find account (input: account name, output: account record)
One to Many Relationship
One Company can have
many employees
One to Many Relationship
One Company can have
many Accounts
One to Many Relationship
One Company can have
many Products
Employees
+ employee_id: 5 digit number
+ last_name: string
+ first_name: string
+ title: string
+ monthly_salary: float
+ status: string
+ modify data attributes (id, lname, fname, title, salary, status)
+ view data attributes (id, lname, fname, title, salary, status)
Account
+ name_of_account: string
+ account_id: string
+ starting_balance: float
+ transaction_amount: float
+ transaction_type: Income/Expense
+ transaction_date: date/time
+ transaction_list: list
+ modify data attributes (name, id, balance, transaction amt, transaction type, date/time stamp)
+ view data attributes (name, id, balance, transaction amt, transaction type, date/time stamp)
+ withdraw from account (input: name of account, amount output: update balance, add to transaction history)
+ deposit to account (input: name of account, amount, output: update balance, add to transaction history)
+ transfer between accounts (input: account, amount, output: update balance in each account, add to transaction history)
Product
+ product_id: string
+ product_description: string
+ product_price: float
+ product_inventory_qty: INT
+ modify data attributes (id, description, price, inv qty)
+ view data attributes (id, description, price, inv qty)

<u>Class Descriptions</u>

<u>Company Class:</u>

- Company ID: 5 Digit Number (Read Only)
- Company_Name: String (Read/Modify)
- Company_Address: String (Read/Modify)
- Company_City: String (Read/Modify)
- Company_State: String (2 character) (Read/Modify)
- Company_Country: USA (Default)
- Employee_List: Dict/List (Read/Write)

- Product_Catalog_List: Dict/List (Read/Write)
- Customer_List: Dict/List (Read/Write)

Methods-

1. Get Data Attributes (input: company instance, output: display value)
2. Set Data Attributes (input: company instance, output: change value)
3. create employee (input: company name, output: employee instance)
4. create product (input: company name, output: product object)
5. create account (input: company name, output: account object)
6. view/find employee (input: employee name, output: employee record)
7. view/find product (input: product name, output: product record)
8. view/find account (input: account name, output: account record)

Employee Class

- Employee_ID: 5 Digit Number (Read)
- Last_Name: String (Read/Write)
- First_Name: String (Read/Write)
- Title: String (Read/Write)
- Monthly Salary: Float (Read/Write)

Methods-

1. modify data attributes (id, lname, fname, title, salary, status)
2. view data attributes (id, lname, fname, title, salary, status)

Account Class

- Name of Account: String
- Starting Balance: Float
- Transaction Amount: Float
- Transaction Type: Income / Expense
- Transaction Date Stamp: Date and Time
- Transaction List: Store History

Methods-

1. modify data attributes (name, id, balance, transaction amt, transaction type, date/time stamp)
2. view data attributes (name, id, balance, transaction amt, transaction type, date/time stamp)
3. withdraw from account (input: name of account, amount output: update balance, add to transaction history, update balance history)
4. deposit to account (input: name of account, amount, output: update balance, add to transaction history, update balance history)
5. transfer between accounts (input: account, amount, output: update balance in each account, add to transaction history, update balance history)

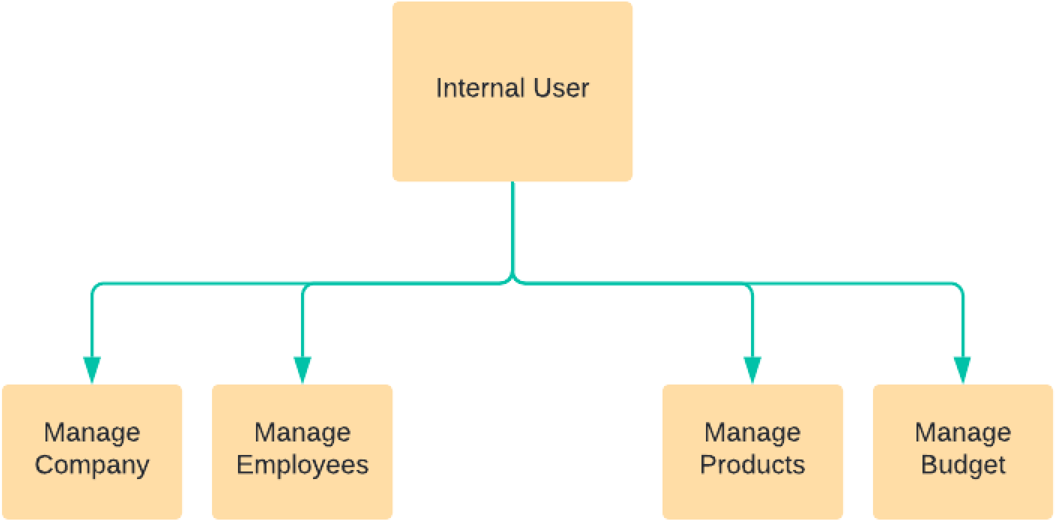
Product Class

- product_id: string
- product_description: string
- product_price: float
- product_inventory_qty: INT

Methods-

1. modify data attributes (id, description, price, inv qty)
2. view data attributes (id, description, price, inv qty)

Menu Structure and Flow


Internal User
Manage Company
Manage Employees
Manage Products
Manage Budget