Object Oriented Design

There's more...?

Outline

- Object Oriented Design
  - Identify the Classes
  - Identify what Information each Class Needs
  - Identify what each Class Needs to Do

OUTLINE TOPIC

Software Development Life Cycle

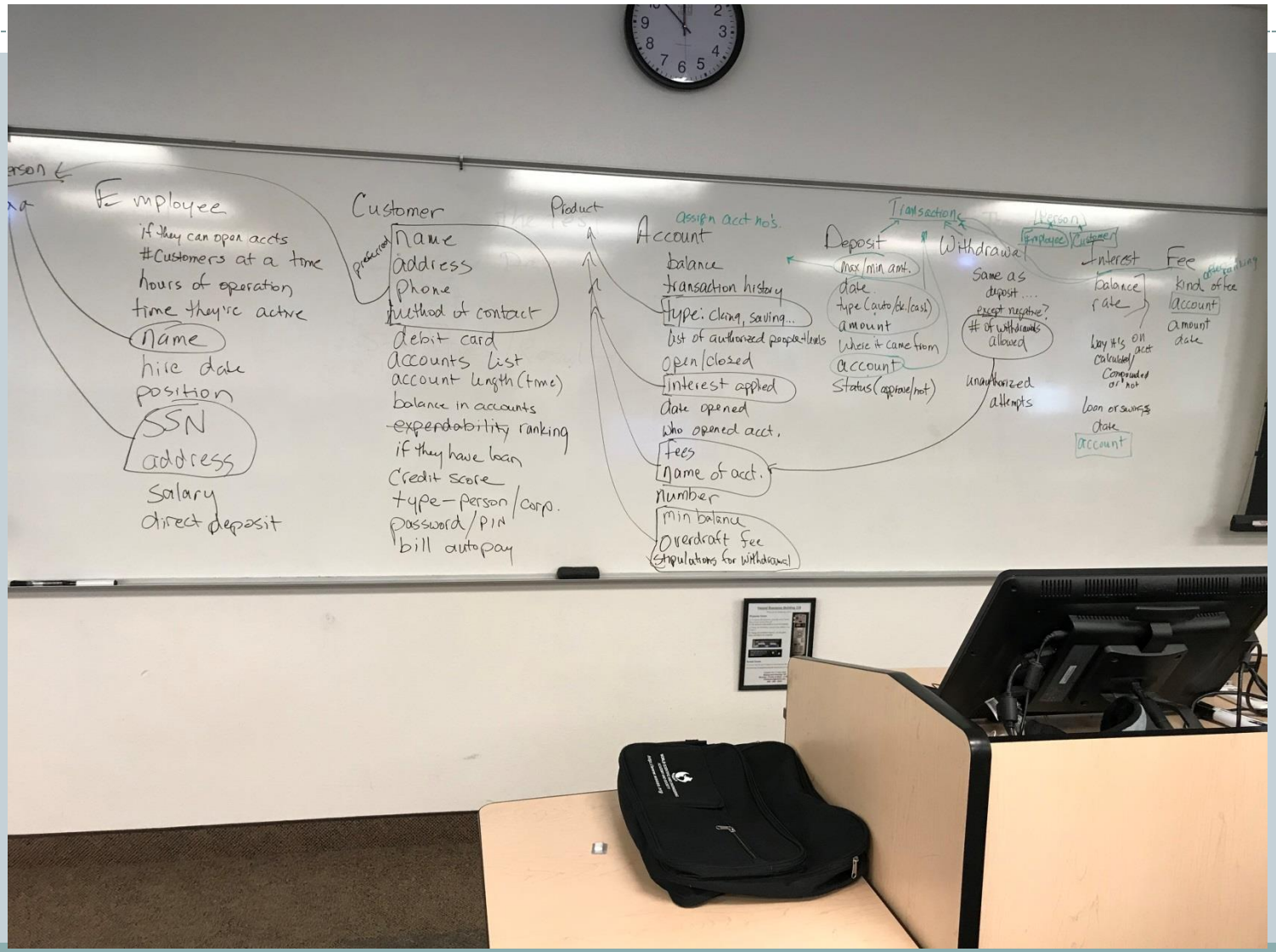
1. Understand the Problem = Requirements Analysis
2. Work out the Logic = Design
3. Convert it to Code = Implementation
4. Test/Debug
5. Maintenance

Today we will talk about requirements analysis and object oriented design.

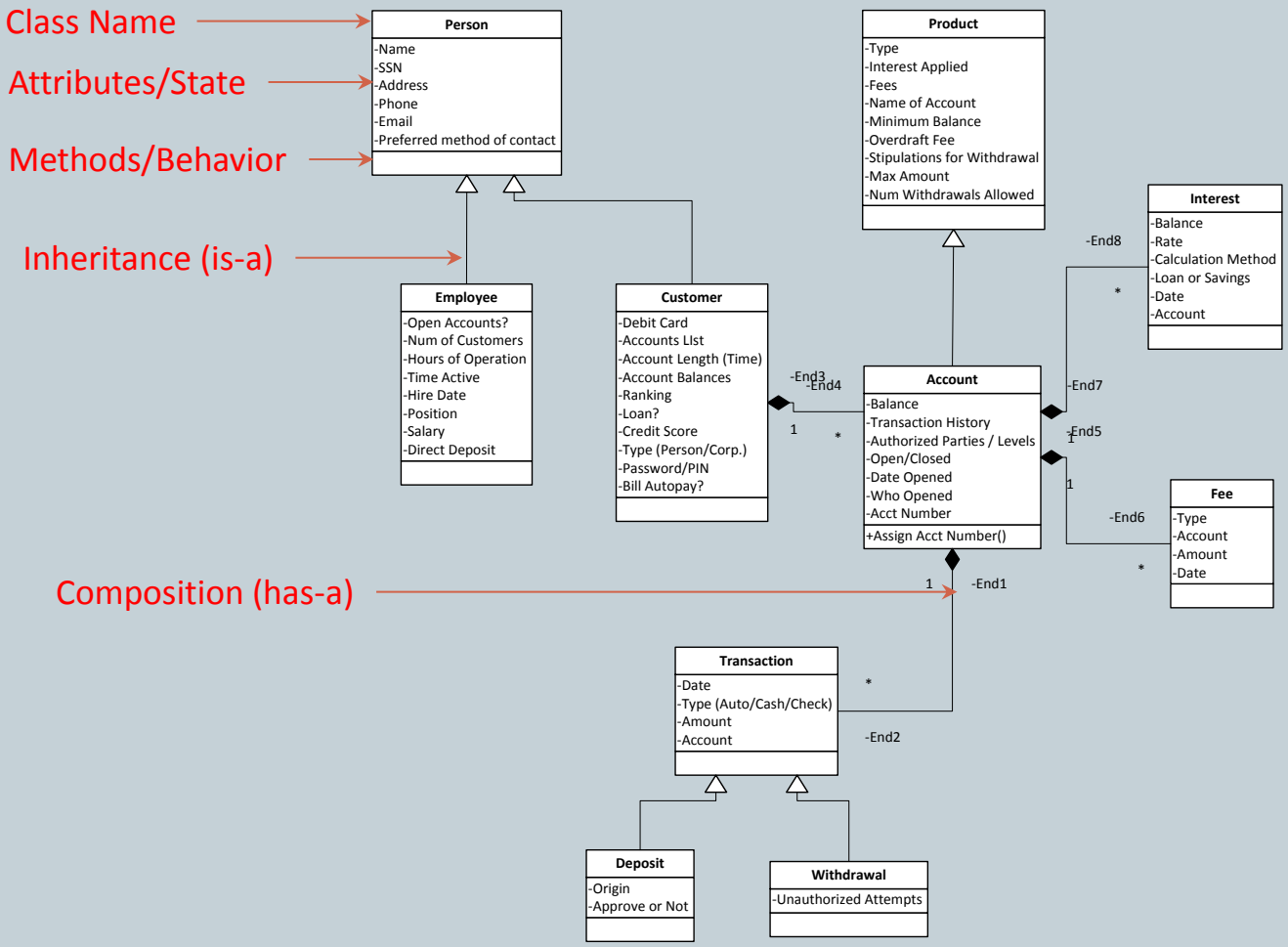
What are the Nouns?

- You have been hired to automate bank operations for a local credit union. They have told you that their business operates as follows:
  - Customers can open accounts. They can make deposits and withdrawals and can close accounts also. On some accounts interest needs to be added, and sometimes fees are deducted.
  - All employees can help customers with deposits and withdrawals. Only some employees are authorized to open and close accounts.

From Monday…


Employee
if they can open accts
#Customers at a time
hours of operation
time they're active
name
hire date
position
SSN
address
salary
direct deposit
Customer
name
address
phone
preferred method of contact
debit card
accounts list
account length (time)
balance in accounts
expendability ranking
if they have loan
Credit score
type – person/corp.
password/PIN
bill autopay
Product
assign acct nos.
Account
balance
transaction history
type: chcng, saving...
list of authorized people+levels
open/closed
interest applied
date opened
who opened acct,
fees
name of acct.
number
min balance
Overdraft fee
stipulations for withdrawal
Transaction
Deposit
max/min amt.
date
type (auto/dr/cash)
amount
where it came from
account
Status (approve/not)
Withdrawal
Same as deposit....
except negative?
# of withdrawals allowed
unauthorized attempts
Interest
balance
rate
how it's on acct
calculated/ Compounded or not
loan or savings
date
account
Fee
kind of fee
account
amount
date

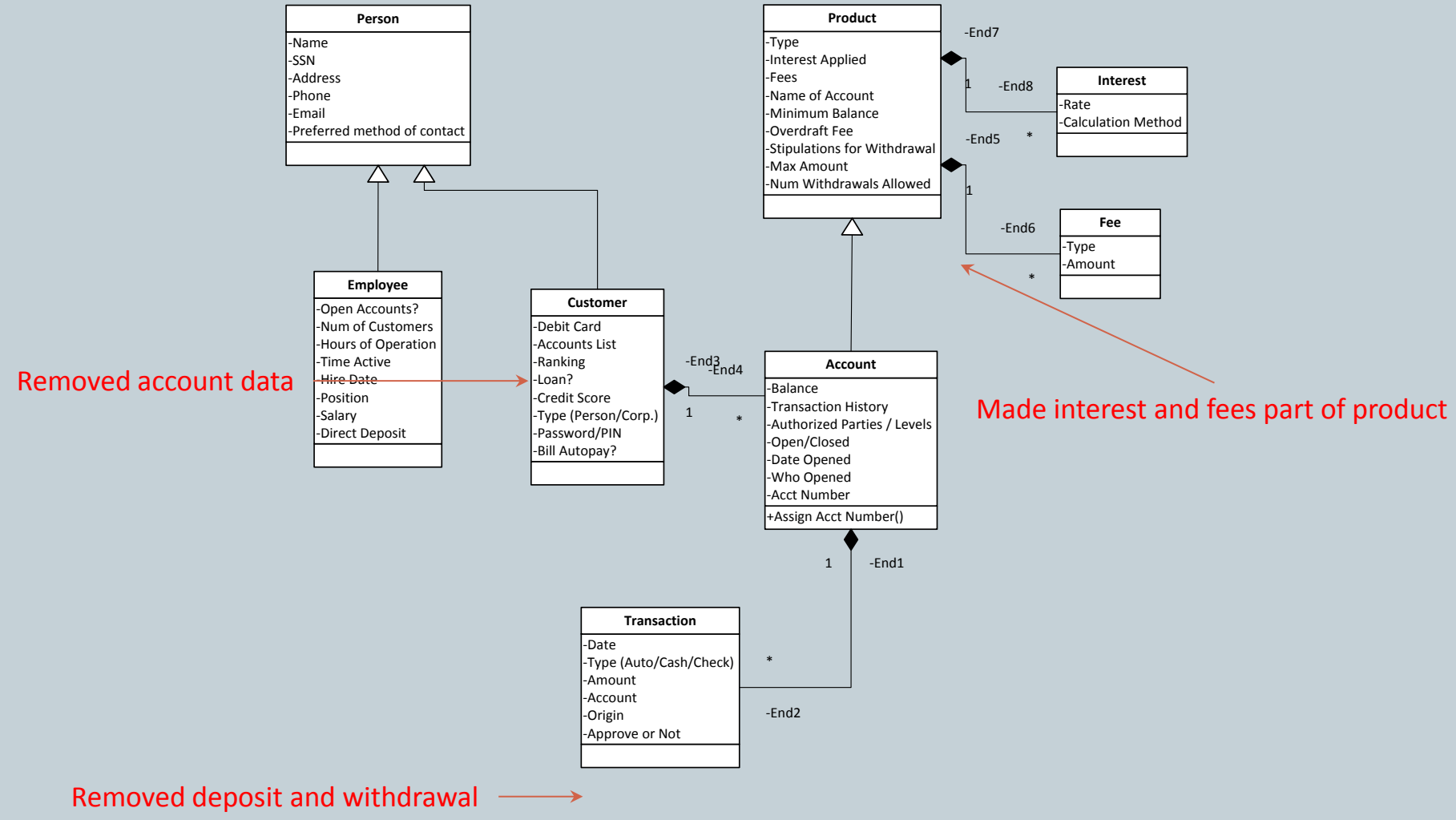
UML Diagram


Class Name
Attributes/State
Methods/Behavior
Inheritance (is-a)
Composition (has-a)
Person
-Name
-SSN
-Address
-Phone
-Email
-Preferred method of contact
Product
-Type
-Interest Applied
-Fees
-Name of Account
-Minimum Balance
-Overdraft Fee
-Stipulations for Withdrawal
-Max Amount
-Num Withdrawals Allowed
Interest
-Balance
-Rate
-Calculation Method
-Loan or Savings
-Date
-Account
Employee
-Open Accounts?
-Num of Customers
-Hours of Operation
-Time Active
-Hire Date
-Position
-Salary
-Direct Deposit
Customer
-Debit Card
-Accounts LIst
-Account Length (Time)
-Account Balances
-Ranking
-Loan?
-Credit Score
-Type (Person/Corp.)
-Password/PIN
-Bill Autopay?
Account
-Balance
-Transaction History
-Authorized Parties / Levels
-Open/Closed
-Date Opened
-Who Opened
-Acct Number
+Assign Acct Number()
Fee
-Type
-Account
-Amount
-Date
Transaction
-Date
-Type (Auto/Cash/Check)
-Amount
-Account
Deposit
-Origin
-Approve or Not
Withdrawal
-Unauthorized Attempts
-End1
-End2
-End3
-End4
-End5
-End6
-End7
-End8
1
*

Simplifying the Design – Classes and Attributes

- Look for repetition of data
  - Try to have each piece of data in only one place
- Look for "modifiers"
  - These might indicate the attribute should be in a different class
    - e.g. Under Customer, we have "Account Length (Time)" and "Account Balance"
      - Since they both reference account, they should probably be in the Account class
- Walk through each attribute and see if it makes sense
  - Does it really applies to that class

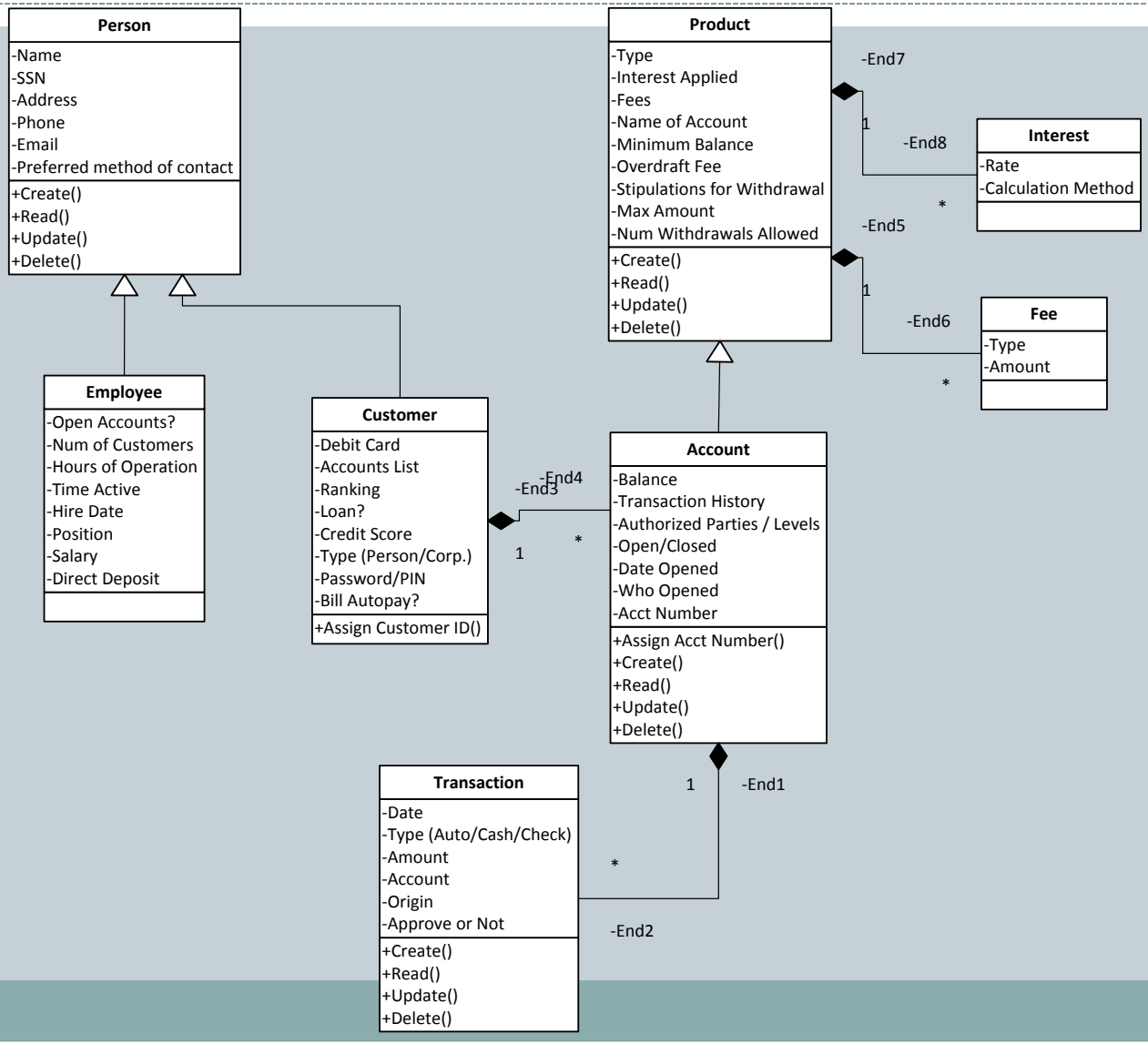
Modified UML



Add Behaviors - What are the Verbs?

- You have been hired to automate bank operations for a local credit union. They have told you that their business operates as follows:
  - Customers can open accounts. They can make deposits and withdrawals and can close accounts also. On some accounts interest needs to be added, and sometimes fees are deducted.
  - All employees can help customers with deposits and withdrawals. Only some employees are authorized to open and close accounts.

UML with Behaviors



More Design

- Use Cases
  - Walk through typical uses of your software and make sure the state and behavior support those cases
- Application Program Interface – API
  - Write an API for the interface to each of your classes
    - For each method, define:
      - Name
      - Input Parameters
      - Return Values
- Define data types for each attribute
  - Might mean splitting a single attribute into several

Implementation

- **Once we are happy with our class definitions, then we get to write some code!!**

Summary

- Object Oriented Design
  - Identify the classes
  - Identify what information each class needs
  - Identify what each class needs to do
  - Identify use cases
  - Define the API
  - Define the instance variables
  - Finally – write some code!