Advanced Java Bootcamp (AJBC) - Phase 1 project

Topics:

1. Java core
2. Inheritance
3. Polymorphism
4. UML
5. GIT
6. AGILE

This document is the SDD (Software Design Document) for the AJBC Online Banking Application.

This document is intended to be a guide to capture the work completed from the Concept phase through the Requirements phase of the development process following the EPRI Software Development Process.

Instructions:

- Please elaborate on each subject.
- If a topic is not applicable to your software, please enter “Not Applicable.”
- Submission and review requirements vary from Software Type & Technical Readiness Level (TRL) – refer to the EPRI Software Development Guidelines for the appropriate process.

-

Software Name:	AJBC BANKING	Revision #:	1
Author:	You		
Date:			
Revision History:		Date:	

Abstract

The AJBC Banking is an application that provides a fully capable online banking service.

It gives the user the ability to create, use and manage a bank account as if it was in a physical branch an in front of a real banker.

The project is defined as an on-going project that should constantly evolve and stand in accordance with the AJBC course.

The heart of the project is OOP, and no compromises are made to keep the OOD principles valid. Therefore, the software design should reflect reusability, scalability and maintainability.

This SDD documents presents a high-level software design for the AJBC Banking software and contains the system over-view, application use cases, functional and non-functional requirements, chosen implementation language, dependencies, versioning, application validation requirements, testing and security requirements, and notes on the enforced coding style.

The aforementioned have been collected in co-operation with AJBC course members and stakeholders, who will also be the first users of the application.

The last two chapters; system architecture and the AJBC data structure, will be updated as plans for the design become more evolved as the course progresses.

AJBC is an open-source project. The source code and documentation in their current form will be released to the public via a web-based version control repository GitHub.

The chosen license for AJBC Toolbox is the GNU Lesser General Public License (LGPL). The documentation, manual and all original graphics will be released with the Creative Commons

1.0 Introduction

Product Description:

- AJBC is a well-known international bank with more than 30 years of banking experience with emphasis on customer satisfaction. This project is the Online banking services of AJBC Bank.
- Online banking is a growing trend in the last decade and AJDB set a goal to provide state of the art mobile and web applications.
- The apps should be super user friendly, efficient, reliable and dependable, accurate, fast.
- The software should provide numerous online banking services including account opening and money transfer.

2.0 Team Members

Alex Batura, Amir Shachar, Anna aba, Ayala Maskalchi, Dana Grosman, Faraj Khanjar, Hodaya David, Ketty Pekarsky, Liran Hadad, Rotem Levi, Shelly Foran, Tanookh Kabishi and Guy Tordjman.

- **Performing Organizations and their Responsibilities**: You are responsible to design and implement the back-end java application. You are also required to provide a CLI as front end.
- **Technical Management and Control**: You are responsible to finish and submit your project for grading at the date and time decided by the project manager.

3.0 Assumptions, Constraints, Schedule and Design

3.1 Assumptions

Assuming you have successfully studied all the materials and principles covered in the first 12 days of the course. You should be able to design and implement the project by the given requirements.

3.2 Constraints

To successfully complete the project, you must follow these constraints:

- OOP programming (Encapsulation, Inheritance, Polymorphism).
- Well organized, well documented, readable, elegant code.
- Separated to packages.
- Create an API style documentation using the Javadoc tool.
- Push at least 10 times to git.
- Your project should reflect a UML class Diagram
- Java minimal version 17.0.1
- Eclipse IDE.
- You may help your colleagues, but do not share your code!
- Push a “Last and Final Commit” message to git before the project submission deadline.

3.3 Schedule

1. DAY 1: 08:00 Project Kick Start.
2. DAY 1: 12:00 UML Class diagram must be submitted in a private SLACK message.
3. DAY 1: 13:00 Project OOD Architecture meeting.
4. DAY 1: 13:30 Implementation Phase 1 Kick Start.
5. DAY 2: 09:30 Scrum Daily
6. DAY 2: 09:45 Implementation Phase 2
7. DAY 2: 13:00 Implementation Phase 3
8. DAY 2: 17:00 Implementation Phase 4
9. DAY 3: 09:30 Scrum Daily
10. DAY 3: 09:45 Implementation Phase 5
11. DAY 3: 13:00 Implementation Phase 6
12. DAY 3: 17:00 Project Submission deadline.
13. DAY 4: 08:00 Project presentation

4.0 General System Description

4.1 System Context

The ADBC OLBS software is a totally self-contained system. Users are Bank account applicants, holders or managers. Applicants can create a bank account. Holders use the banking services of the account and Managers manage the account settings.

4.2 User Characteristics

A user who doesn’t own a bank account is an applicant.

Currently an applicant can open a single bank account but in a future version the number of accounts per a single user would be unlimited.

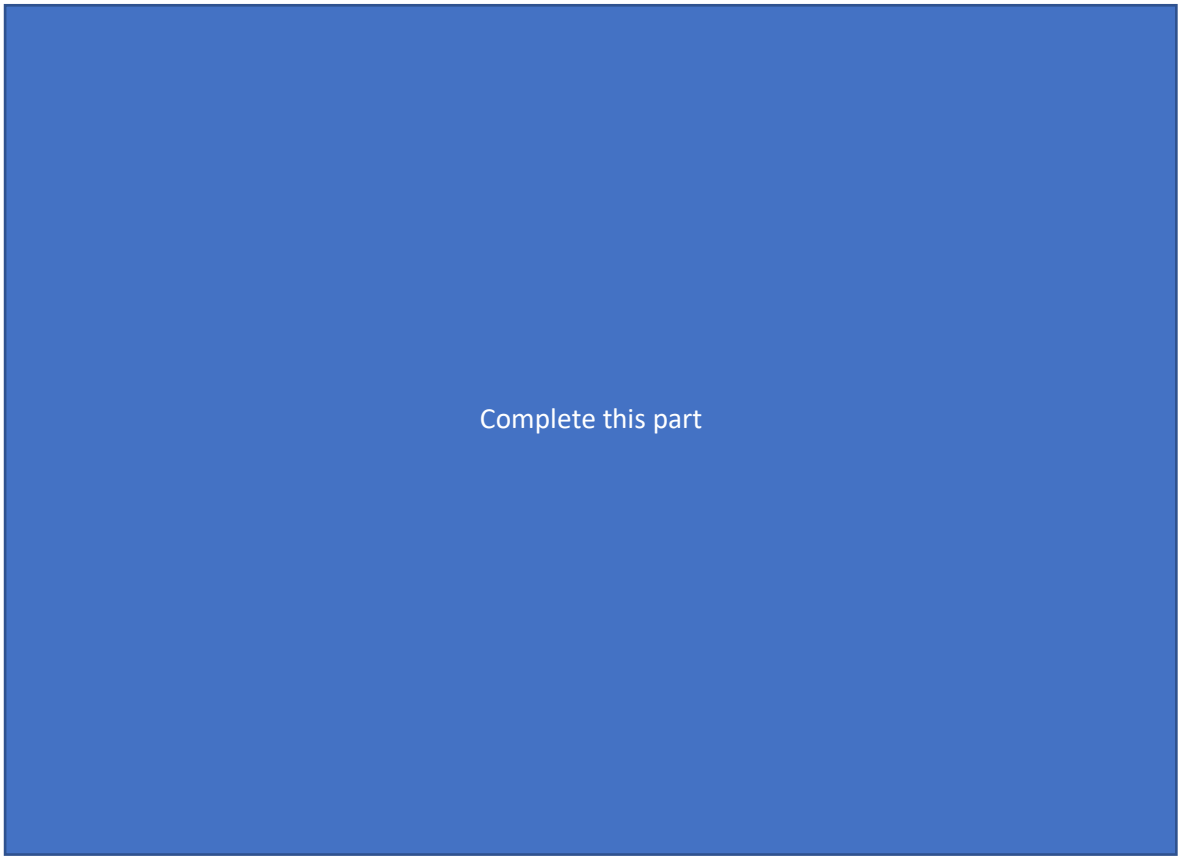
A user who owns a bank account is called and account owner.

An account owner can make unlimited number of transactions including transferring money to other user's accounts.

A bank manager approves the account and sets the interest and operation fees.

A bank manager is the owner of a the AJBC own bank account, it is only used to accumulate its profit from interest and fees.

This is the user inheritance tree:



4.3 Bank Account Types

There are 4 types of bank accounts. They differ in fees, loan and withdrawal amounts and interest rates.

Account Type	Interest rate [%]	Operation fee [NIS]	Max loan amount [NIS]	Max Withdrawal amount (daily) [NIS]
Bronze	**[4.5, 6)**	**[5, 7.5)**	**10,000**	**2,500**
Silver	**[3, 4.5)**	**[3.8, 5)**	**20,000**	**4,000**
Gold	**[1.5, 3)**	**[1.75, 3.8)**	**50,000**	**6,000**
Titanium	**0**	**0**	**NA**	**NA**

Table 1: Account types, their interest and fees

4.4 Bank Account operations

Operations are activities that occur with and related to the funds in the owner's bank account.

The activities are:

1. Deposit cash
2. Deposit Check
3. Withdrawal of cash
4. Pay bill
5. Transfer funds.
6. Withdrawal fee collection

All operations date, time , type and other relevant information are saved and tracked.

5.0 AJBC Use Cases

This chapter describes the main requirements for AJBC as use cases. Use cases describe critical behaviour of the application. The use cases have been defined by the whole project group together in order to find common ground between the developers and the users. This way, the software designers and the people carrying out the implementation can focus on the essential and avoid doing unnecessary work. Figure 1 depicts a high-level use case diagram of the main use cases in Spine Toolbox. It serves as a table of contents for the individual use cases.

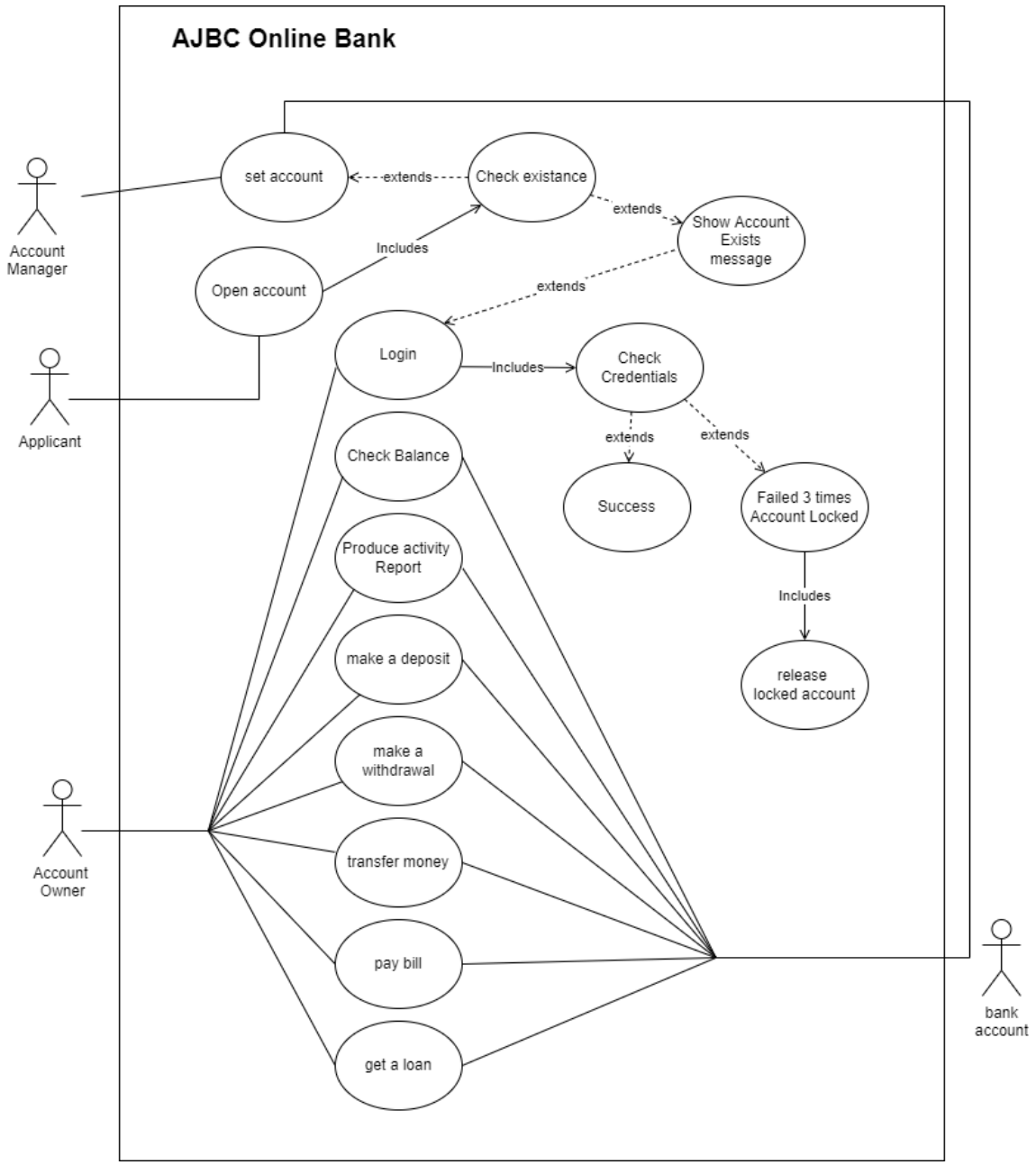


Figure 1: *High-level use case diagram*

5.1 Open Account Use Case

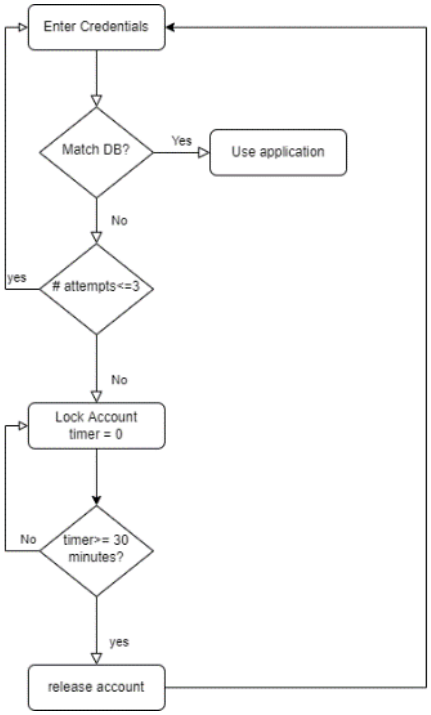
Actors	Applicant
Summary	Open a bank account

1. User selects open a bank account from the menu.
2. User fills an application form (prompt to enter information):
    - Phone number {unique}
    - First name {no digits}
    - Last name {no digits}
    - Date of birth {day, month , year}
    - Username {letters and digits only, unique}
    - Password {4-8 chars, must contain digit and letter}
    - Monthly income {>=0}
3. The system checks if the user has an existing bank account.
4. In case the user already owns a bank account the system prompts it to login instead.
5. The application is waiting for a manager review, approval, and account type setting.

5.2 Login Use Case

Actors	Account Owner
Summary	Login to the banking system

1. User clicks on the logon button (selects login from the menu).
2. User enters username and password for credentials matching in database.
3. In case the username or password doesn’t match the data in the database the system prompts the user to enter the information once again.
4. After 3 attempts the account gets locked for 30 minutes and the user is presented with a blocked account message stating a release time.
5. In case the credential match – the user gets logged in to the system and is presented with the menu.



Flowchart 1: Login use case

5.3 Check Balance Use Case

Actors	Account Owner
Summary	Check account balance

1. A logged in user clicks on the check balance button (selects the option from the menu).
2. User is presented with the account balance
3. The user is represented with the menu.

5.4 Produce Activity Report Use Case

Actors	Account Owner
Summary	The user is displayed with a full report of the account activity within a range of dates

1. A logged in user clicks on the activity report button (selects the option from the menu).
2. AcountOwner enters the query start date.
3. In case The user is an AcountOwner than it gets presented with a sorted list of the account activities starting at the start date until the current date.
4. The AcountOwner is presented with the account balance.

5. The AcountOwner is presented with the loan summary that shows his loan amount, interest, monthly payment and current debt.
6. In case the user is a bankManager than it should be presented with the balance of the bank's bank account and the change in balance since the given date.
7. The user is represented with the menu.

5.5 Make a deposit Use Case

Actors	Account Owner
Summary	User makes a deposit directly to an ATM box using the app authentication code.

1. A logged in user clicks on the make a deposit button (selects the option from the menu).
2. User receives an authentication 4 digits code.
3. The user enters the authentication code and the amount of funds to deposit.
4. The user drops the money in the ATM box.
5. The system registers a successful deposit.
6. The user is represented with the menu.

5.6 Withdrawal Use Case

Actors	Account Owner
Summary	User makes a withdrawal request through the app and get the funds when authenticated in an ATM.

1. A logged in user clicks on the make a withdrawal button (selects the option from the menu).
2. The user enters the amount of funds to withdraw.
3. In case the amount exceeds the daily max, the user is prompt with a message and the operation terminates.
4. The user gets the funds from the ATM.
5. The system registers a successful withdrawal.
6. The user is represented with the menu.

5.7 Transfer funds Use Case

Actors	Account Owner
Summary	User makes a fast transfer to another user by providing a phone number and an amount.

1. A logged in user clicks on the make a transfer button (selects the option from the menu).
2. The user enters the receiving user phone number and amount to transfer.
3. In case the amount exceeds 2000NIS, the user is prompt with a message and the operation terminates.

4. The system locates the receiving user by its phone number.
5. In case the receiving user is not found, the user is prompt with a message and the operation terminates.
6. The system registers a successful deposit to the receiving account.
7. The system registers a successful withdrawal from the sending account.
8. The user is represented with the menu.

5.8 Pay bill Use Case

Actors	Account Owner
Summary	User makes a payment to a payee.

1. A logged in user clicks on the pay a bill button (selects the option from the menu).
2. A payee can only be the AJBC bank itself (loan return) and the phone, water or electric company.
3. The user selects the payee and enters the bill amount.
4. In case the amount exceeds 5000NIS, the user is prompt with a message and the operation terminates.
5. The system registers a successful withdrawal from the sending account.
6. The user is represented with the menu.

5.9 Get Loan Use Case

Actors	Account Owner
Summary	User makes a fast transfer to another user by providing a phone number and an amount.

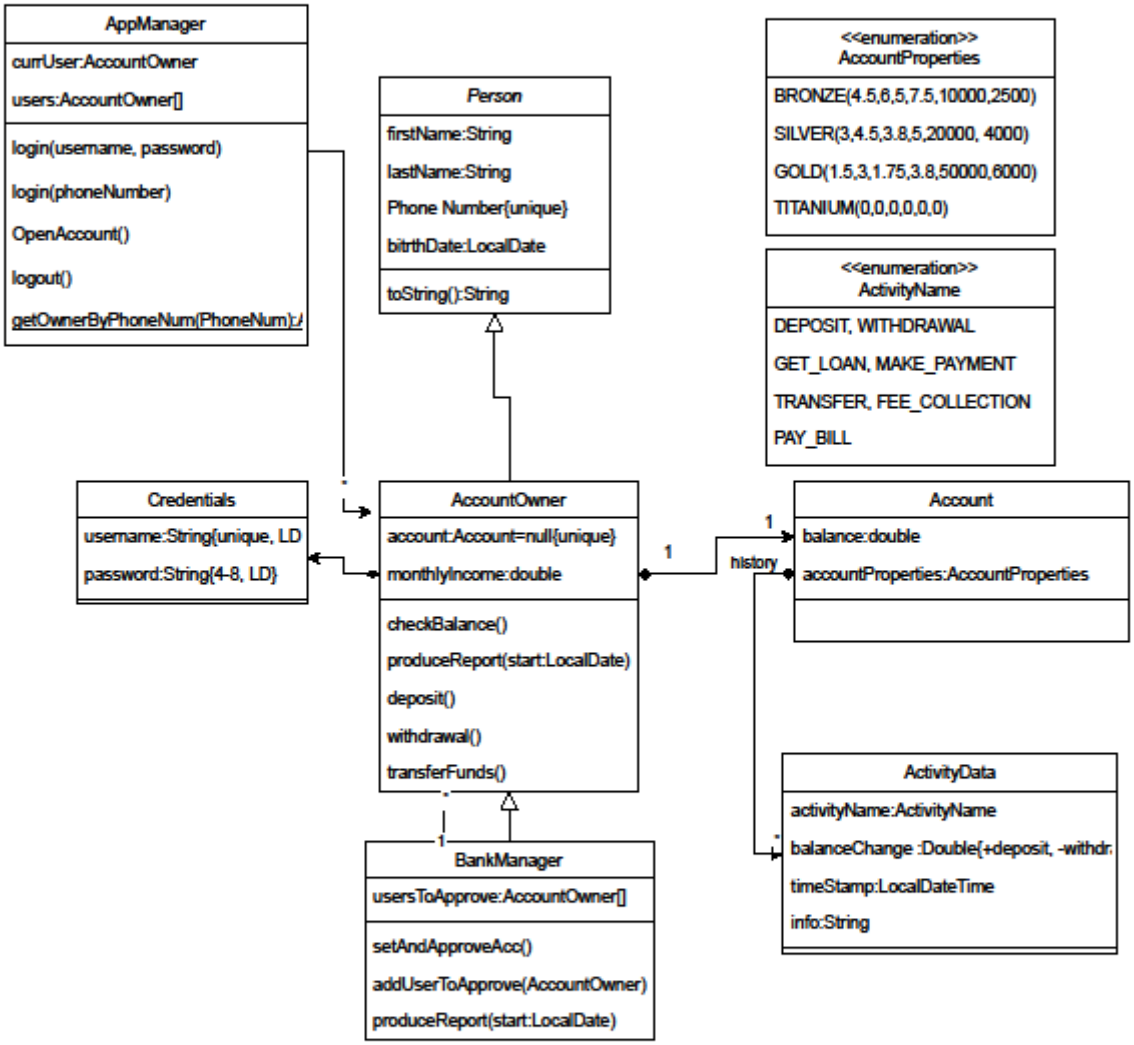
1. A logged in user clicks on the get a loan button (selects the option from the menu).
2. The user enters the requested loan amount and number of monthly payments.
3. In case the amount exceeds the max loan amount, the user is prompt with a message and the operation terminates.
4. In case the number of payments exceeds sixty, the user is prompt with a message and the operation terminates.
5. The user is presented with the amount of the monthly return.
6. The system registers a successful deposit to the account.
7. The system registers a successful withdrawal from the main bank account.
8. The user is represented with the menu.

6.0 System Architecture

This chapter presents the architecture of the system

Your job is to design the class diagram of the project.

You need to get it approved by me before you can start the implementation.


AppManager
currUser:AccountOwner
users:AccountOwner[]
login(username, password)
login(phoneNumber)
OpenAccount()
logout()
getOwnerByPhoneNum(PhoneNum):
Person
firstName:String
lastName:String
Phone Number{unique}
bitrthDate:LocalDate
toString():String
<<enumeration>>
AccountProperties
BRONZE(4.5,6,5,7.5,10000,2500)
SILVER(3,4.5,3.8,5,20000, 4000)
GOLD(1.5,3,1.75,3.8,50000,6000)
TITANIUM(0,0,0,0,0,0)
<<enumeration>>
ActivityName
DEPOSIT, WITHDRAWAL
GET_LOAN, MAKE_PAYMENT
TRANSFER, FEE_COLLECTION
PAY_BILL
Credentials
username:String{unique, LD
password:String{4-8, LD}
AccountOwner
account:Account=null{unique}
monthlyIncome:double
checkBalance()
produceReport(start:LocalDate)
deposit()
withdrawal()
transferFunds()
Account
balance:double
accountProperties:AccountProperties
history
BankManager
usersToApprove:AccountOwner[]
setAndApproveAcc()
addUserToApprove(AccountOwner)
produceReport(start:LocalDate)
ActivityData
activityName:ActivityName
balanceChange :Double{+deposit, -withdr
timeStamp:LocalDateTime
info:String