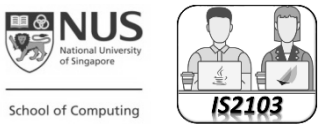

IS2103 – Enterprise Systems Server-side Design and Development

AY 2019/20 Semester 1

Practical Lab 03 – Enterprise JavaBeans (I)

ejb has its own exception handling

Part 1 – Basic Programming

Recall that in Practical Lab 02, we have discussed about the implementation of the retail commerce distributed computing case study using **Java RMI (Remote Method Invocation)**. Review your own implementation and compare it against the suggested answer that can be downloaded from LumiNUS.

Thereafter, implement the case study again using **Java EE (Java Platform, Enterprise Edition) EJB (Enterprise JavaBeans)**. As usual, you do not need to implement the actual business processing logic. It will suffice the requirement by simply printing out some text messages.

When you are done, please answer the following questions:

a. Explain the general similarities and differences between the Java RMI implementation and Java EE EJB implementation. Which implementation technique do you prefer?

b. In the case study, it is assumed that the CardProcessing component is a local object. In all likelihood, the CardProcessing component could be deployed separately from the Checkout component or a remote client may need to utilise its services directly.

   Compare the similarities and differences in how you look up a local object using Java RMI and Java EE EJB. Is the Java EE EJB approach a better one? Briefly explain your answer.

Part 2 – Intermediate Programming

Recall that in Practical Lab 01, we have discussed about Version 3 of the retail **Point-of-Sale (POS) system**. Version 3 of the POS system is a Java EE application exhibiting a three-tier architecture and uses a relational database for data persistence.

The Java EE application contains the following four EJB session beans:

- StaffEntitySessionBean
- ProductEntitySessionBean
- SaleTransactionEntitySessionBean
- CheckoutSessionBean

At this juncture, you should have already studied the code for each of the four session beans and understood what each one of them is doing.

a. Briefly explain the choice of session bean type for each of the four session beans.

Version 3 of the retail POS system is incomplete with missing system functionalities to support the business use cases. In fact, the retail POS system contains a major business processing logic error that needs to be fixed. With reference to the UML component diagram of Version 3 of the retail POS system shown in Figure 3 of the worksheet for Practical Lab 01, make the following changes to the source files:

b. In the source file PointOfSaleSystemV30.PointOfSaleSystemV30Client.pointofsalesystemv30client.SystemAdministrationModule.java, complete the source code to implement the system functionalities for supporting the following business use cases:

   - Create New Product
   - View Product Details
   - Update Product
   - Delete Product

c. PointOfSaleSystemV30.PointOfSaleSystemV3-ejb.dao.ProductEntityManager.java currently uses hardcoding in the retrieval of product records. Change the source file to retrieve the product records from the relational database.

d. Fix the major business processing logic error in the implementation of the system functionalities for supporting the business use cases Checkout, and Void or Refund.

Part 3 – Case Development (Ungraded Homework)

Merlion Bank is a new retail bank that will be opening in Singapore soon. You have been asked to develop a **Retail Core Banking System (RCBS)** for Merlion Bank that will consist of i) a core backend to be developed with a component-based architecture; and ii) multiple retail banking applications to support the business operation of the bank. The RCBS is to be developed in multiple phases over a period of 9 weeks.

In the first phase, you are required to create the basic NetBeans project structure for the RCBS together with two enterprise application clients representing the Teller Terminal application and Automated Teller Machine (ATM) application. Your NetBeans enterprise application project structure should resemble the following:

- RetailCoreBankingSystem:
  - RetailCoreBankingSystem-ejb
  - RetailCoreBankingSystemLibrary → existing and remote interface
  - TellerTerminalClient
  - AutomatedTellerMachineClient

The logical data model of the RCBS is depicted in the UML class diagram shown in Figure 1. You do not need to implement the entire logical data model at this juncture. Only those entity classes that are required to implement system functionalities for supporting the business use cases to be delivered in the current phase need to be implemented.

For this phase, you are required to implement system functionalities to support the business use cases depicted in the UML use case diagram as shown in Figure 2. A brief description of each business use case is given in Table 1 below.

S/N	Use Case	Description/Business Rules
1	Create Customer	• For a new customer, teller needs to create a new customer record first before performing any other business activities. • Each customer should be uniquely identified with one customer record.
2	Open Deposit Account	• Teller opens a new deposit account for an existing customer. • Customer needs to provide an initial cash deposit of any amount. • A customer can have multiple deposit accounts of different types. • For this phase, only savings account is required. • For this phase, you may assume that there is only individual account, i.e., customers cannot open a joint account.
3	Issue ATM Card	• Teller issue a new or replacement ATM card to customer. • For replacement of lost or damaged ATM card, the corresponding record of the previous ATM card should be deleted first. • An ATM card may be associated with one or more deposit accounts. • Each deposit account may be associated with zero or one ATM card.
4	Change PIN	• Customer change current PIN of the ATM card.
5	Enquire Available Balance	• Customer enquires available balance for a deposit account associated with the ATM card. • Available balance refers to the balance amount in a deposit account that is available for spending, withdrawal or transfers. • Ledger balance is equal to the sum of available balance plus any holding balance. • If there is no holding balance, then ledger balance is equal to available balance.

Table 1 – Use case descriptions and business rules of RCBS for the first phase of development.

You are required to design and implement an appropriate component-based architecture for the RCBS and then implement it using the appropriate EJB session beans (refer to Figure 3 for an example). You may use either file object serialisation or relational database to implement data persistence.

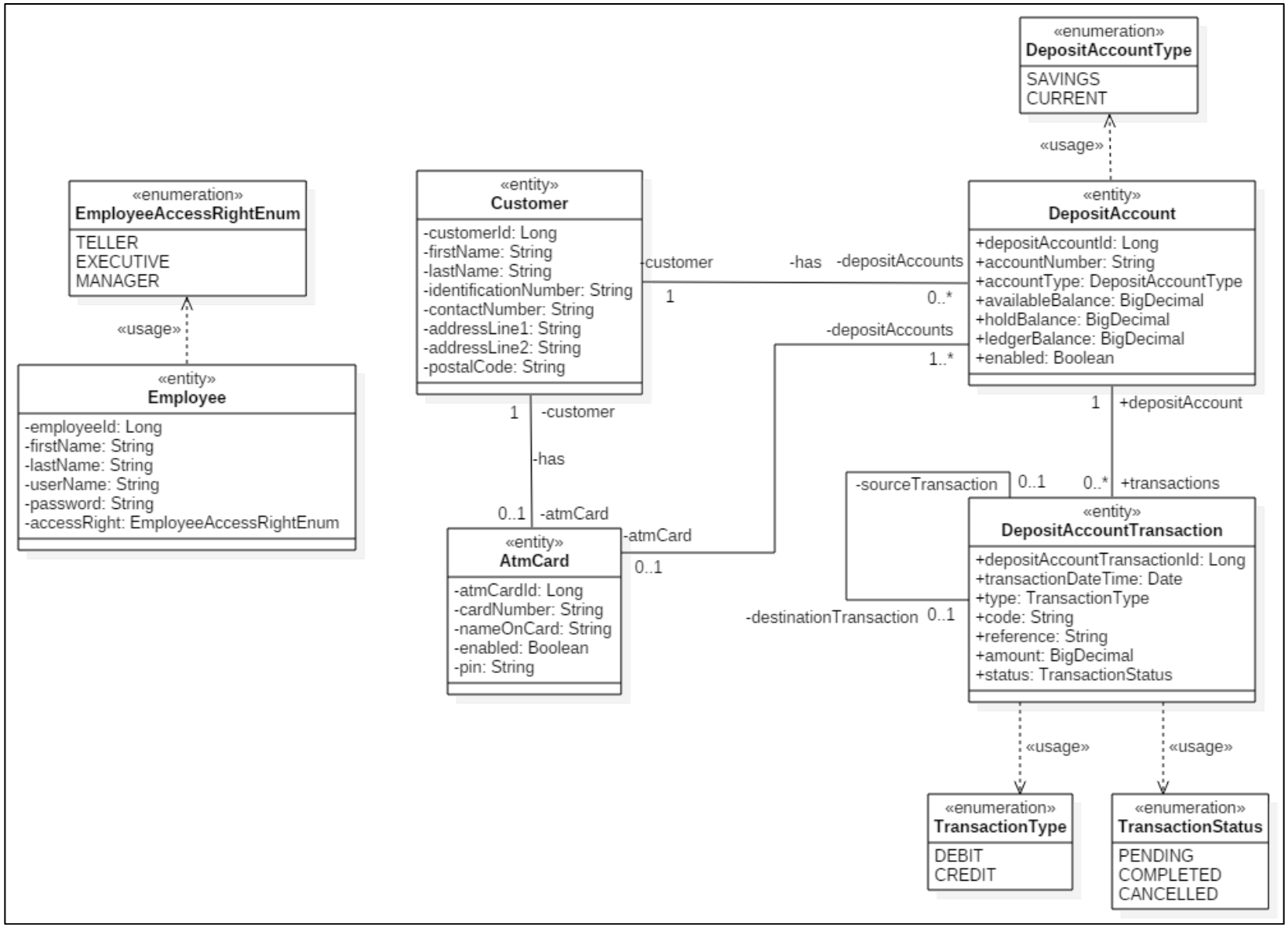

Figure 1 – Logical data model of the RCBS.

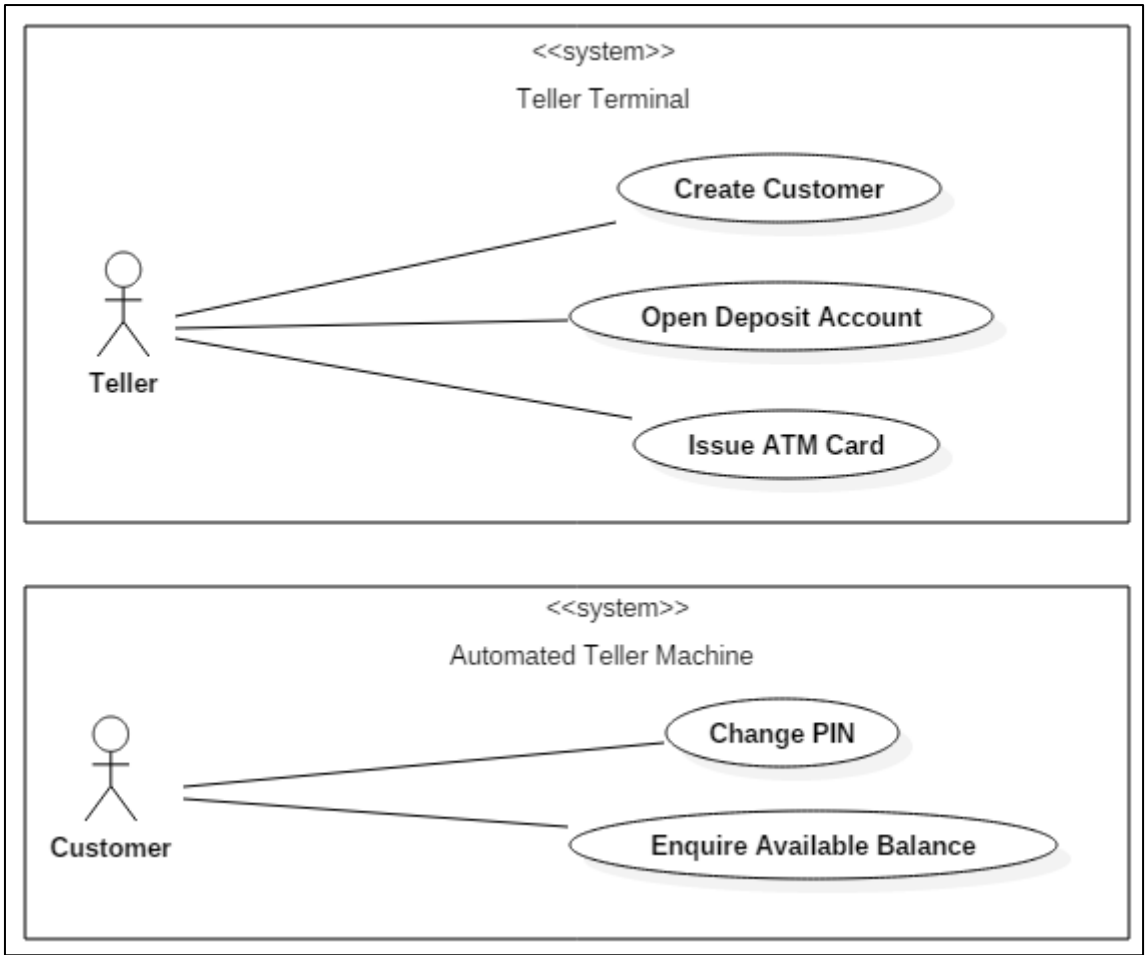

Figure 2 – Use case diagram of the RCBS for the first phase of development.