School of Information Technology
Department of Computer Science

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

COS326 – Database Systems
Practical 2 2024

Release Date:	08 August 2024
Submission Date:	15 August 2024 @ 23:59Hrs
Lecturer:	Mr S.M Makura
Total: 50 Marks	

A. Objectives

Demonstrate that you can use ObjectDB for Java to insert, modify, retrieve and delete objects from an ObjectDB database and that you understand ObjectDB.

In this practical you will have to write a Java program that:

1. Implements Java classes and makes them persistent by storing them in a ObjectDB database.
2. Executes queries on the ObjectDB database.

B. Submission Procedure:

1. Create a folder as follows:
   The folder name should be ***uXXXXXXXX***(*XXXXXXXX is your student number)* and should contain the project files (your.jar, .java and other necessary files needed for your project to run).
2. Create a zip file of the folder and upload it to ClickUP via the submission link provided for practical 2.

NO LATE submissions will be accepted after the submission date and time has lapsed. Do not wait till the last minute to submit and start giving excuses that you faced technical challenges when you tried to submit.

Question 1: ObjectDB Database Programming

Scenario

Note: This practical builds upon the concepts covered in practical 1.

After completing your first task as a software engineer at FinGuard Solutions, the lead developer at the company was very impressed by the Java application that you developed. Now they would like you to develop another application in order to show relationships between the entity objects. They have provided you with the following information:

JPA Annotations for Relationships

Relationships are persistent fields in persistable classes that reference other entity objects. The four relationship modes are represented by the following annotations:

javax.persistence.ManyToMany

javax.persistence.ManyToOne

javax.persistence.OneToMany

javax.persistence.OneToOne

You can read more about these annotations here:

https://www.objectdb.com/api/java/jpa/annotations/relationship

Source: Makura S.M (2024) & ObjectDB User Manual (2024)

In this practical, you will implement one of the relationships using one of the annotations depicted above. Now we know that each bank account can have many transactions. Based on this information, create a Java application using NetBeans or any Java IDE you are comfortable with. The Java application must interact with an ObjectDB database and must have the following files/capabilities:

Requirements

a) Consist of two entity classes, namely **BankAccount** and **Transaction**:

The `BankAccount` class should set or get the account number and account holder's name. In the `Transaction` class, you will need to define a variable of type `BankAccount` and utilise one of the relationship annotations mentioned in the scenario above. Then include the getters and setters for the `BankAccount` variable. Note, you must select the appropriate relationship annotations here. One for the `BankAccount` class and the other for the

`Transaction` class. If you choose the wrong annotations, you will lose marks.

b) Must have a GUI where you can perform CRUD operations on the ObjectDB database. You will also need a main class where you will implement all the methods necessary to perform the CRUD operations. The CRUD operations expected are:

i. **Create/Store** – The bank account details (account number, account holder's name) and the transaction details (transaction ID, transaction date, amount, sender account number, receiver account number, transaction type). The details must then be saved in the ObjectDB database through a "Save" button. You are welcome to either add the transactions one by one or all together at once. Use an appropriate Java Swing/JavaFX GUI control to display a message to confirm if the operation has been done successfully.

ii. **Read** the bank account details from the ObjectDB database and display all the transactions associated with that bank account via a GUI interface. Have a search button to search by the account number and use an appropriate Java Swing/JavaFX GUI control to display the results (i.e., the account number, account holder's name, and the associated transactions).

iii. **Delete** the specified bank account details based on the account number. Have a delete button which, when clicked after entering the account number, will delete the bank account and the transactions associated with that account. Use an appropriate Java Swing/JavaFX GUI control to display a message to confirm if the operation has been done successfully.

iv. The update operation is **not** required.

Ensure that your application handles any basic exceptions (i.e., error checking, validation checks, etc.). Full marks will be awarded for a fully functional Java application based on specification.

[Total Marks: 50]