



Banking APP

With JDBC



CONTENT

- 
- | | |
|----|--------------------------------|
| 01 | JAVA |
| 02 | JDBC Connections |
| 03 | SCOPE OF PROJECT |
| 04 | PROJECT CONFIGURATIONS |
| 05 | DEPENDENCIES |
| 06 | FUNCTIONAL REQUIREMENTS |
| 07 | OPERATING ENVIRONMENT USED |
| 08 | HIERARCHY OF PROJECT ARTIFACTS |

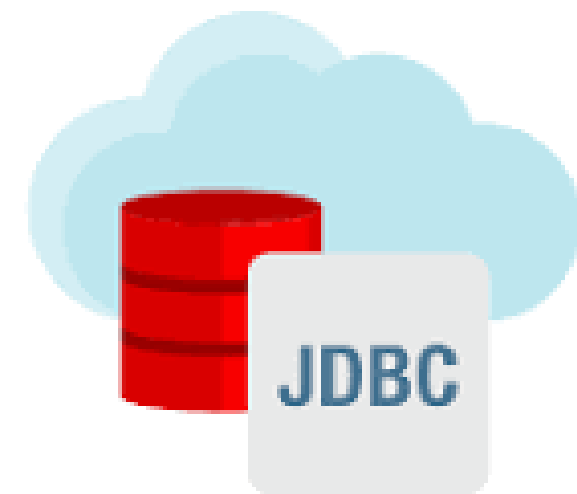
Java

- Java is a popular object-oriented programming language
- known for its strong syntax, extensive libraries, and platform flexibility. Its WORA principle allows programs to run on any device with the JVM installed. Java EE and SE provide enterprise-level and general-purpose computing frameworks. IDEs like Eclipse and JavaFX speed up coding.



JDBC

Java Database Connectivity (JDBC) is a crucial component of Java's functionality for interacting with relational databases. It provides a standardized interface for database communication, enabling SQL queries, updates, and result retrieval in Java applications. JDBC is platform-neutral and database agnostic, with connection pooling and exception handling features for efficient database operations.





SCOPE OF PROJECT

The Bank App aims to provide a comprehensive solution for managing financial transactions in a banking environment. It encompasses various aspects such as transaction processing, verification based on established guidelines, classification into "VALID" and "INVALID" categories, communication with a MySQL database for storing and retrieving transaction data, and the display of a summary of valid and invalid transactions for easy monitoring.



Project CONFIGs

- Database Connection Configuration
- Connection String like JDBC and URL
- Project Specific Configurations

DEPENDENCIES

- MySQL-connector: The MySQL database server can communicate with applications thanks to this software component.
- The drivers or libraries offered by MySQL Connector make connection and communication procedures easier.
- MySQL Driver: It serves as an interface or bridge that connects programs and computer languages to MySQL databases.
- Give the DriverManager's JDBC `URL.getConnection()`; with the MYSQL DB information Such as username and password.

COMPONENTS

BTransationProcessor:

Primary class responsible for
managing bank transactions
and run entire functionality

BASE CLASS

Connection Provider Class:

Class responsible for managing
database connection.

Other Classes

Utility Methods

Method:

- `createStatement()`

Creates a statement object

Method 1

Method:

-

Method 2

- `getConnection`
method responsible for
obtaining database
connection.

- `closeConnection`
method responsible for
closing the database
connection

Method 2

Method CHAINING

Programming technique known as "method chaining" entails invoking several methods on an object within a single, continuous line of code.

- Every method in a method chain is called based on the outcome of the one before it, and the process keeps going until the intended tasks are finished.
- It enhances code readability and conciseness, making complex operations more straightforward.

REQUIREMENT ID	REQUIREMENT CATEGORY	REQUIREMENT TYPE	PRIORITY	HIERARCHY	REF
R001	FUNCTIONAL	STATED	HIGH		

REQUIREMENT DESCRIPTION	ESTABLISHING DATABASE CONNECTION
SCOPE	This functionality involves establishing a connection to the MySQL database to facilitate interactions with transaction records.
REQUIREMENT METHODOLOGICAL DETAILS	<p>Implementation Details: Uses JDBC to load the MySQL driver and establish a connection. Ensures the connection is created only if it does not already exist.</p> <p>Considerations: Handles exceptions related to database connection, printing stack traces if necessary.</p>

REQUIREMENT ID	REQUIREMENT CATEGORY	REQUIREMENT TYPE	PRIORITY	HIERARCHY	REF
R001	FUNCTIONAL	STATED	HIGH		

REQUIREMENT DESCRIPTION	UPDATING TRANSACTION RECORDS
SCOPE	This functionality involves updating transaction details, such as new balance and transaction status, in the database.
REQUIREMENT METHODOLOGICAL DETAILS	<p>Parameters:</p> <p>NewBal: New balance after the transaction.</p> <p>Validity: Transaction validity status (Valid/Invalid).</p> <p>TransID: Unique identifier for the transaction.</p> <p>Implementation Details:</p> <p>Prepares and executes an SQL update statement using JDBC.</p> <p>Updates the NewBal and TransStat columns in the transactions table.</p>

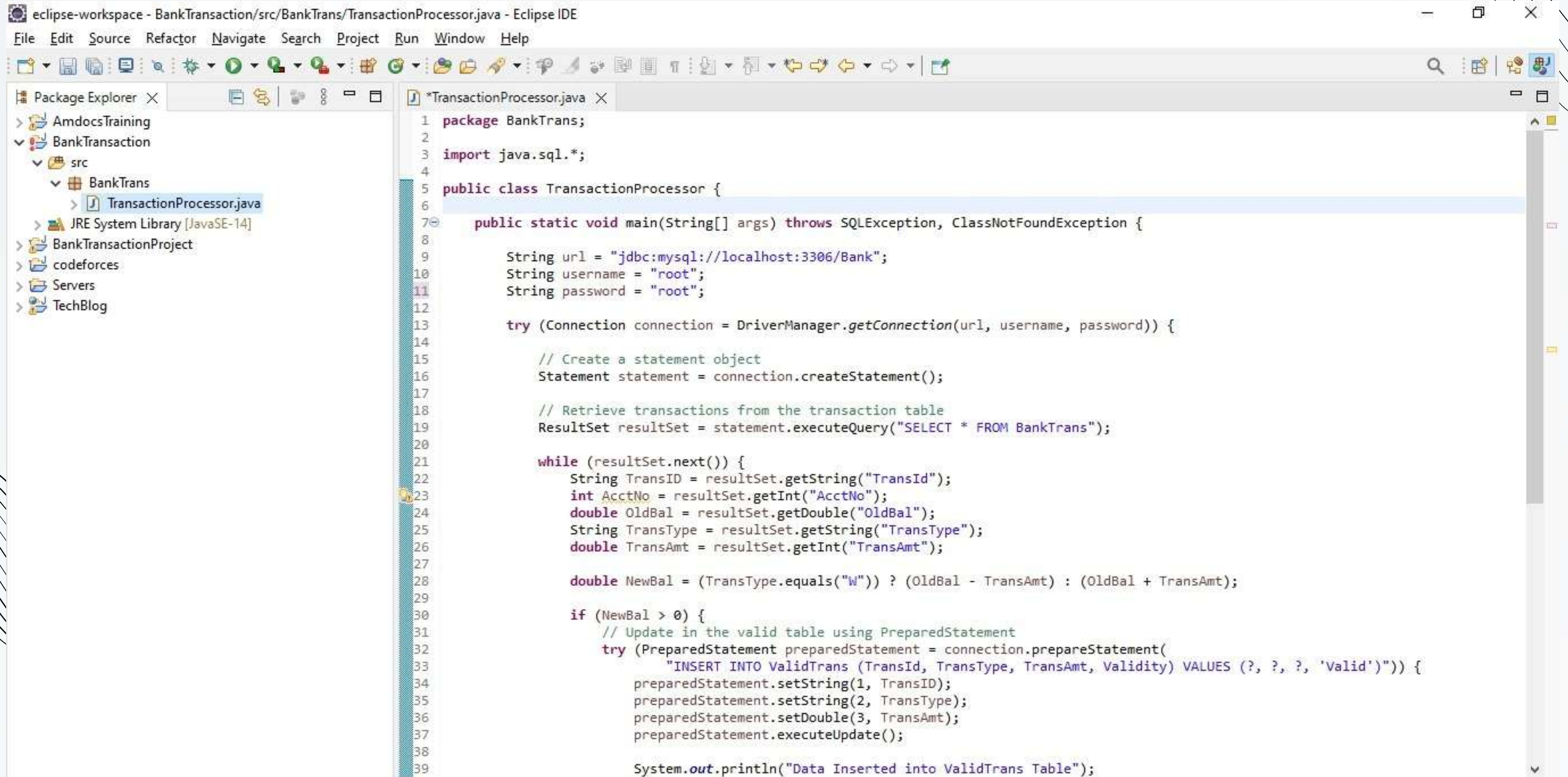
REQUIREMENT ID	REQUIREMENT CATEGORY	REQUIREMENT TYPE	PRIORITY	HIERARCHY	REF
R001	FUNCTIONAL	STATED	HIGH		

REQUIREMENT DESCRIPTION	INSERTING TRANSACTION RECORDS
SCOPE	This functionality involves inserting transaction records into appropriate tables based on their validity.
REQUIREMENT METHODOLOGICAL DETAILS	<p>Parameters:</p> <p>tableName: Name of the table (ValidTrans/InvalidTrans).</p> <p>transID, transType, transAmt: Transaction details.</p> <p>validity: Transaction validity status (Valid/Invalid).</p> <p>Implementation Details:</p> <p>Determines the appropriate SQL query based on the table name.</p> <p>Prepares and executes the insertion statement using JDBC.</p>

REQUIREMENT ID	REQUIREMENT CATEGORY	REQUIREMENT TYPE	PRIORITY	HIERARCHY	REF
R001	FUNCTIONAL	STATED	HIGH		

REQUIREMENT DESCRIPTION	DISPLAYING TRANSACTION RECORDS
SCOPE	This functionality involves displaying all transaction records from the transactions, ValidTrans and InvalidTrans.
REQUIREMENT METHODOLOGICAL DETAILS	<p>Implementation Details : Executes SQL queries to fetch records from the respective tables. Prints the retrieved records to the console in a formatted manner.</p> <p>Considerations : Handles SQL exceptions and prints stack traces in case of errors.</p>

Hierarchy of Project Artifacts



The screenshot displays the Eclipse IDE interface. On the left, the Package Explorer shows the project hierarchy: AmdocsTraining, BankTransaction (expanded), src, BankTrans (expanded), and TransactionProcessor.java (selected). The main editor window shows the source code of TransactionProcessor.java. The code is as follows:

```
1 package BankTrans;
2
3 import java.sql.*;
4
5 public class TransactionProcessor {
6
7     public static void main(String[] args) throws SQLException, ClassNotFoundException {
8
9         String url = "jdbc:mysql://localhost:3306/Bank";
10        String username = "root";
11        String password = "root";
12
13        try (Connection connection = DriverManager.getConnection(url, username, password)) {
14
15            // Create a statement object
16            Statement statement = connection.createStatement();
17
18            // Retrieve transactions from the transaction table
19            ResultSet resultSet = statement.executeQuery("SELECT * FROM BankTrans");
20
21            while (resultSet.next()) {
22                String TransID = resultSet.getString("TransId");
23                int AcctNo = resultSet.getInt("AcctNo");
24                double OldBal = resultSet.getDouble("OldBal");
25                String TransType = resultSet.getString("TransType");
26                double TransAmt = resultSet.getDouble("TransAmt");
27
28                double NewBal = (TransType.equals("W")) ? (OldBal - TransAmt) : (OldBal + TransAmt);
29
30                if (NewBal > 0) {
31                    // Update in the valid table using PreparedStatement
32                    try (PreparedStatement preparedStatement = connection.prepareStatement(
33                        "INSERT INTO ValidTrans (TransId, TransType, TransAmt, Validity) VALUES (?, ?, ?, 'Valid')")) {
34                        preparedStatement.setString(1, TransID);
35                        preparedStatement.setString(2, TransType);
36                        preparedStatement.setDouble(3, TransAmt);
37                        preparedStatement.executeUpdate();
38
39                        System.out.println("Data Inserted into ValidTrans Table");
40                    }
41                }
42            }
43        }
44    }
45 }
```

THANK YOU

By : Siddhi Soni

