Banking App with JDBC

Trainer - Mr. Sachin Hadap

Name - Amey Patil

16 Jan 2024

AMDOCS

Today's agenda

- Introduction to Java
- JDBC Connections
- Scope of Project
- Operating Environment
- Project Configurations
- Dependencies
- Hierarchy of Project Artifacts
- Conclusion.

Introduction to Java

- **Java** is widely-used programming language that plays a crucial role in developing various applications.

- **Platform Independence:** Ability to run on any device or platform without modification. This "write once, run anywhere" approach ensures portability across different platforms.

- **Object-Oriented Programming:** Java follows an Object-Oriented Programming (OOP) paradigm, which allows us to organize our code into reusable and modular components.

- **Multithreading:** Java supports multithreading, allowing concurrent execution of multiple threads within a single program.

- **Spring Framework:** It offers features like dependency injection, aspect-oriented programming, and easy integration with databases which makes easier to build scalable and maintainable applications.

JDBC Connections

- **JDBC (Java Database Connectivity):** JDBC is a Java-based API that allows Java applications to interact with relational databases.

- JDBC provides a standard way for Java applications to execute SQL queries, update records, and retrieve data from a database.

- JDBC facilitates the connection between our Java application and the Oracle database by providing a set of steps to load drivers, establish connections, execute queries, and process results

Steps for Establishing a Database Connection using JDBC:

- Load the JDBC Driver:
  (oracle.jdbc.driver.OracleDriver).

- Establish a Connection:
  ```
  Connection connection = DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
  ```

- Create a Statement:
  ```
  Statement statement = connection.createStatement();
  ```

- Execute SQL Queries:
  ```
  ResultSet resultSet = statement.executeQuery("SELECT * FROM  BankTrans");
  ```

- Process Results:
  ```
  while (resultSet.next()) {
      // Process each row of the result set
  }
  ```

- Close the Connection:
  ```
  connection.close();
  ```

Scope of Project

Scope:

- Banking App with JDBC aims to create a reliable and efficient system for managing financial transactions. It encompasses functionalities such as recording transactions, updating account balances, and categorizing transactions as valid or invalid.

Objectives:

- Transaction Processing: The primary goal is to process transactions efficiently. This involves updating account balances based on deposits and withdrawals.
- Data Logging: We aim to log transaction information into separate tables, such as ValidTrans and InvalidTrans, based on the validity of the transactions.

Operating Environment

- The operating environment refers to the set of software and hardware components where our Banking App with JDBC will operate.

- **Java Environment:** Program will run on any device that has a Java Virtual Machine (JVM) installed.

- **Database Environment:** Oracle Database Management System (DBMS) is part of our operating environment. The Oracle DBMS manages the storage and retrieval of transaction data.

- **Development Tools:** Eclipse IDE is used for writing, compiling, and debugging Java code. Additionally, a database administration tool, such as Oracle SQL command Line used for managing the Oracle database.

Project Configurations

- Configurations are like the keys that unlock the door to the database, ensuring our Banking App can retrieve and store transaction data.

- **Database URL (JDBC_URL):** This is the address that specifies where the Oracle database is located. It includes details like the database name,host name, port number, and service name . For example: "jdbc:oracle:thin:@localhost:1521:XE".

- **Username and Password:**To access the Oracle database, a valid username and password are required. These credentials allow our Java application to connect securely and perform operations on the database.

Dependencies

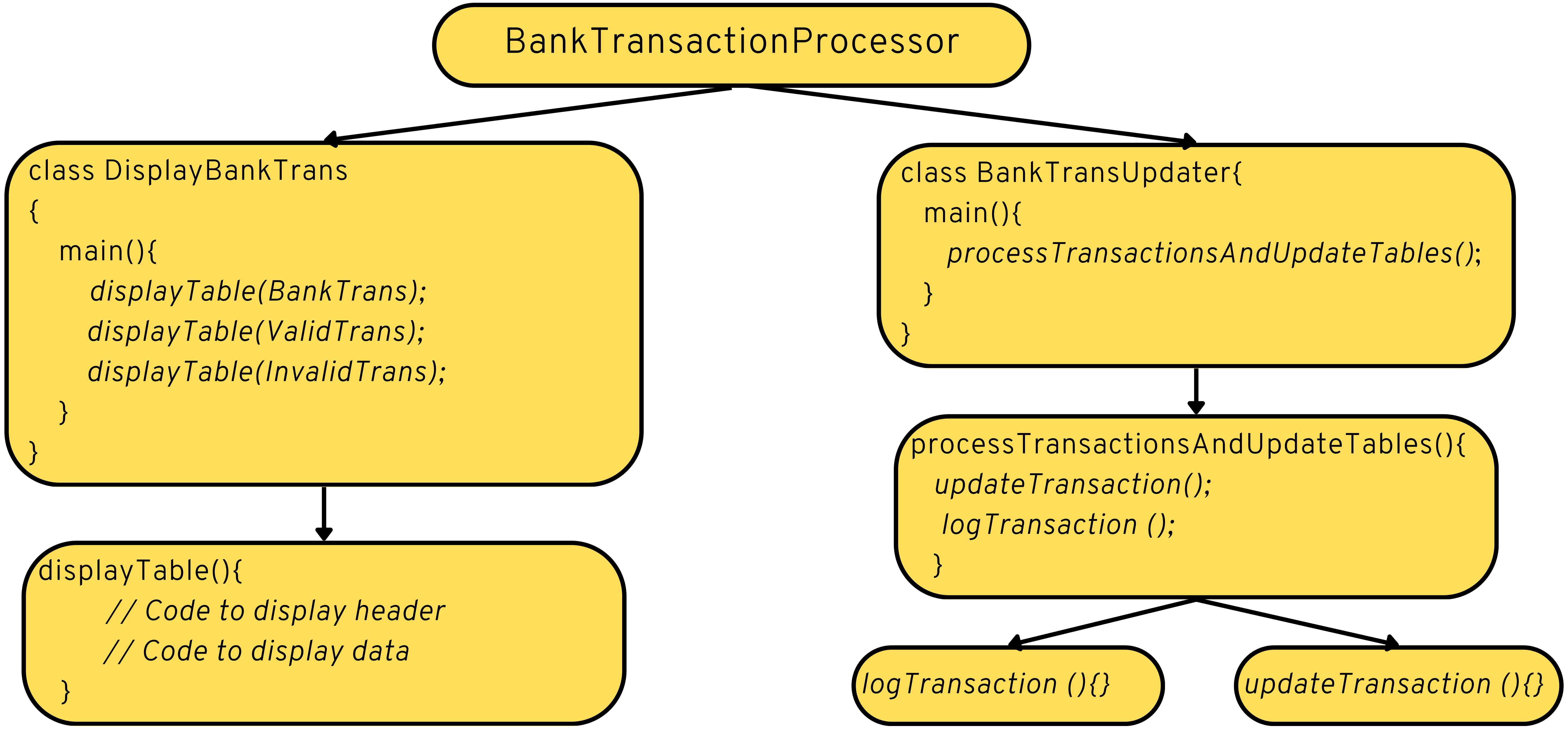
- **Ojdbc14.jar:** This is a Java Archive (JAR) file that contains the Oracle JDBC driver classes. It's like a special driver that helps our Java application talk to the Oracle database.

- **OracleDriver:** The OracleDriver is part of the JDBC driver provided by Oracle. It's the driver class that knows how to communicate with the Oracle database.

- **Tnsnames.ora:** This is a configuration file that contains details about Oracle database connections, including the database's network address.

- **Ojdbc14.jar** provides the tools, **OracleDriver** knows how to use them, and **Tnsnames.ora** helps locate the database on the network. Together, they ensure the App runs smoothly and communicates effectively with the Oracle database.

Hierarchy of Project Artifacts



```
class DisplayBankTrans
{
   main(){
      displayTable(BankTrans);
      displayTable(ValidTrans);
      displayTable(InvalidTrans);
   }
}
```

```
displayTable(){
        // Code to display header
        // Code to display data
   }
```

```
class BankTransUpdater{
  main(){
    processTransactionsAndUpdateTables();
  }
}
```

```
processTransactionsAndUpdateTables(){
  updateTransaction();
   logTransaction ();
  }
```

```
logTransaction (){}
```

```
updateTransaction (){}
```

Thank You !!!