A Command-Line Based Online Shopping System with Oracle DBMS, Group 25

Wang Ruijie, Zeng Tianyi, Zhu Jin Shun and Liu Yuyang

Deparment of Computing, The Hong Kong Polytechnic University

Introduction

The Command-Line Based Online Shopping System (OSS) is aimed at achieving, and has realized the following expectations:

- The connection and operation on Oracle database using OJDBC;
- The registration, login, and management of the user accounts for administrators and customers;
- The definement, management, search, and purchase of the products;
- The implementation of other useful additional functions;
- The friendly interaction with users through a command-line user interface.

Introduction

The Command-Line Based Online Shopping System (OSS) is aimed at achieving, and has realized the following expectations:

- The connection and operation on Oracle database using OJDBC;
- The registration, login, and management of the user accounts for administrators and customers;
- The definement, management, search, and purchase of the products;
- The implementation of other useful additional functions;
- The friendly interaction with users through a command-line user interface.

Introduction

The Command-Line Based Online Shopping System (OSS) is aimed at achieving, and has realized the following expectations:

- The connection and operation on Oracle database using OJDBC;
- The registration, login, and management of the user accounts for administrators and customers;
- The definement, management, search, and purchase of the products;
- The implementation of other useful additional functions;
- The friendly interaction with users through a command-line user interface.

Introduction

The Command-Line Based Online Shopping System (OSS) is aimed at achieving, and has realized the following expectations:

- The connection and operation on Oracle database using OJDBC;
- The registration, login, and management of the user accounts for administrators and customers;
- The definement, management, search, and purchase of the products;
- The implementation of other useful additional functions;
- The friendly interaction with users through a command-line user interface.

Introduction

The Command-Line Based Online Shopping System (OSS) is aimed at achieving, and has realized the following expectations:

- The connection and operation on Oracle database using OJDBC;
- The registration, login, and management of the user accounts for administrators and customers;
- The definement, management, search, and purchase of the products;
- The implementation of other useful additional functions;
- The friendly interaction with users through a command-line user interface.

System Architecture

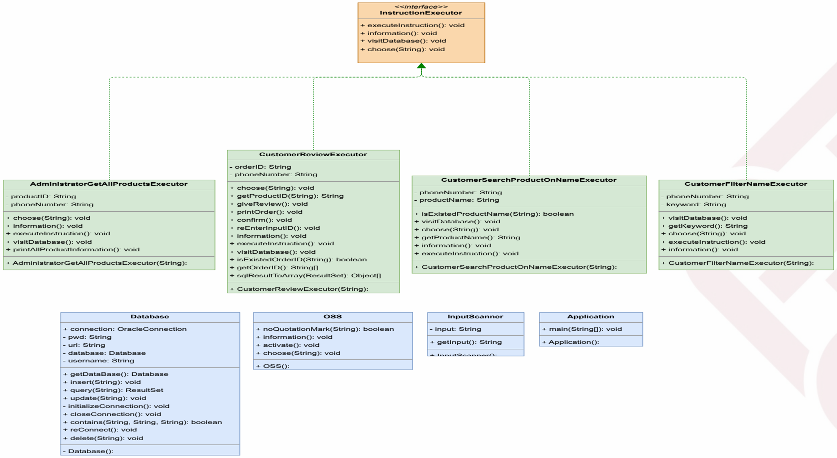

Figure 1: An example of the system architecture

Connection to Oracle DBMS and the Class of Database

The first problem to solve is to get connected to the Oracle DBMS. Hence, We define a *Database* class to collect all methods related to the interaction with the database. It contains:

- Initialization of the connection to the database.
- A getter method to return the database itself, which provides the receptor of all other database-related operations.
- Wrapper methods for the execution of queries and updates, so that only the strings of SQL statements are needed outside this class.

Connection to Oracle DBMS and the Class of Database

The first problem to solve is to get connected to the Oracle DBMS. Hence, We define a *Database* class to collect all methods related to the interaction with the database. It contains:

- Initialization of the connection to the database.
- A getter method to return the database itself, which provides the receptor of all other database-related operations.
- Wrapper methods for the execution of queries and updates, so that only the strings of SQL statements are needed outside this class.

Connection to Oracle DBMS and the Class of Database

The first problem to solve is to get connected to the Oracle DBMS. Hence, We define a *Database* class to collect all methods related to the interaction with the database. It contains:

- Initialization of the connection to the database.
- A getter method to return the database itself, which provides the receptor of all other database-related operations.
- Wrapper methods for the execution of queries and updates, so that only the strings of SQL statements are needed outside this class.

System constructs

In our design, the operation of OSS and the implementation of the functions rely on the necessary system constructs.

- An *InstructionExecutor* interface to specify the behaviors that should be present in the implementation of each function:
  - *information(): to include all prompt messages to print;*
  - *choose(): to specify all allowed user inputs and determine the solution to invalid inputs;*
  - *visitDatabase(): to determine the SQL statements and their execution methods;*
  - *executeInstruction(): to provide the logic of calling the methods above.*
- An input scanner to acquire user inputs.
- An initial interface.
- User panels for function selection.

System constructs

In our design, the operation of OSS and the implementation of the functions rely on the necessary system constructs.

- An *InstructionExecutor* interface to specify the behaviors that should be present in the implementation of each function:
  - *information(): to include all prompt messages to print;*
  - *choose(): to specify all allowed user inputs and determine the solution to invalid inputs;*
  - *visitDatabase(): to determine the SQL statements and their execution methods;*
  - *executeInstruction(): to provide the logic of calling the methods above.*
- An input scanner to acquire user inputs.
- An initial interface.
- User panels for function selection.

System constructs

In our design, the operation of OSS and the implementation of the functions rely on the necessary system constructs.

- An *InstructionExecutor* interface to specify the behaviors that should be present in the implementation of each function:
    - *information(): to include all prompt messages to print;*
    - *choose(): to specify all allowed user inputs and determine the solution to invalid inputs;*
    - *visitDatabase(): to determine the SQL statements and their execution methods;*
    - *executeInstruction(): to provide the logic of calling the methods above.*
- An input scanner to acquire user inputs.
- An initial interface.
- User panels for function selection.

System constructs

In our design, the operation of OSS and the implementation of the functions rely on the necessary system constructs.

- An *InstructionExecutor* interface to specify the behaviors that should be present in the implementation of each function:
  - *information(): to include all prompt messages to print;*
  - *choose(): to specify all allowed user inputs and determine the solution to invalid inputs;*
  - *visitDatabase(): to determine the SQL statements and their execution methods;*
  - *executeInstruction(): to provide the logic of calling the methods above.*
- An input scanner to acquire user inputs.
- An initial interface.
- User panels for function selection.

System constructs

In our design, the operation of OSS and the implementation of the functions rely on the necessary system constructs.

- An *InstructionExecutor* interface to specify the behaviors that should be present in the implementation of each function:
  - *information(): to include all prompt messages to print;*
  - *choose(): to specify all allowed user inputs and determine the solution to invalid inputs;*
  - *visitDatabase(): to determine the SQL statements and their execution methods;*
  - *executeInstruction(): to provide the logic of calling the methods above.*
- An input scanner to acquire user inputs.
- An initial interface.
- User panels for function selection.

System constructs

In our design, the operation of OSS and the implementation of the functions rely on the necessary system constructs.

- An *InstructionExecutor* interface to specify the behaviors that should be present in the implementation of each function:
    - *information(): to include all prompt messages to print;*
    - *choose(): to specify all allowed user inputs and determine the solution to invalid inputs;*
    - *visitDatabase(): to determine the SQL statements and their execution methods;*
    - *executeInstruction(): to provide the logic of calling the methods above.*
- An input scanner to acquire user inputs.
- An initial interface.
- User panels for function selection.

System constructs

In our design, the operation of OSS and the implementation of the functions rely on the necessary system constructs.

- An *InstructionExecutor* interface to specify the behaviors that should be present in the implementation of each function:
  - *information(): to include all prompt messages to print;*
  - *choose(): to specify all allowed user inputs and determine the solution to invalid inputs;*
  - *visitDatabase(): to determine the SQL statements and their execution methods;*
  - *executeInstruction(): to provide the logic of calling the methods above.*
- An input scanner to acquire user inputs.
- An initial interface.
- User panels for function selection.

System constructs

In our design, the operation of OSS and the implementation of the functions rely on the necessary system constructs.

- An *InstructionExecutor* interface to specify the behaviors that should be present in the implementation of each function:
    - *information(): to include all prompt messages to print;*
    - *choose(): to specify all allowed user inputs and determine the solution to invalid inputs;*
    - *visitDatabase(): to determine the SQL statements and their execution methods;*
    - *executeInstruction(): to provide the logic of calling the methods above.*
- An input scanner to acquire user inputs.
- An initial interface.
- User panels for function selection.

Function Classification

The executors of the instructions are defined in three packages: *system*, *administrator* and *customer*.

- Each executor realizes a function listed on the user panel.
- We divide the functions into five parts:
  - *System functions of account registration and login*
  - *Account information modification;*
  - *Searching and filtering*
  - *Purchase and management of products*
  - *Analysis report generation*

Function Classification

The executors of the instructions are defined in three packages: *system*, *administrator* and *customer*.

- Each executor realizes a function listed on the user panel.
- We divide the functions into five parts:
  - *System functions of account registration and login;*
  - *Account information modification;*
  - *Searching and filtering*
  - *Purchase and management of products*
  - *Analysis report generation*

Function Classification

The executors of the instructions are defined in three packages: *system*, *administrator* and *customer*.

- Each executor realizes a function listed on the user panel.
- We divide the functions into five parts:
  - *System functions of account registration and login;*
  - *Account information modification;*
  - *Searching and filtering*
  - *Purchase and management of products*
  - *Analysis report generation*

Function Classification

The executors of the instructions are defined in three packages: *system*, *administrator* and *customer*.

- Each executor realizes a function listed on the user panel.
- We divide the functions into five parts:
  - *System functions of account registration and login;*
  - *Account information modification;*
  - *Searching and filtering*
  - *Purchase and management of products*
  - *Analysis report generation*

Function Classification

The executors of the instructions are defined in three packages: *system*, *administrator* and *customer*.

- Each executor realizes a function listed on the user panel.
- We divide the functions into five parts:
  - *System functions of account registration and login;*
  - *Account information modification;*
  - *Searching and filtering*
  - *Purchase and management of products*
  - *Analysis report generation*

Function Classification

The executors of the instructions are defined in three packages: *system*, *administrator* and *customer*.

- Each executor realizes a function listed on the user panel.
- We divide the functions into five parts:
  - *System functions of account registration and login;*
  - *Account information modification;*
  - *Searching and filtering*
  - *Purchase and management of products*
  - *Analysis report generation*

Function Classification

The executors of the instructions are defined in three packages: *system*, *administrator* and *customer*.

- Each executor realizes a function listed on the user panel.
- We divide the functions into five parts:
  - *System functions of account registration and login;*
  - *Account information modification;*
  - *Searching and filtering*
  - *Purchase and management of products*
  - *Analysis report generation*

Database Schema

We regard the functions as the query or updating SQL statements and the following effects of the functions as the extraction of the returned result sets. Therefore, we adopt the following relational schema for the implementation of the functions:

- ADMIN and CUSTOMER (primary key: PHONE_NUMBER);
- PRODUCT (primary key: PRODUCT_ID);
- SHOPPING_CART (no primary key constraint);
- ORDER_ (primary key: ORDER_ID);
- REVIEW (primary key: ORDER_ID).

Database Schema

We regard the functions as the query or updating SQL statements and the following effects of the functions as the extraction of the returned result sets. Therefore, we adopt the following relational schema for the implementation of the functions:

- ADMIN and CUSTOMER (primary key: PHONE_NUMBER);
- PRODUCT (primary key: PRODUCT_ID);
- SHOPPING_CART (no primary key constraint);
- ORDER_ (primary key: ORDER_ID);
- REVIEW (primary key: ORDER_ID).

Database Schema

We regard the functions as the query or updating SQL statements and the following effects of the functions as the extraction of the returned result sets. Therefore, we adopt the following relational schema for the implementation of the functions:

- ADMIN and CUSTOMER (primary key: PHONE_NUMBER);
- PRODUCT (primary key: PRODUCT_ID);
- SHOPPING_CART (no primary key constraint);
- ORDER_ (primary key: ORDER_ID);
- REVIEW (primary key: ORDER_ID).

Database Schema

We regard the functions as the query or updating SQL statements and the following effects of the functions as the extraction of the returned result sets. Therefore, we adopt the following relational schema for the implementation of the functions:

- ADMIN and CUSTOMER (primary key: PHONE_NUMBER);
- PRODUCT (primary key: PRODUCT_ID);
- SHOPPING_CART (no primary key constraint);
- ORDER_ (primary key: ORDER_ID);
- REVIEW (primary key: ORDER_ID).

Database Schema

We regard the functions as the query or updating SQL statements and the following effects of the functions as the extraction of the returned result sets. Therefore, we adopt the following relational schema for the implementation of the functions:

- ADMIN and CUSTOMER (primary key: PHONE_NUMBER);
- PRODUCT (primary key: PRODUCT_ID);
- SHOPPING_CART (no primary key constraint);
- ORDER_ (primary key: ORDER_ID);
- REVIEW (primary key: ORDER_ID).

Demonstration

Let's start the demonstration!