EECS 3421

Project 2

Student name: Jiachen Hou

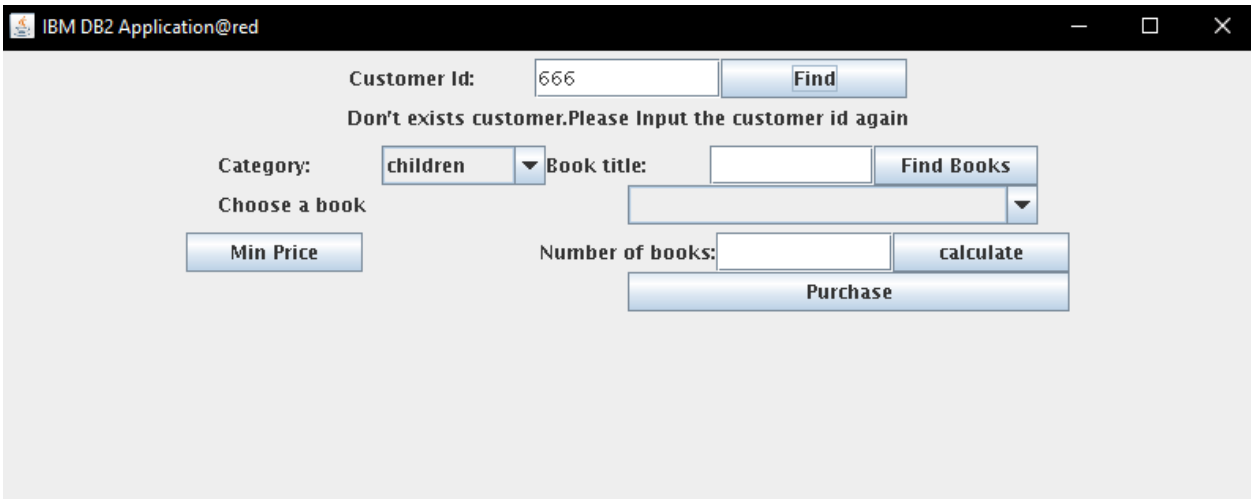
Student number: 213152590

Recorded session:

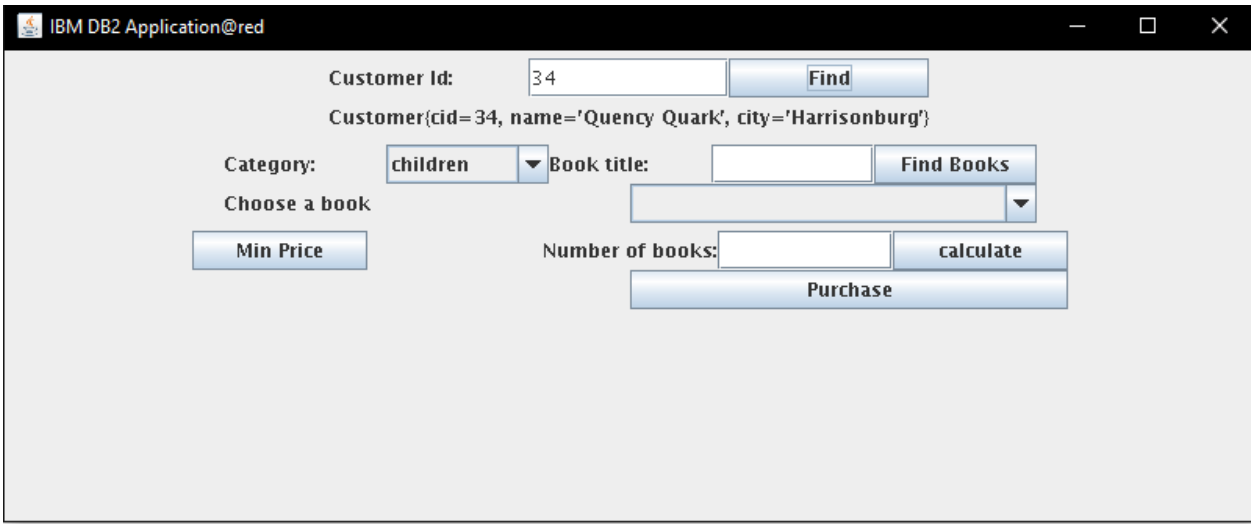
*Bold font contains instruction for GUI using.

If the customer does not exist, the program displays an error message and requests the customer id again.

Input cid and click 'Find' button to search for a customer ID.



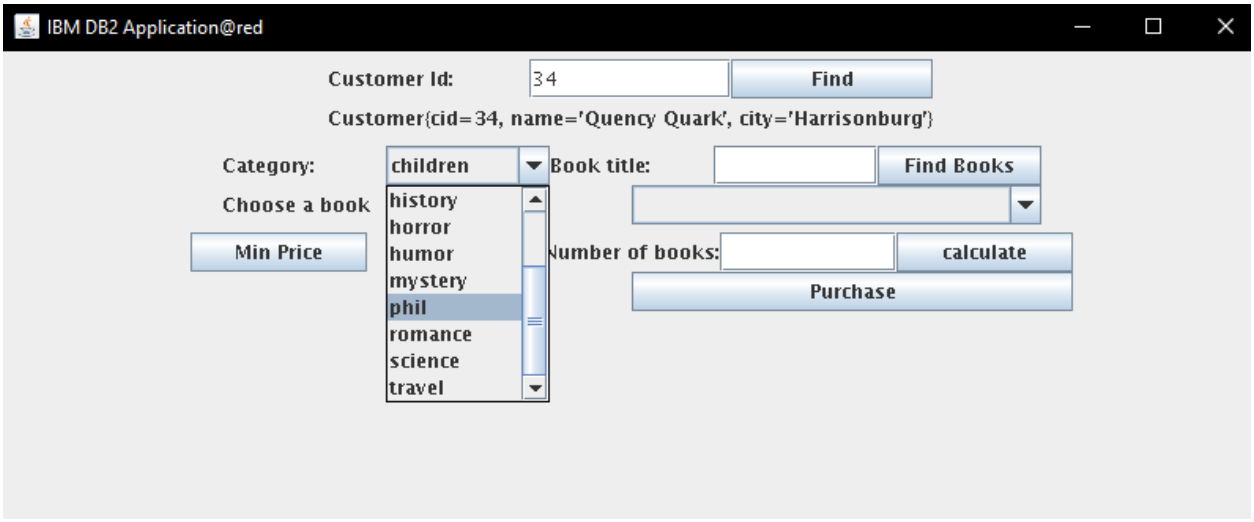
If the customer exists, the query returns and displays the customer information (the customer id, name, and city).

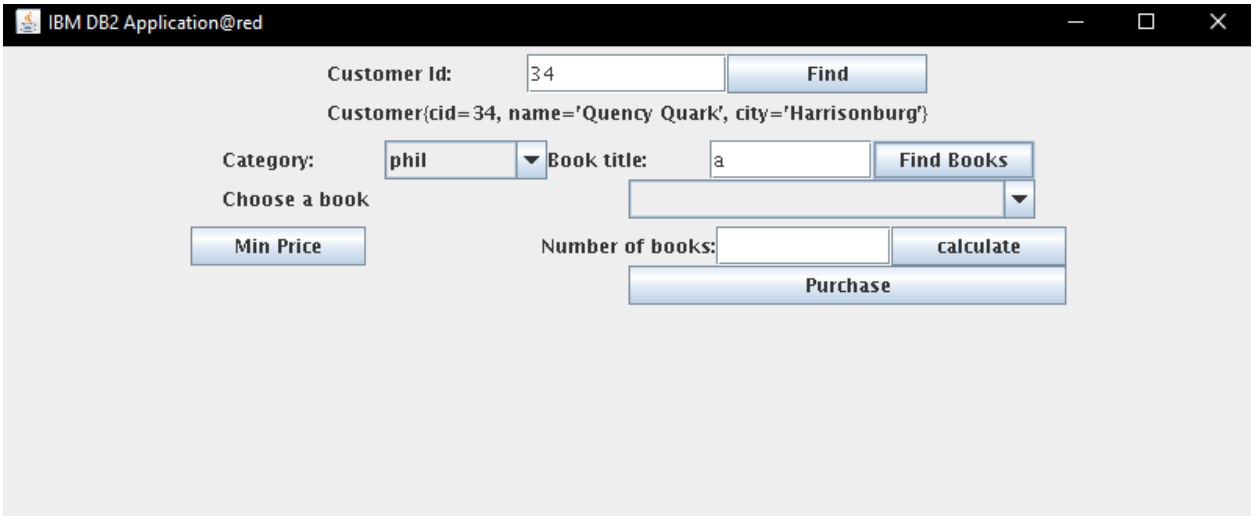


If the customer exists:

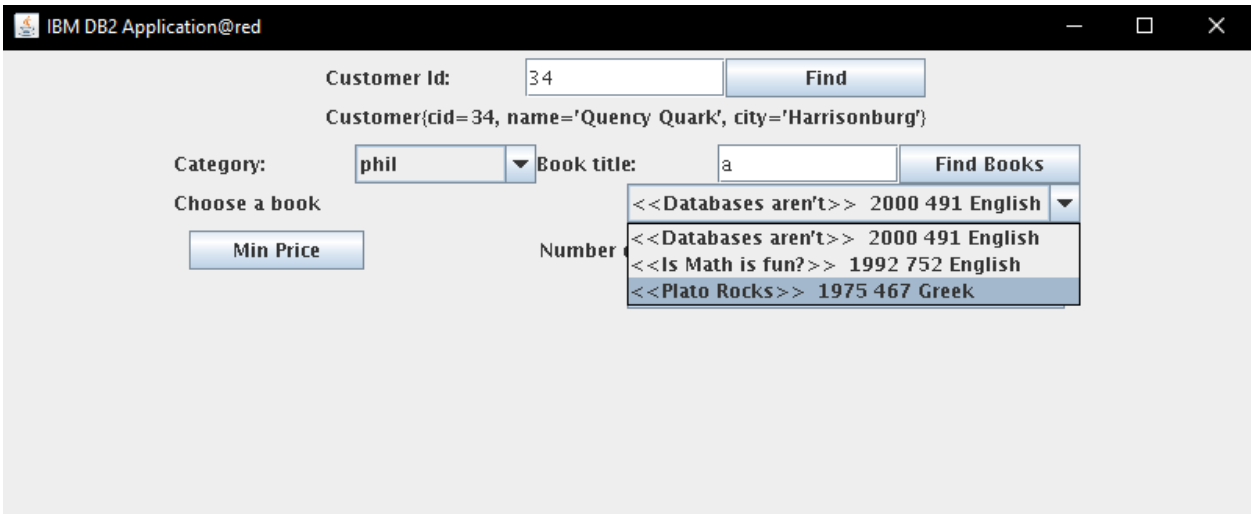
Customer can choose a category from the list.

After choosing a category by the customer, customer can enter the title of the book. If the given title with the selected category exists, return the book information (title, year, language, weight). If the book title cannot be found in database, customer may re-enter book title. If the book title is valid, customer may choose the book since it could have more than one same book.





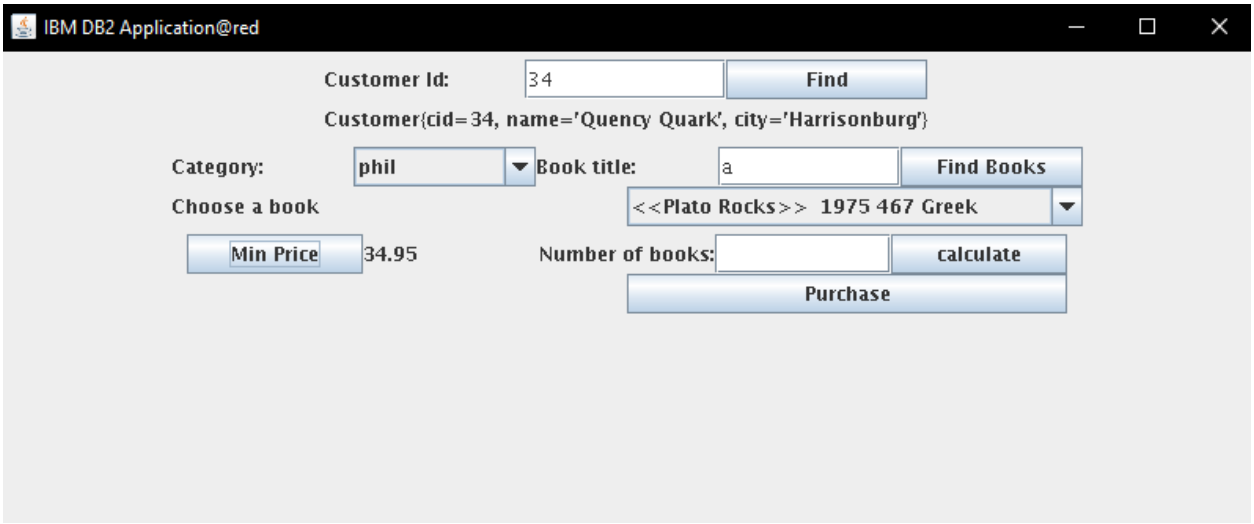
Input partial or complete book title and click 'Find Books' button to search a book in yrb.

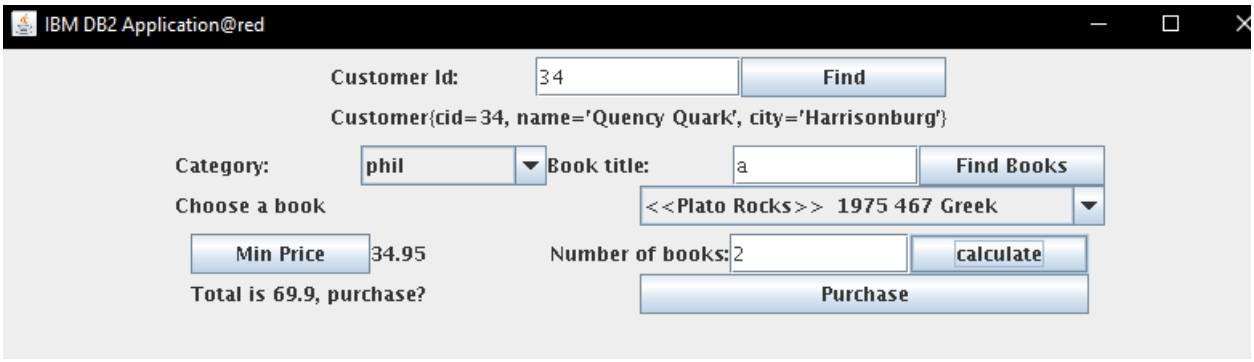


The user selects a book to buy, the minimum price for that book will be retrieved from the database (yrb_offer). **Click the "Min Price" button to show the minimum price for the given book.**

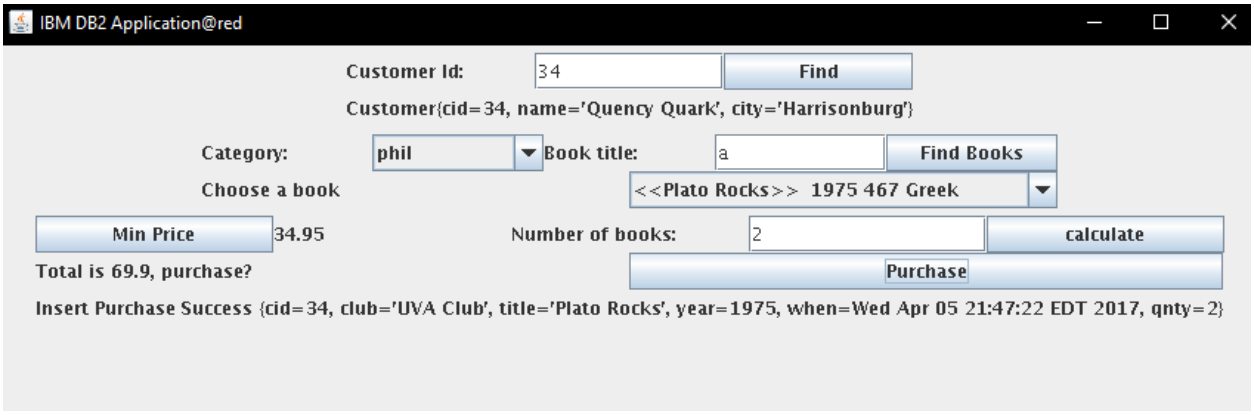
The minimum price will be displayed to the user.

Ask the user to enter the number of books (the quantity) to buy. **After entered the quantity, click 'Calculate' to obtain a total price.** The total price is calculated and displayed to the user.





If the user approves (ask if the user wants to purchase the book/books?), the purchase information will be stored in the purchase table with current date and time. **Click 'Purchase' button to make a purchase.**



Once purchase complete, connect to db2 and enter the query "select * from yrb_purchase" to check the recent purchase history.(insert purchase to database)

```
   34 AAA             Is Math is fun?          1992 2017-04-05-18.09.17.107000     11
   34 UVA Club        Plato Rocks              1975 2017-04-05-18.32.37.262000     11
   34 UVA Club        Plato Rocks              1975 2017-04-05-18.36.03.913000     11
   34 UVA Club        Plato Rocks              1975 2017-04-05-18.38.30.874000      1
   34 UVA Club        Plato Rocks              1975 2017-04-05-18.53.06.759000      5
   33 Oprah           Where are my Socks?      1994 2017-04-05-18.53.20.101000      2
   34 UVA Club        Brats like Us            1995 2017-04-05-19.00.07.202000     12
   34 UVA Club        Brats like Us            1995 2017-04-05-19.00.08.400000     12
   34 UVA Club        Plato Rocks              1975 2017-04-05-21.47.22.237000      2

  354 record(s) selected.

db2 =>
```

The latest purchase for 2 'Plato Rocks' shows in the database.

Source code:

JdbcQueryUtil.java (Query)

```
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Date;

public class JdbcQueryUtil {

    String jdbcClassName="com.ibm.db2.jcc.DB2Driver";
    String url="jdbc:db2:c3421m";

    Connection connection = null;
    String  queryText = "";     // The SQL text.
    PreparedStatement querySt   = null;   // The query handle.
    ResultSet answers   = null;   // A cursor.


    public static JdbcQueryUtil instance = new JdbcQueryUtil();

    private JdbcQueryUtil() {
        try {
            try {
                Class.forName(jdbcClassName).newInstance();
            } catch (InstantiationException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            }
            System.out.println("Start Connecting");
            connection = DriverManager.getConnection(url);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally{
            if(connection!=null){
                System.out.println("Connected successfully.");
            }
        }
    }

    public static JdbcQueryUtil getInstance() {
        return instance;
    }
/*
 * Find customer with given id.
 */
    public Customer findCustomer(int cid) {
        Customer customer = null;
        queryText = "SELECT * " + "FROM yrb_customer " + "WHERE cid = ?";
```

```
        try {
            querySt = connection.prepareStatement(queryText);
            querySt.setInt(1, cid);
            answers = querySt.executeQuery();
            if (answers.next()) {
                customer = new Customer();
                customer.setCid(cid);
                customer.setName(answers.getString("name"));
                customer.setCity(answers.getString("city"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return customer;
    }
/*
 * Find member that belongs to which clubs.
 */
    public Member findMember(int cid) {
        Member member = null;
        queryText = "SELECT * " + "FROM yrb_member " + "WHERE cid = ?";
        try {
            querySt = connection.prepareStatement(queryText);
            querySt.setInt(1, cid);
            answers = querySt.executeQuery();
            if (answers.next()) {
                member = new Member();
                member.setCid(cid);
                member.setClub(answers.getString("club"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return member;
    }
/*
 * Fetch all categories from database.
 */
    public List<Category> fetchCategories() {
        List<Category> categories = new ArrayList<>();
        queryText = "SELECT * " + "FROM yrb_category";
        try {
            querySt = connection.prepareStatement(queryText);
            answers = querySt.executeQuery();
            while (answers.next()) {
                Category category = new Category();
                category.setCat(answers.getString("cat"));
                categories.add(category);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return categories;
    }
/*
```

```
 *Find book from yrb_book with given partial title in given category.
 */
    public List<Book> findBook(String title, String cat) {
        List<Book> books = new ArrayList<>();
        queryText = "SELECT * " + "FROM yrb_book " + "WHERE title LIKE ? AND cat = ?";
        try {
            querySt = connection.prepareStatement(queryText);
            querySt.setString(1, "%" + title + "%");
            querySt.setString(2, cat);
            answers = querySt.executeQuery();
            while (answers.next()) {
                Book book = new Book();
                book.setTitle(answers.getString("title"));
                book.setCat(cat);
                book.setLanguage(answers.getString("language"));
                book.setYear(answers.getInt("year"));
                book.setWeight(answers.getInt("weight"));
                books.add(book);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return books;
    }
/*
 * Shows the minimum price that can be offered by customer's club. Customers are not allowed
to buy books in the club without member.
 */
    public Offer minPrice(int cid, String title) {

       Offer offer = null;
        queryText = "SELECT * FROM yrb_offer WHERE club in (SELECT club FROM yrb_member
where cid = ?) and title = ? ORDER BY price limit 1";
        try {
            querySt = connection.prepareStatement(queryText);
            querySt.setInt(1, cid);
            querySt.setString(2, title);
            answers = querySt.executeQuery();
            if (answers.next()) {
                offer = new Offer();
                offer.setTitle(title);
                offer.setYear(answers.getInt("year"));
                offer.setClub(answers.getString("club"));
                offer.setPrice(answers.getFloat("price"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return offer;
    }
/*
 * Insert the purchase into yrb_purchase.
 */
    public boolean insertPurchase(Purchase purchase) {
        boolean result = false;
```

```
        queryText = "insert into yrb_purchase (cid,club,title,year,when,qnty) values
(?, ?, ?, ?, ?, ?)";
        try {
            querySt = connection.prepareStatement(queryText);
            querySt.setInt(1, purchase.getCid());
            querySt.setString(2, purchase.getClub());
            querySt.setString(3, purchase.getTitle());
            querySt.setInt(4, purchase.getYear());
            querySt.setTimestamp(5, new Timestamp(purchase.getWhen().getTime()));
            querySt.setInt(6, purchase.getQnty());
            result = querySt.execute();
            System.out.println(result);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return result;
    }
}
```

Book.java

```
public class Book {

    private String title;

    private int year;

    private String language;

    private String cat;

    private int weight;

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public int getYear() {

        return year;

    }

    public void setYear(int year) {

        this.year = year;

    }

    public String getCat() {

        return cat;
```

```
    }

    public void setCat(String cat) {
        this.cat = cat;
    }

    public String getLanguage() {
        return language;
    }

    public void setLanguage(String language) {
        this.language = language;
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    @Override
    public String toString() {
        return  "<<" + title + ">> " + " " + year + " " + weight + " " + language;
    }
}
```

Category.java

```
public class Category {

    private String cat;

    public String getCat() {
        return cat;
    }

    public void setCat(String cat) {
        this.cat = cat;
    }

    @Override
    public String toString() {
        return cat;
    }
}
```

Club.java

```
public class Club {

    private String club;
    private String desc;

    public Club(String club, String desc) {
        this.club = club;
        this.desc = desc;
    }

    public String getClub() {
        return club;
    }

    public void setClub(String club) {
        this.club = club;
    }

    public String getDesc() {
        return desc;
    }

    public void setDesc(String desc) {
        this.desc = desc;
    }
}
```

Customer.java

```java
public class Customer {

    private int cid;
    private String name;
    private String city;

    public int getCid() {
        return cid;
    }

    public void setCid(int cid) {
        this.cid = cid;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    @Override
    public String toString() {
        return "Customer{" +
                "cid=" + cid +
                ", name='" + name + '\'' +
                ", city='" + city + '\'' +
                '}';
    }
}
```

Member.java

```
public class Member {

    private String club;
    private int cid;

    public String getClub() {
        return club;
    }

    public void setClub(String club) {
        this.club = club;
    }

    public int getCid() {
        return cid;
    }

    public void setCid(int cid) {
        this.cid = cid;
    }
}
```

Offer.java

```
public class Offer {

    private String club;
    private String title;
    private int year;
    private float price;

    public String getClub() {
        return club;
    }

    public void setClub(String club) {
        this.club = club;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public int getYear() {
        return year;
    }
```

```
    public void setYear(int year) {
        this.year = year;
    }

    public float getPrice() {
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }
}
```

Purchase.java

```
import java.util.Date;

public class Purchase {

    private int cid;
    private String club;
    private String title;
    private int year;
    private Date when;
    private int qnty;

    public int getCid() {
        return cid;
    }

    public void setCid(int cid) {
        this.cid = cid;
    }

    public String getClub() {
        return club;
    }

    public void setClub(String club) {
        this.club = club;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public int getYear() {
        return year;
    }
```

```
    public void setYear(int year) {
        this.year = year;
    }

    public Date getWhen() {
        return when;
    }

    public void setWhen(Date when) {
        this.when = when;
    }

    public int getQnty() {
        return qnty;
    }

    public void setQnty(int qnty) {
        this.qnty = qnty;
    }

    @Override
    public String toString() {
        return "Insert Purchase Success {" +
                "cid=" + cid +
                ", club='" + club + '\'' +
                ", title='" + title + '\'' +
                ", year=" + year +
                ", when=" + when +
                ", qnty=" + qnty +
                '}';
    }
}
```

Test.java (Main class)

```
/*
 * Main class for test the program.
 *
 */
public class Test {

    public static void main(String[] args) {
        JdbcQueryUtil mJdbcQueryUtil = JdbcQueryUtil.getInstance();
        mJdbcQueryUtil.findCustomer(1);
        Window window = new Window();
        window.initUI();
        window.initEvent();
    }
}
```

Window.java (GUI)

```
import javax.swing.*;
import java.awt.*;
import java.util.Date;
import java.util.List;

public class Window extends JFrame {

    public static final int WINDOW_WIDTH = 800;
    public static final int WINDOW_HEIGHT = 300;

    JPanel customerPanel;
    JLabel customerJTFLabel;
    JTextField customerJTF;
    JButton customerBtn;
    JLabel customerLabel;

    JPanel categoryPanel;
    JLabel categoryComboBoxLabel;
    JComboBox<Category> categoryComboBox;
    JLabel bookJTFLabel;
    JTextField bookJTF;
    JButton bookBtn;
    JLabel bookChooseLabel = new JLabel("Choose a book");
    JComboBox<Book> bookComboBox;

    JPanel purchasePanel;
    JButton priceBtn = new JButton("Min Price");
    JLabel priceLabel = new JLabel();
    JLabel quantityLabel = new JLabel("Number of books:");
    JTextField quantityJTF = new JTextField();
    JButton calculateBtn = new JButton("calculate");
//    JLabel countLabel = new JLabel("You wants to purchase the book/books?");
    JLabel countLabel = new JLabel();
```

```
JButton purchaseBtn = new JButton("Purchase");
JLabel purchaseResultLabel = new JLabel();


// DB2
JdbcQueryUtil mJdbcQueryUtil;

// varibles
Customer customer;
Category category;
Book book;
Offer offer;
int quantity;
float count;

public Window() {
    mJdbcQueryUtil = JdbcQueryUtil.getInstance();
}

public void initUI() {

    this.setTitle("IBM DB2 Application");
    this.setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    this.setDefaultCloseOperation(3);
    this.setLocationRelativeTo(null);
    this.setLayout(new FlowLayout());

    customerPanel = new JPanel();
    customerPanel.setLayout(new GridLayout(2, 1));
    JPanel containerPanel = new JPanel();
    containerPanel.setLayout(new GridLayout(1, 3));
    customerJTFLabel = new JLabel("Customer Id:");
    customerJTF = new JTextField();
    customerJTF.setSize(100, 20);
    customerBtn = new JButton("Find");
    customerLabel = new JLabel();
    containerPanel.add(customerJTFLabel);
    containerPanel.add(customerJTF);
    containerPanel.add(customerBtn);
    customerPanel.add(containerPanel);
    customerPanel.add(customerLabel);

    categoryPanel = new JPanel();
    categoryPanel.setLayout(new GridLayout(2,1));
    JPanel containerPanel2 = new JPanel();
    containerPanel2.setLayout(new GridLayout(1, 5));
    categoryComboBoxLabel = new JLabel("Category:");
    categoryComboBox = new JComboBox<>();
    List<Category> categories = mJdbcQueryUtil.fetchCategories();
    for (Category category : categories) {
        categoryComboBox.addItem(category);
    }
    bookJTFLabel = new JLabel("Book title:");
    bookJTF = new JTextField();
    bookBtn = new JButton("Find Books");
```

```
        containerPanel2.add(categoryComboBoxLabel);
        containerPanel2.add(categoryComboBox);
        containerPanel2.add(bookJTFLabel);
        containerPanel2.add(bookJTF);
        containerPanel2.add(bookBtn);
        categoryPanel.add(BorderLayout.NORTH, containerPanel2);
        JPanel containerPanel3 = new JPanel();
        containerPanel3.setLayout(new GridLayout(1, 2));
        containerPanel3.add(bookChooseLabel);
        bookComboBox = new JComboBox<>();
        containerPanel3.add(bookComboBox);
        categoryPanel.add(containerPanel3);

        purchasePanel = new JPanel();
        purchasePanel.setLayout(new GridLayout(3, 1));
        JPanel containerPanel4 = new JPanel();
        JPanel containerPanel5 = new JPanel();
        // JPanel containerPanel6 = new JPanel();
        containerPanel4.setLayout(new GridLayout(1, 5));
        containerPanel4.add(priceBtn);
        containerPanel4.add(priceLabel);
        containerPanel4.add(quantityLabel);
        containerPanel4.add(quantityJTF);
        containerPanel4.add(calculateBtn);
        containerPanel5.setLayout(new GridLayout(1, 2));
        containerPanel5.add(countLabel);
        containerPanel5.add(purchaseBtn);
        // JLabel purchaseResultLabel = new JLabel();
        purchasePanel.add(containerPanel4);
        purchasePanel.add(containerPanel5);
        purchasePanel.add(purchaseResultLabel);

        this.add(customerPanel);
        this.add(categoryPanel);
        this.add(purchasePanel);
        this.setVisible(true);
    }

    public void initEvent() {
/*
 * Add function button for find customers. If customer does not exist, show error message
and ask user to re-enter.
 */
        customerBtn.addActionListener(event -> {
            String cid = customerJTF.getText().trim();
            customer = mJdbcQueryUtil.findCustomer(Integer.parseInt(cid));
            if (customer != null) {
                customerLabel.setText(customer.toString());
            } else {
                customerLabel.setText("Don't exists customer.Please Input the customer id
again");
            }
        });
/*
 * Add category selection.
```

```
 */
        categoryComboBox.addActionListener(event -> {
            category = categoryComboBox.getItemAt(categoryComboBox.getSelectedIndex());
            System.out.println(category.toString());
        });
/*
 * Add find book function button. The button will search book with given partial title
in given category.
 */
        bookBtn.addActionListener(event -> {
            bookComboBox.removeAllItems();
            String title = bookJTF.getText().trim();
            if (category != null) {
                List<Book> books = mJdbcQueryUtil.findBook(title, category.getCat());
                for (Book book : books) {
                    bookComboBox.addItem(book);
                }
            }
        });
/*
 * An output for searching result.
 */
            bookComboBox.addActionListener(event ->  {
            book = bookComboBox.getItemAt(bookComboBox.getSelectedIndex());
            System.out.println(book);
        });
/*
 * Show the minimum price for the book offer by customer's club.
 */
        priceBtn.addActionListener(event -> {
            if (book != null) {
                offer = mJdbcQueryUtil.minPrice(customer.getCid(),book.getTitle());
                if (offer != null) {
                    priceLabel.setText(offer.getPrice() + "");
                }
            }
        });
/*
 * Calculation button for calculate the total price.
 */
        calculateBtn.addActionListener(event -> {
            if (offer != null) {
                String number = quantityJTF.getText().trim();
                quantity = Integer.parseInt(number);
                count = offer.getPrice() * quantity;
                countLabel.setText("Total is " + count + ", purchase?");
            }
        });
/*
 * Once purchase succeed, a message will shows up with cid, club, quantity, book title
and current time.
 */
        purchaseBtn.addActionListener(event -> {
            Member member = mJdbcQueryUtil.findMember(customer.getCid());
            if (member != null) {
```

```
                Purchase purchase = new Purchase();
                purchase.setCid(member.getCid());
                purchase.setClub(offer.getClub());
                purchase.setQnty(quantity);
                purchase.setTitle(offer.getTitle());
                purchase.setWhen(new Date());
                purchase.setYear(offer.getYear());
                if (mJdbcQueryUtil.insertPurchase(purchase)) {
                     purchaseResultLabel.setText("Purchase false.");
                } else {
                    purchaseResultLabel.setText(purchase.toString());
                }

            }
        });


    }

}
```