Relational Databases with MySQL Week 6 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document to the repository. Additionally, push an .sql file with all your queries and your Java project code to the same repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

This week you will be working together as a **team** to create a full CRUD application.

Your console CRUD application will need to use a database to store all the application data.

As a team, decide what you want your project to do. Get instructor approval early in the week before beginning development.

You need to have at least 3 entities.

Users should be able to interact via the console (i.e. Scanner(System.in)))

Everyone must individually submit their own assignment documents and the full code for the entire project. Inside the code, use comments to make it clear which code you specifically wrote.

Although git provides collaboration functionality, you are not required to use it to collaborate back and forth with your teammates. You can use any method you decide on as a team.

Everyone will be graded on their individual contributions.

Project Name:

Goaltivity

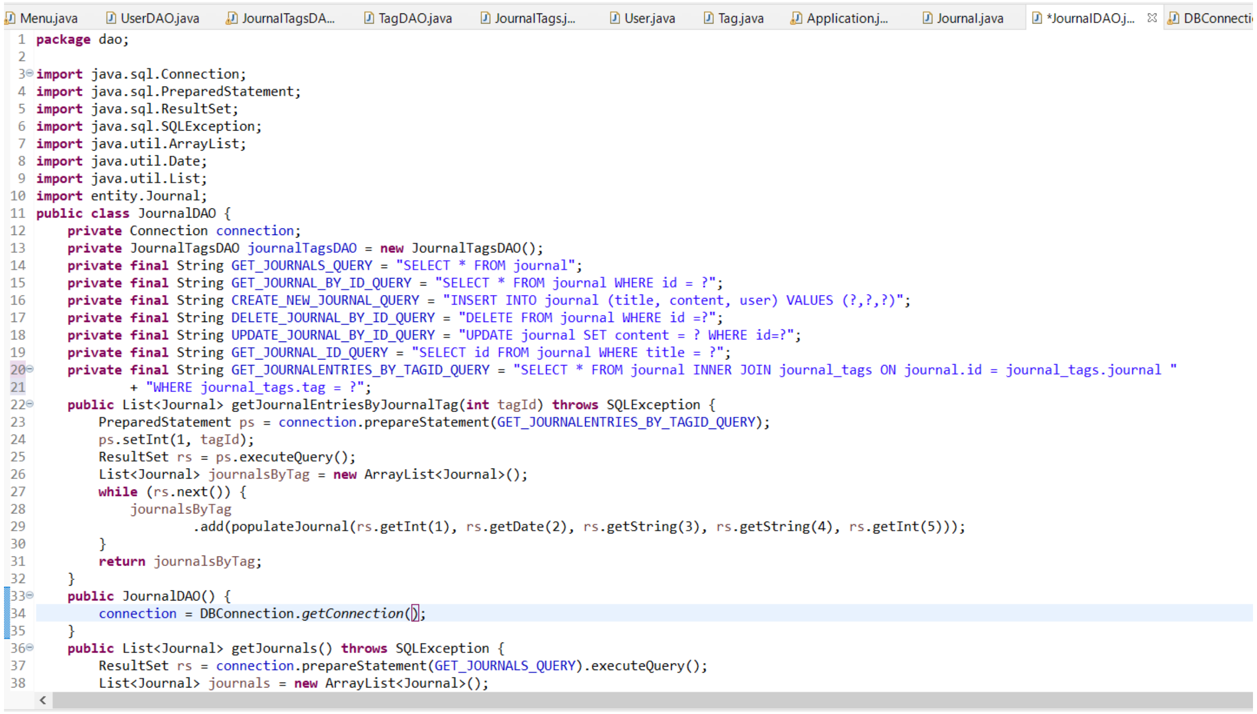
Project Team Members:

Myles

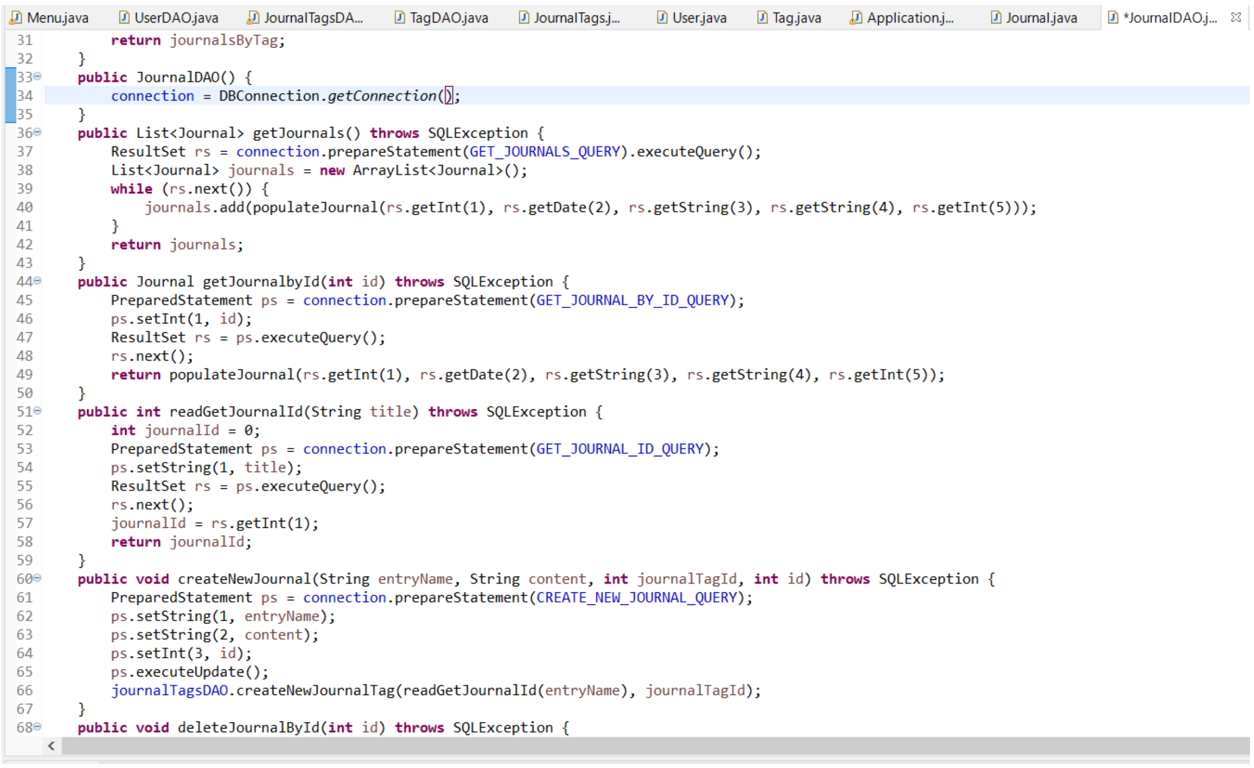
Heather

My Contribution to the Project:

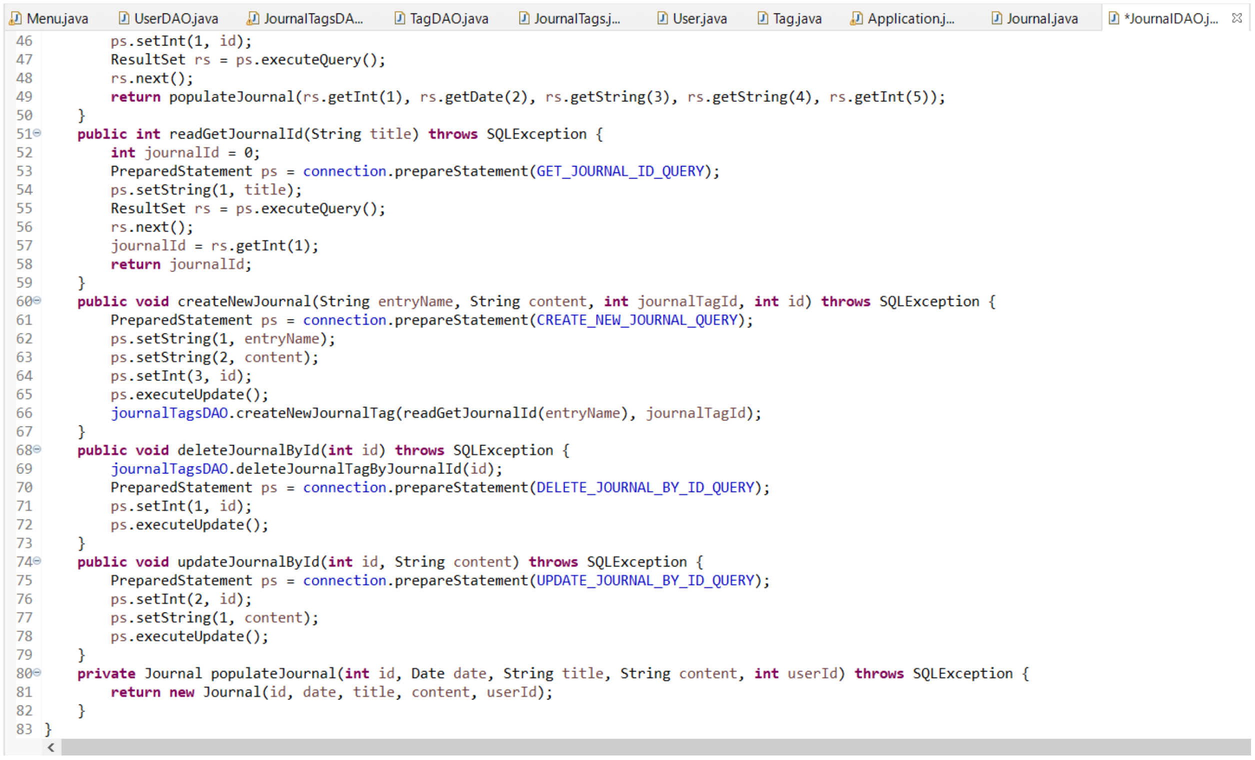
I did the journal and journal tags operations mainly but Myles and I both zoomed to help each other work through problems in the user and part of the menu. My complete contributions can be seen by the github repo https://github.com/Goaltivity/Goaltivity but this is the main group of code I worked on.



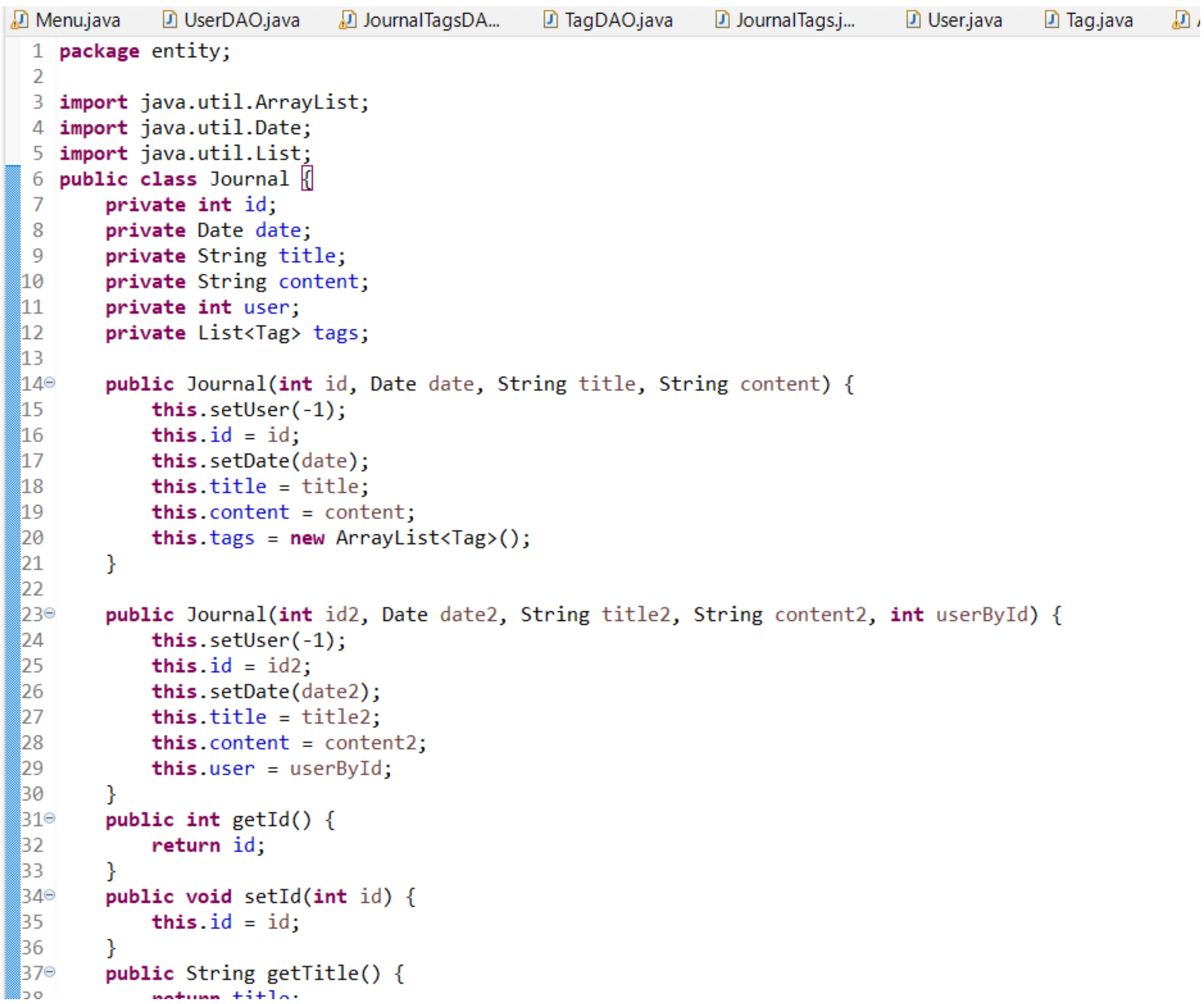
```java
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import entity.Journal;
public class JournalDAO {
    private Connection connection;
    private JournalTagsDAO journalTagsDAO = new JournalTagsDAO();
    private final String GET_JOURNALS_QUERY = "SELECT * FROM journal";
    private final String GET_JOURNAL_BY_ID_QUERY = "SELECT * FROM journal WHERE id = ?";
    private final String CREATE_NEW_JOURNAL_QUERY = "INSERT INTO journal (title, content, user) VALUES (?,?,?)";
    private final String DELETE_JOURNAL_BY_ID_QUERY = "DELETE FROM journal WHERE id =?";
    private final String UPDATE_JOURNAL_BY_ID_QUERY = "UPDATE journal SET content = ? WHERE id=?";
    private final String GET_JOURNAL_ID_QUERY = "SELECT id FROM journal WHERE title = ?";
    private final String GET_JOURNALENTRIES_BY_TAGID_QUERY = "SELECT * FROM journal INNER JOIN journal_tags ON journal.id = journal_tags.journal "
            + "WHERE journal_tags.tag = ?";
    public List<Journal> getJournalEntriesByJournalTag(int tagId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_JOURNALENTRIES_BY_TAGID_QUERY);
        ps.setInt(1, tagId);
        ResultSet rs = ps.executeQuery();
        List<Journal> journalsByTag = new ArrayList<Journal>();
        while (rs.next()) {
            journalsByTag
                    .add(populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
        }
        return journalsByTag;
    }
    public JournalDAO() {
        connection = DBConnection.getConnection();
    }
    public List<Journal> getJournals() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_JOURNALS_QUERY).executeQuery();
        List<Journal> journals = new ArrayList<Journal>();
```



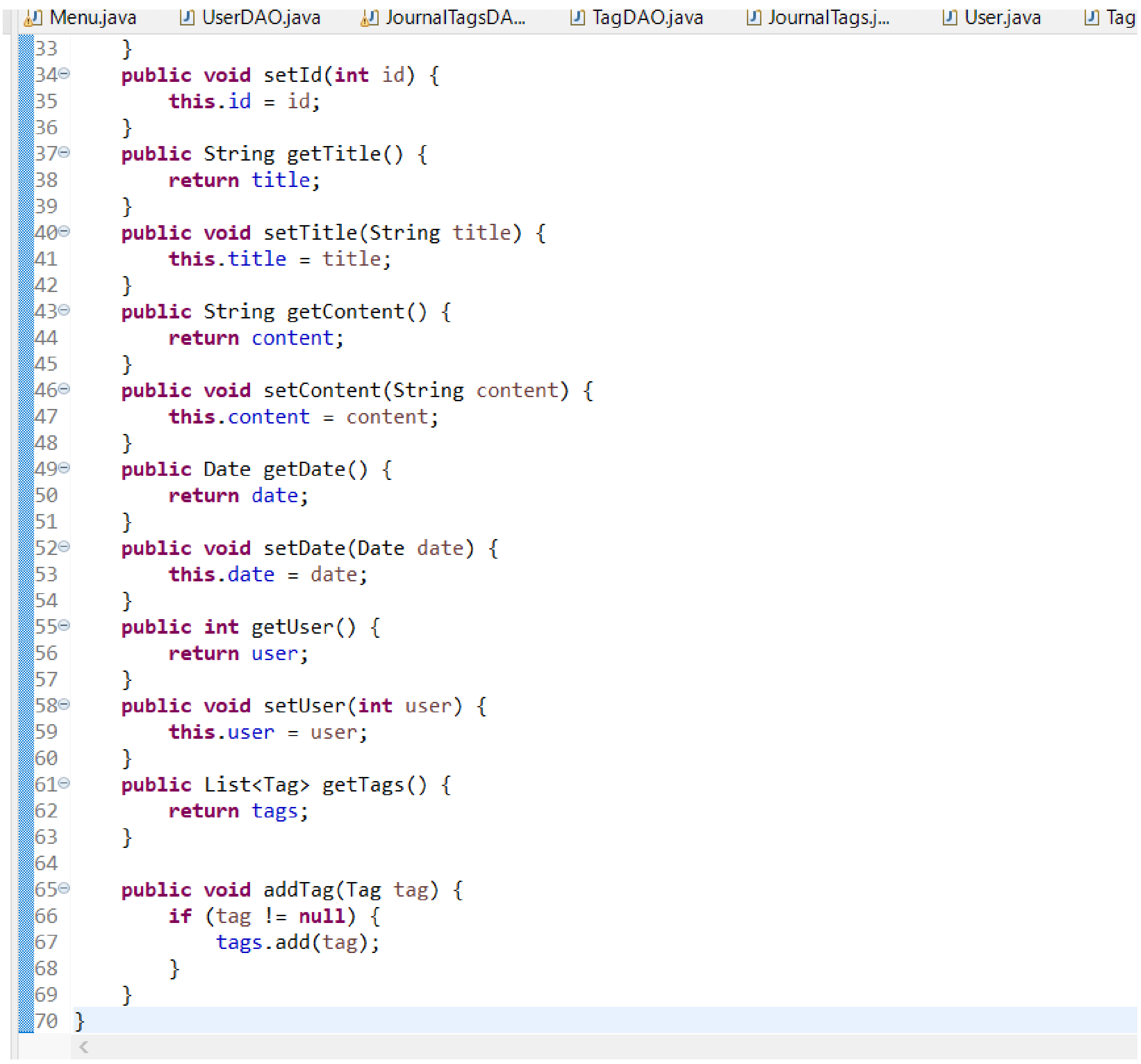
```java
        return journalsByTag;
    }
    public JournalDAO() {
        connection = DBConnection.getConnection();
    }
    public List<Journal> getJournals() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_JOURNALS_QUERY).executeQuery();
        List<Journal> journals = new ArrayList<Journal>();
        while (rs.next()) {
            journals.add(populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
        }
        return journals;
    }
    public Journal getJournalbyId(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_BY_ID_QUERY);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5));
    }
    public int readGetJournalId(String title) throws SQLException {
        int journalId = 0;
        PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_ID_QUERY);
        ps.setString(1, title);
        ResultSet rs = ps.executeQuery();
        rs.next();
        journalId = rs.getInt(1);
        return journalId;
    }
    public void createNewJournal(String entryName, String content, int journalTagId, int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNAL_QUERY);
        ps.setString(1, entryName);
        ps.setString(2, content);
        ps.setInt(3, id);
        ps.executeUpdate();
        journalTagsDAO.createNewJournalTag(readGetJournalId(entryName), journalTagId);
    }
    public void deleteJournalById(int id) throws SQLException {
```



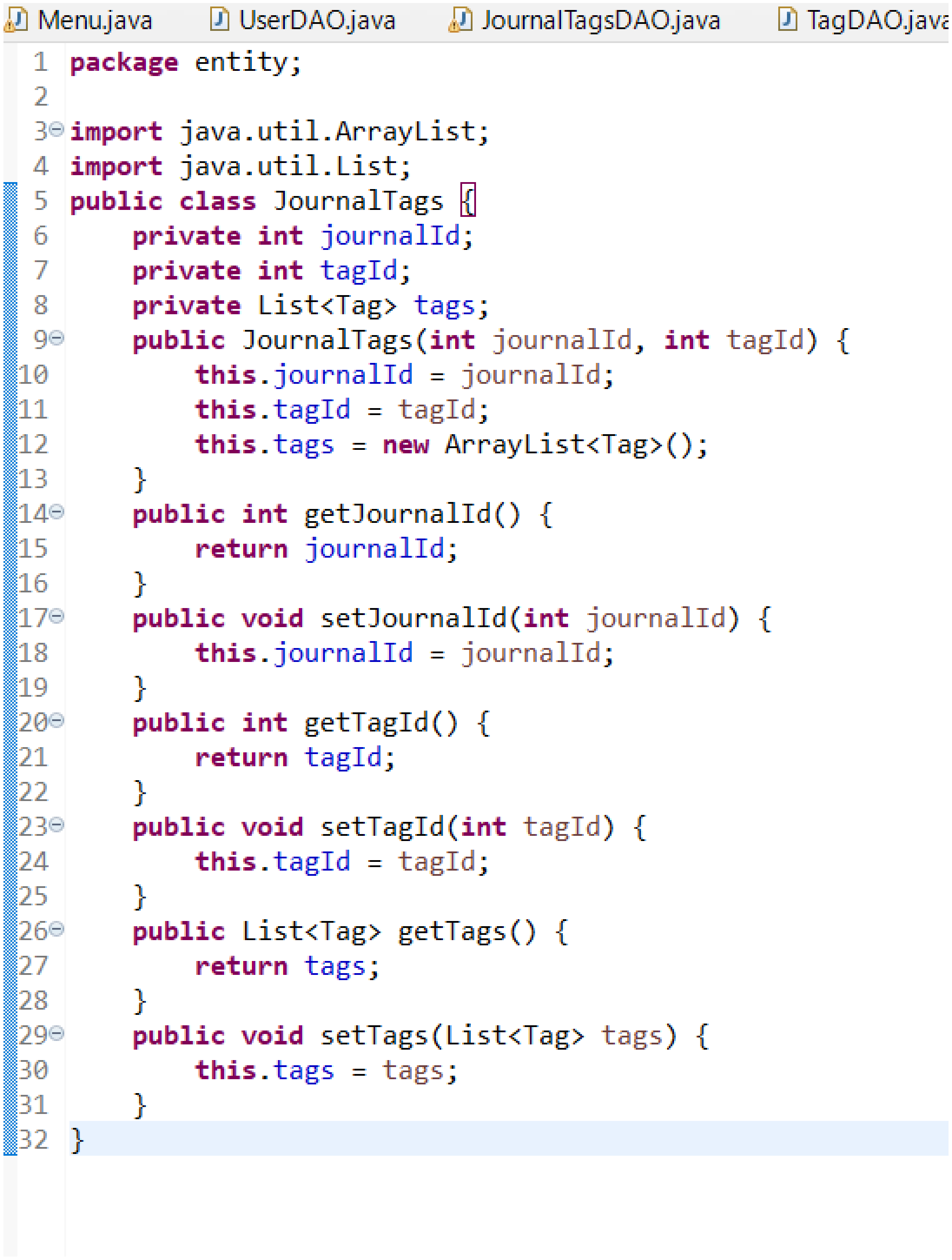
```java
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5));
    }
    public int readGetJournalId(String title) throws SQLException {
        int journalId = 0;
        PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_ID_QUERY);
        ps.setString(1, title);
        ResultSet rs = ps.executeQuery();
        rs.next();
        journalId = rs.getInt(1);
        return journalId;
    }
    public void createNewJournal(String entryName, String content, int journalTagId, int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNAL_QUERY);
        ps.setString(1, entryName);
        ps.setString(2, content);
        ps.setInt(3, id);
        ps.executeUpdate();
        journalTagsDAO.createNewJournalTag(readGetJournalId(entryName), journalTagId);
    }
    public void deleteJournalById(int id) throws SQLException {
        journalTagsDAO.deleteJournalTagByJournalId(id);
        PreparedStatement ps = connection.prepareStatement(DELETE_JOURNAL_BY_ID_QUERY);
        ps.setInt(1, id);
        ps.executeUpdate();
    }
    public void updateJournalById(int id, String content) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_JOURNAL_BY_ID_QUERY);
        ps.setInt(2, id);
        ps.setString(1, content);
        ps.executeUpdate();
    }
    private Journal populateJournal(int id, Date date, String title, String content, int userId) throws SQLException {
        return new Journal(id, date, title, content, userId);
    }
}
```



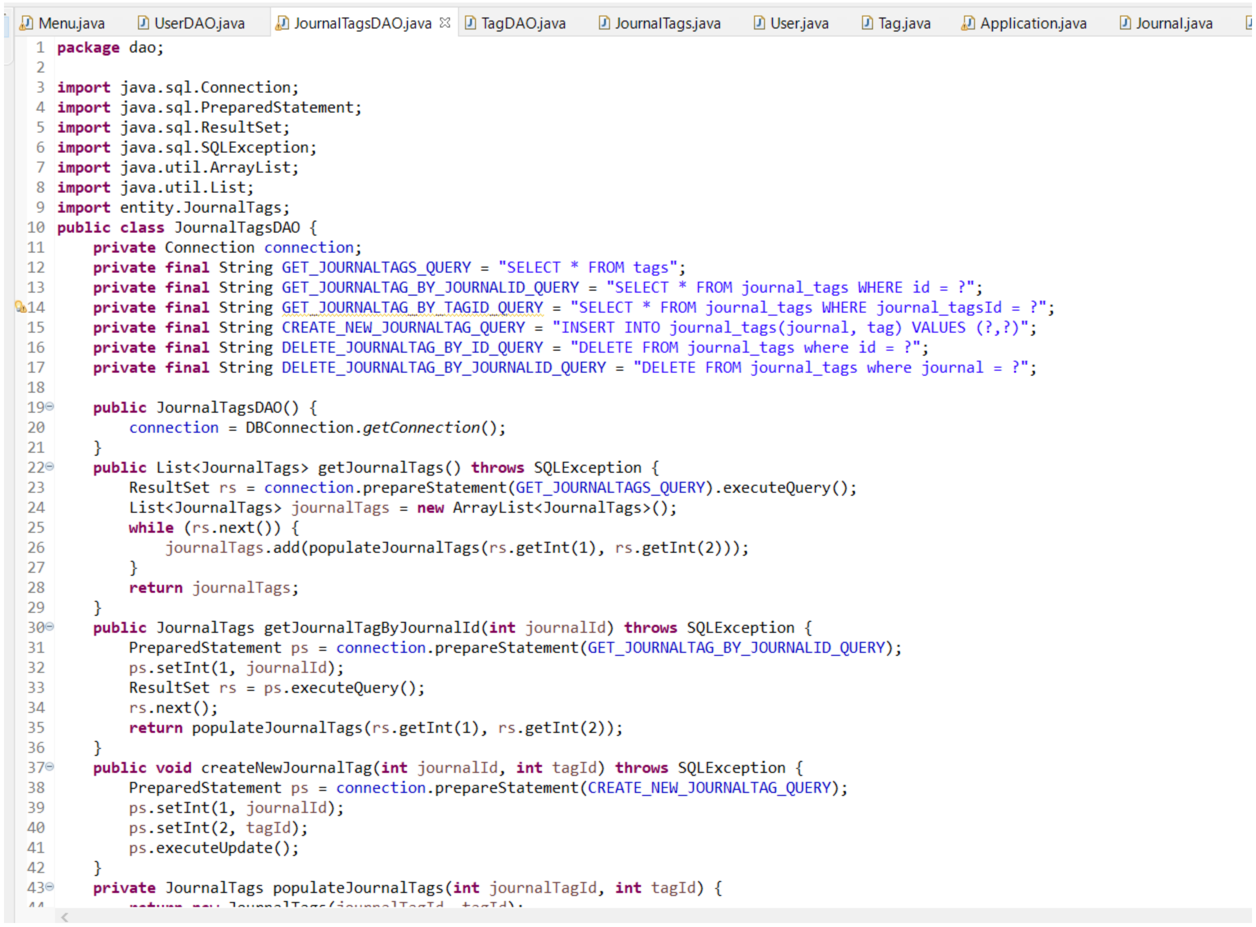
```java
package entity;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
public class Journal {
    private int id;
    private Date date;
    private String title;
    private String content;
    private int user;
    private List<Tag> tags;

    public Journal(int id, Date date, String title, String content) {
        this.setUser(-1);
        this.id = id;
        this.setDate(date);
        this.title = title;
        this.content = content;
        this.tags = new ArrayList<Tag>();
    }

    public Journal(int id2, Date date2, String title2, String content2, int userById) {
        this.setUser(-1);
        this.id = id2;
        this.setDate(date2);
        this.title = title2;
        this.content = content2;
        this.user = userById;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTitle() {
```



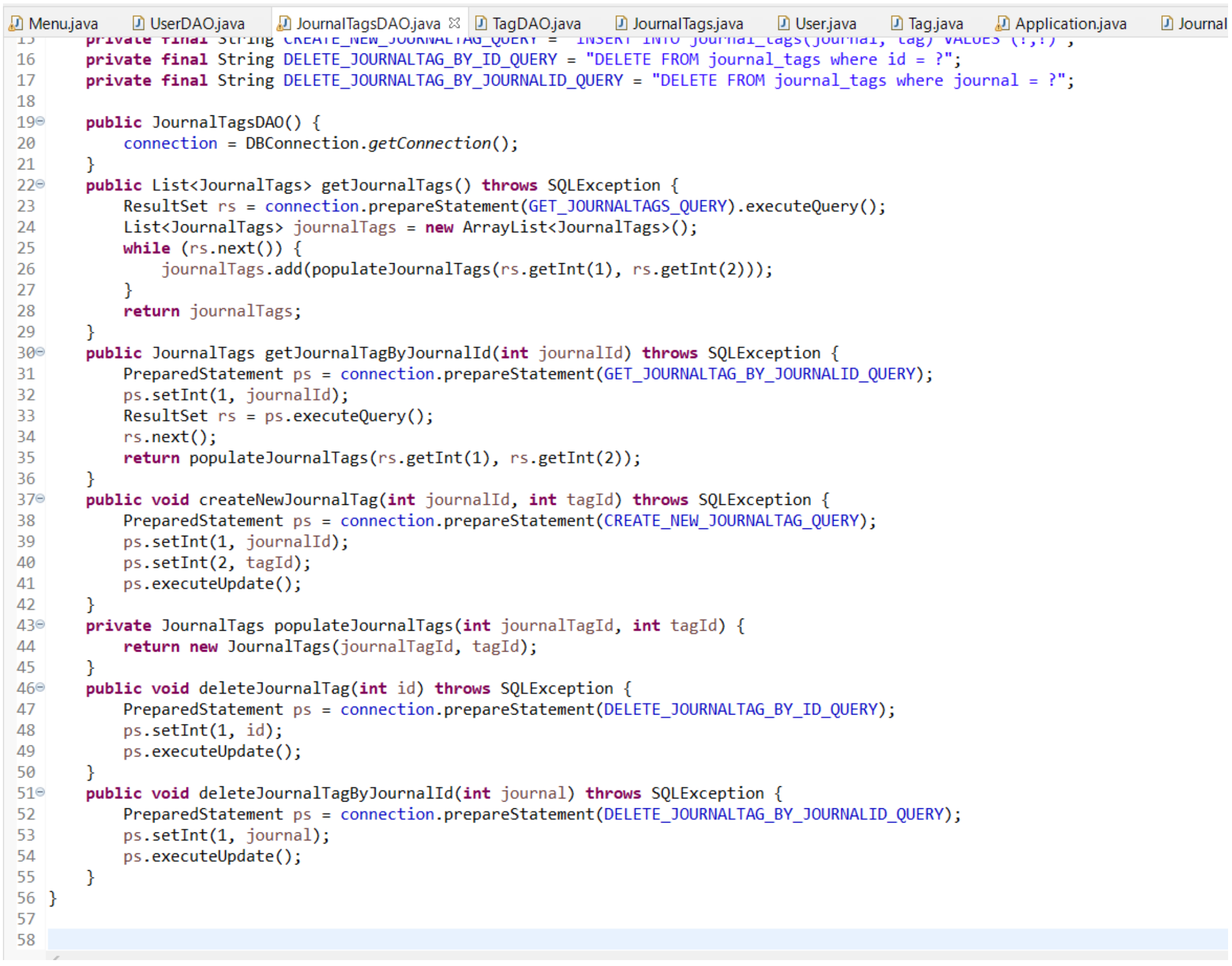
```
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
    public int getUser() {
        return user;
    }
    public void setUser(int user) {
        this.user = user;
    }
    public List<Tag> getTags() {
        return tags;
    }

    public void addTag(Tag tag) {
        if (tag != null) {
            tags.add(tag);
        }
    }
}
```



```java
package entity;

import java.util.ArrayList;
import java.util.List;
public class JournalTags {
    private int journalId;
    private int tagId;
    private List<Tag> tags;
    public JournalTags(int journalId, int tagId) {
        this.journalId = journalId;
        this.tagId = tagId;
        this.tags = new ArrayList<Tag>();
    }
    public int getJournalId() {
        return journalId;
    }
    public void setJournalId(int journalId) {
        this.journalId = journalId;
    }
    public int getTagId() {
        return tagId;
    }
    public void setTagId(int tagId) {
        this.tagId = tagId;
    }
    public List<Tag> getTags() {
        return tags;
    }
    public void setTags(List<Tag> tags) {
        this.tags = tags;
    }
}
```

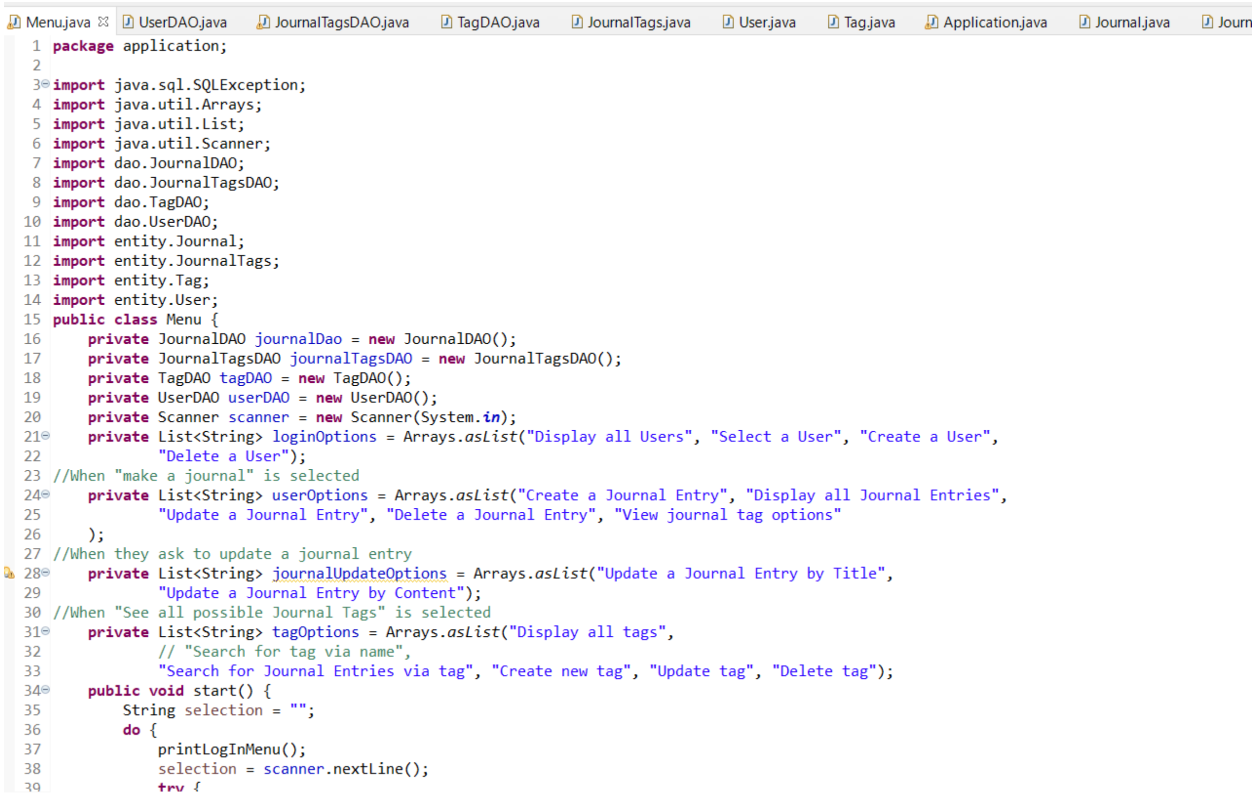


```java
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import entity.JournalTags;
public class JournalTagsDAO {
    private Connection connection;
    private final String GET_JOURNALTAGS_QUERY = "SELECT * FROM tags";
    private final String GET_JOURNALTAG_BY_JOURNALID_QUERY = "SELECT * FROM journal_tags WHERE id = ?";
    private final String GET_JOURNALTAG_BY_TAGID_QUERY = "SELECT * FROM journal_tags WHERE journal_tagsId = ?";
    private final String CREATE_NEW_JOURNALTAG_QUERY = "INSERT INTO journal_tags(journal, tag) VALUES (?,?)";
    private final String DELETE_JOURNALTAG_BY_ID_QUERY = "DELETE FROM journal_tags where id = ?";
    private final String DELETE_JOURNALTAG_BY_JOURNALID_QUERY = "DELETE FROM journal_tags where journal = ?";

    public JournalTagsDAO() {
        connection = DBConnection.getConnection();
    }
    public List<JournalTags> getJournalTags() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_JOURNALTAGS_QUERY).executeQuery();
        List<JournalTags> journalTags = new ArrayList<JournalTags>();
        while (rs.next()) {
            journalTags.add(populateJournalTags(rs.getInt(1), rs.getInt(2)));
        }
        return journalTags;
    }
    public JournalTags getJournalTagByJournalId(int journalId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_JOURNALTAG_BY_JOURNALID_QUERY);
        ps.setInt(1, journalId);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateJournalTags(rs.getInt(1), rs.getInt(2));
    }
    public void createNewJournalTag(int journalId, int tagId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNALTAG_QUERY);
        ps.setInt(1, journalId);
        ps.setInt(2, tagId);
        ps.executeUpdate();
    }
    private JournalTags populateJournalTags(int journalTagId, int tagId) {
```

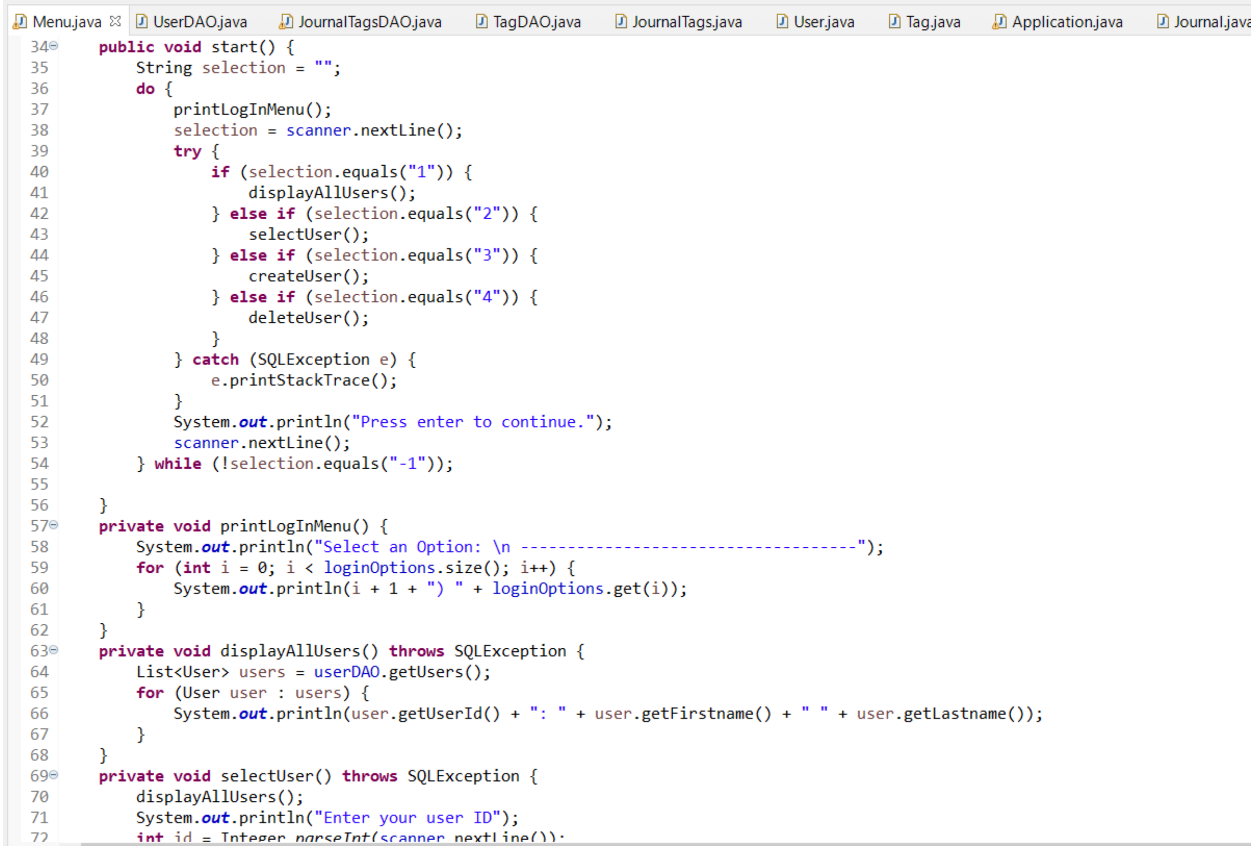


```java
    private final String CREATE_NEW_JOURNALTAG_QUERY = "INSERT INTO journal_tags(journal, tag) VALUES (?,?)";
    private final String DELETE_JOURNALTAG_BY_ID_QUERY = "DELETE FROM journal_tags where id = ?";
    private final String DELETE_JOURNALTAG_BY_JOURNALID_QUERY = "DELETE FROM journal_tags where journal = ?";

    public JournalTagsDAO() {
        connection = DBConnection.getConnection();
    }
    public List<JournalTags> getJournalTags() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_JOURNALTAGS_QUERY).executeQuery();
        List<JournalTags> journalTags = new ArrayList<JournalTags>();
        while (rs.next()) {
            journalTags.add(populateJournalTags(rs.getInt(1), rs.getInt(2)));
        }
        return journalTags;
    }
    public JournalTags getJournalTagByJournalId(int journalId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_JOURNALTAG_BY_JOURNALID_QUERY);
        ps.setInt(1, journalId);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateJournalTags(rs.getInt(1), rs.getInt(2));
    }
    public void createNewJournalTag(int journalId, int tagId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNALTAG_QUERY);
        ps.setInt(1, journalId);
        ps.setInt(2, tagId);
        ps.executeUpdate();
    }
    private JournalTags populateJournalTags(int journalTagId, int tagId) {
        return new JournalTags(journalTagId, tagId);
    }
    public void deleteJournalTag(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_JOURNALTAG_BY_ID_QUERY);
        ps.setInt(1, id);
        ps.executeUpdate();
    }
    public void deleteJournalTagByJournalId(int journal) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_JOURNALTAG_BY_JOURNALID_QUERY);
        ps.setInt(1, journal);
        ps.executeUpdate();
    }
}
```

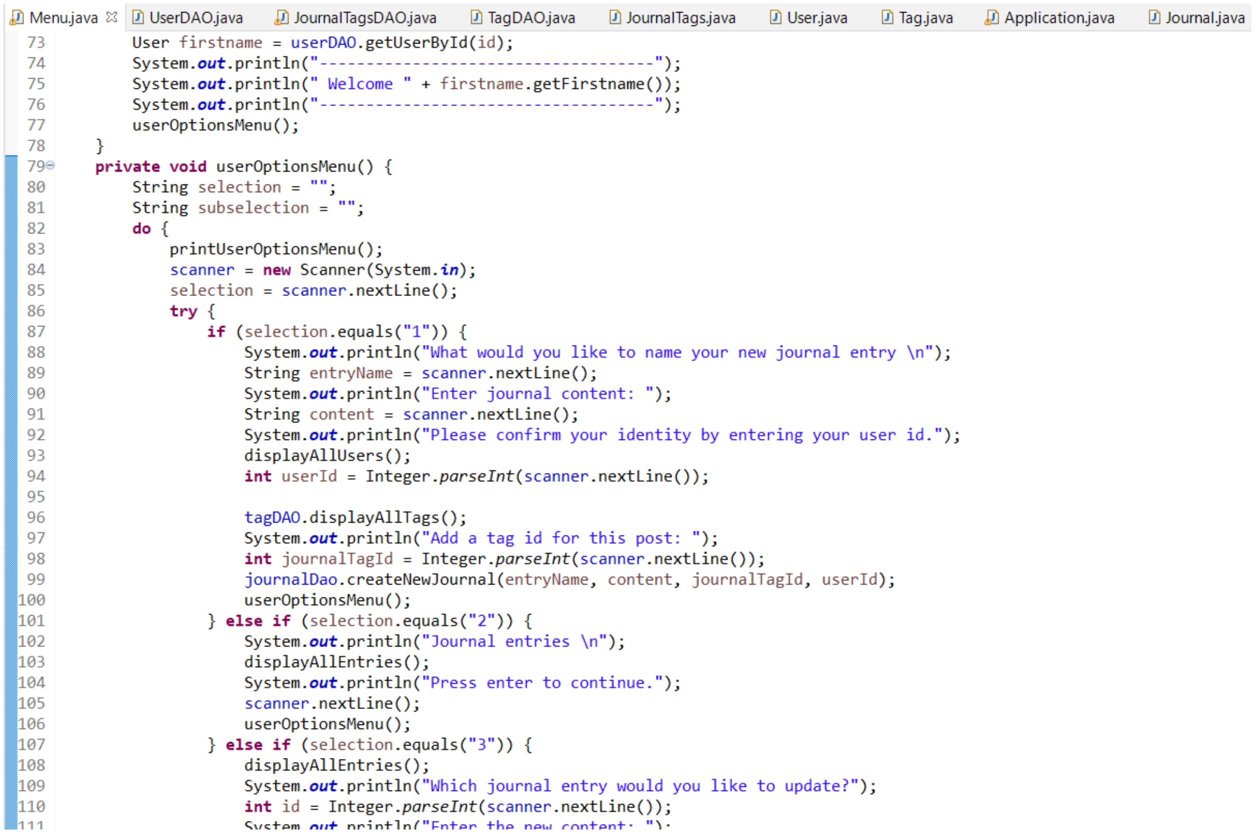
Screenshots of Code:



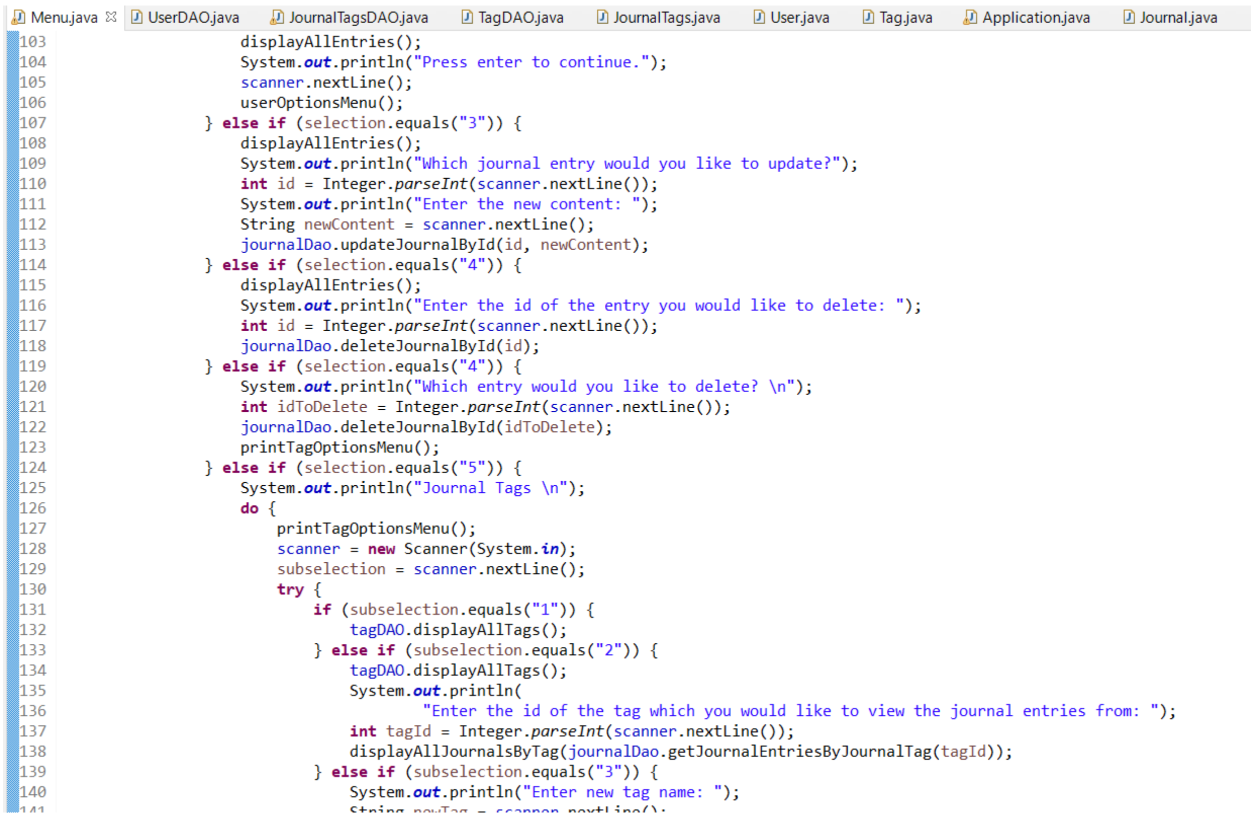
```java
package application;

import java.sql.SQLException;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
import dao.JournalDAO;
import dao.JournalTagsDAO;
import dao.TagDAO;
import dao.UserDAO;
import entity.Journal;
import entity.JournalTags;
import entity.Tag;
import entity.User;
public class Menu {
    private JournalDAO journalDao = new JournalDAO();
    private JournalTagsDAO journalTagsDAO = new JournalTagsDAO();
    private TagDAO tagDAO = new TagDAO();
    private UserDAO userDAO = new UserDAO();
    private Scanner scanner = new Scanner(System.in);
    private List<String> loginOptions = Arrays.asList("Display all Users", "Select a User", "Create a User",
            "Delete a User");
//When "make a journal" is selected
    private List<String> userOptions = Arrays.asList("Create a Journal Entry", "Display all Journal Entries",
            "Update a Journal Entry", "Delete a Journal Entry", "View journal tag options"
    );
//When they ask to update a journal entry
    private List<String> journalUpdateOptions = Arrays.asList("Update a Journal Entry by Title",
            "Update a Journal Entry by Content");
//When "See all possible Journal Tags" is selected
    private List<String> tagOptions = Arrays.asList("Display all tags",
            // "Search for tag via name",
            "Search for Journal Entries via tag", "Create new tag", "Update tag", "Delete tag");
    public void start() {
        String selection = "";
        do {
            printLogInMenu();
            selection = scanner.nextLine();
            try {
```



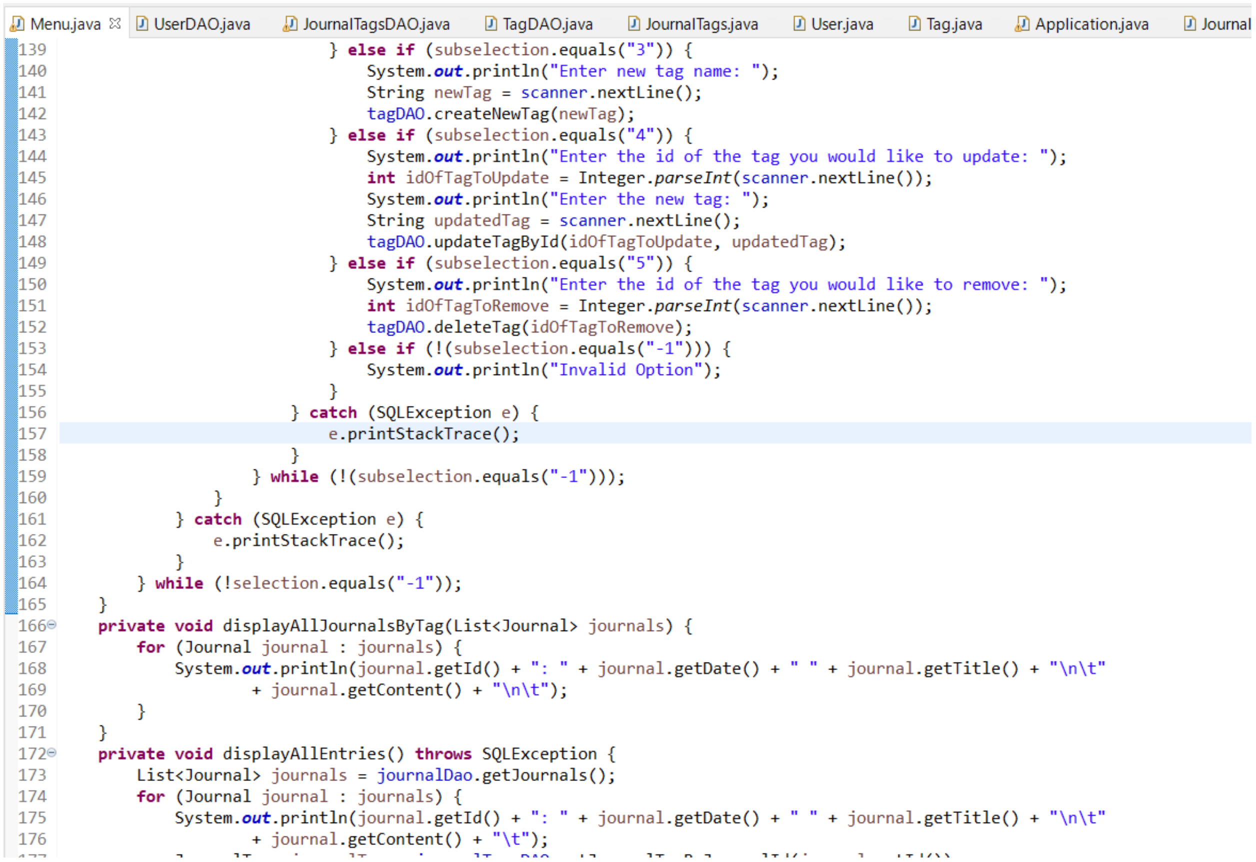
```java
    public void start() {
        String selection = "";
        do {
            printLogInMenu();
            selection = scanner.nextLine();
            try {
                if (selection.equals("1")) {
                    displayAllUsers();
                } else if (selection.equals("2")) {
                    selectUser();
                } else if (selection.equals("3")) {
                    createUser();
                } else if (selection.equals("4")) {
                    deleteUser();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            System.out.println("Press enter to continue.");
            scanner.nextLine();
        } while (!selection.equals("-1"));

    }
    private void printLogInMenu() {
        System.out.println("Select an Option: \n -----------------------------------");
        for (int i = 0; i < loginOptions.size(); i++) {
            System.out.println(i + 1 + ") " + loginOptions.get(i));
        }
    }
    private void displayAllUsers() throws SQLException {
        List<User> users = userDAO.getUsers();
        for (User user : users) {
            System.out.println(user.getUserId() + ": " + user.getFirstname() + " " + user.getLastname());
        }
    }
    private void selectUser() throws SQLException {
        displayAllUsers();
        System.out.println("Enter your user ID");
        int id = Integer.parseInt(scanner.nextLine());
```



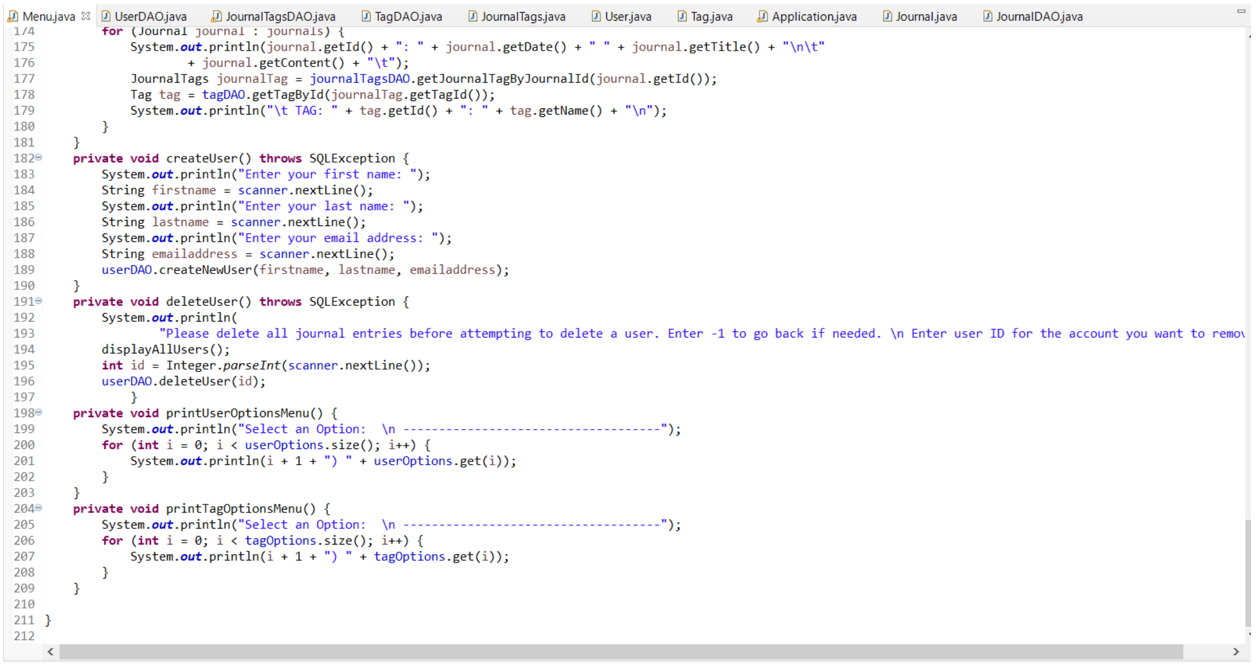
```java
        User firstname = userDAO.getUserById(id);
        System.out.println("------------------------------------");
        System.out.println(" Welcome " + firstname.getFirstname());
        System.out.println("------------------------------------");
        userOptionsMenu();
    }
    private void userOptionsMenu() {
        String selection = "";
        String subselection = "";
        do {
            printUserOptionsMenu();
            scanner = new Scanner(System.in);
            selection = scanner.nextLine();
            try {
                if (selection.equals("1")) {
                    System.out.println("What would you like to name your new journal entry \n");
                    String entryName = scanner.nextLine();
                    System.out.println("Enter journal content: ");
                    String content = scanner.nextLine();
                    System.out.println("Please confirm your identity by entering your user id.");
                    displayAllUsers();
                    int userId = Integer.parseInt(scanner.nextLine());

                    tagDAO.displayAllTags();
                    System.out.println("Add a tag id for this post: ");
                    int journalTagId = Integer.parseInt(scanner.nextLine());
                    journalDao.createNewJournal(entryName, content, journalTagId, userId);
                    userOptionsMenu();
                } else if (selection.equals("2")) {
                    System.out.println("Journal entries \n");
                    displayAllEntries();
                    System.out.println("Press enter to continue.");
                    scanner.nextLine();
                    userOptionsMenu();
                } else if (selection.equals("3")) {
                    displayAllEntries();
                    System.out.println("Which journal entry would you like to update?");
                    int id = Integer.parseInt(scanner.nextLine());
                    System.out.println("Enter the new content: ");
```



```java
                    displayAllEntries();
                    System.out.println("Press enter to continue.");
                    scanner.nextLine();
                    userOptionsMenu();
                } else if (selection.equals("3")) {
                    displayAllEntries();
                    System.out.println("Which journal entry would you like to update?");
                    int id = Integer.parseInt(scanner.nextLine());
                    System.out.println("Enter the new content: ");
                    String newContent = scanner.nextLine();
                    journalDao.updateJournalById(id, newContent);
                } else if (selection.equals("4")) {
                    displayAllEntries();
                    System.out.println("Enter the id of the entry you would like to delete: ");
                    int id = Integer.parseInt(scanner.nextLine());
                    journalDao.deleteJournalById(id);
                } else if (selection.equals("4")) {
                    System.out.println("Which entry would you like to delete? \n");
                    int idToDelete = Integer.parseInt(scanner.nextLine());
                    journalDao.deleteJournalById(idToDelete);
                    printTagOptionsMenu();
                } else if (selection.equals("5")) {
                    System.out.println("Journal Tags \n");
                    do {
                        printTagOptionsMenu();
                        scanner = new Scanner(System.in);
                        subselection = scanner.nextLine();
                        try {
                            if (subselection.equals("1")) {
                                tagDAO.displayAllTags();
                            } else if (subselection.equals("2")) {
                                tagDAO.displayAllTags();
                                System.out.println(
                                        "Enter the id of the tag which you would like to view the journal entries from: ");
                                int tagId = Integer.parseInt(scanner.nextLine());
                                displayAllJournalsByTag(journalDao.getJournalEntriesByJournalTag(tagId));
                            } else if (subselection.equals("3")) {
                                System.out.println("Enter new tag name: ");
                                String newTag = scanner.nextLine();
```

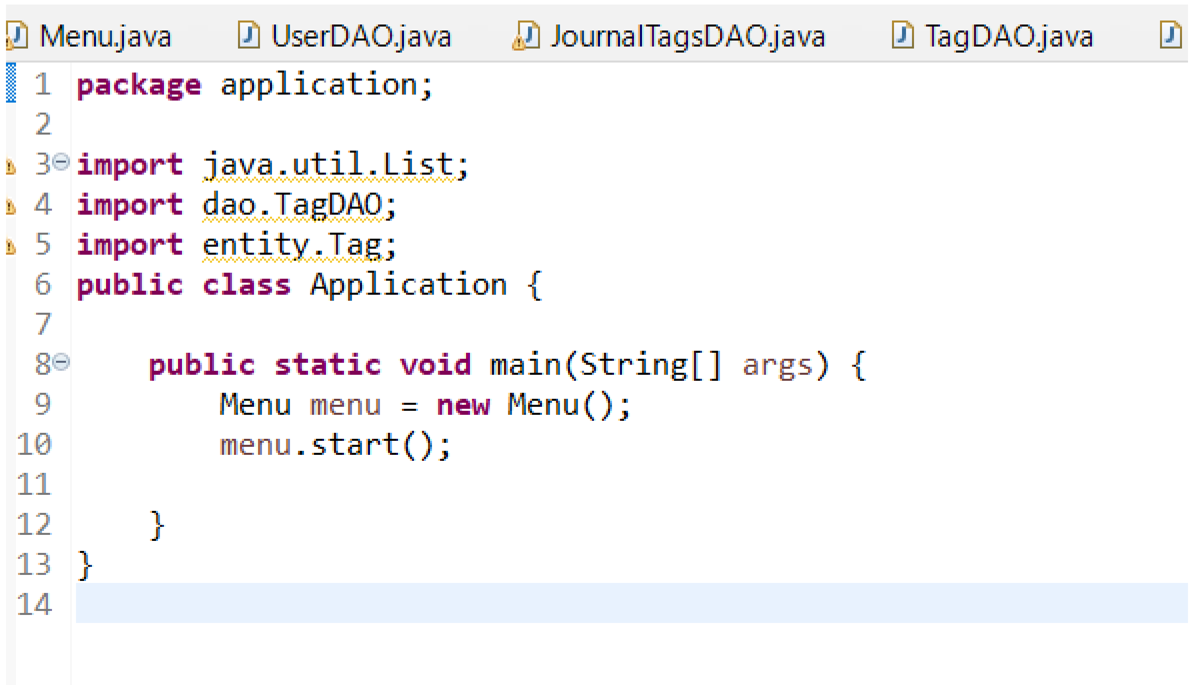


```java
                            } else if (subselection.equals("3")) {
                                System.out.println("Enter new tag name: ");
                                String newTag = scanner.nextLine();
                                tagDAO.createNewTag(newTag);
                            } else if (subselection.equals("4")) {
                                System.out.println("Enter the id of the tag you would like to update: ");
                                int idOfTagToUpdate = Integer.parseInt(scanner.nextLine());
                                System.out.println("Enter the new tag: ");
                                String updatedTag = scanner.nextLine();
                                tagDAO.updateTagById(idOfTagToUpdate, updatedTag);
                            } else if (subselection.equals("5")) {
                                System.out.println("Enter the id of the tag you would like to remove: ");
                                int idOfTagToRemove = Integer.parseInt(scanner.nextLine());
                                tagDAO.deleteTag(idOfTagToRemove);
                            } else if (!(subselection.equals("-1"))) {
                                System.out.println("Invalid Option");
                            }
                        } catch (SQLException e) {
                            e.printStackTrace();
                        }
                    } while (!(subselection.equals("-1")));
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        } while (!selection.equals("-1"));
    }
    private void displayAllJournalsByTag(List<Journal> journals) {
        for (Journal journal : journals) {
            System.out.println(journal.getId() + ": " + journal.getDate() + " " + journal.getTitle() + "\n\t"
                    + journal.getContent() + "\n\t");
        }
    }
    private void displayAllEntries() throws SQLException {
        List<Journal> journals = journalDao.getJournals();
        for (Journal journal : journals) {
            System.out.println(journal.getId() + ": " + journal.getDate() + " " + journal.getTitle() + "\n\t"
                    + journal.getContent() + "\t");
```



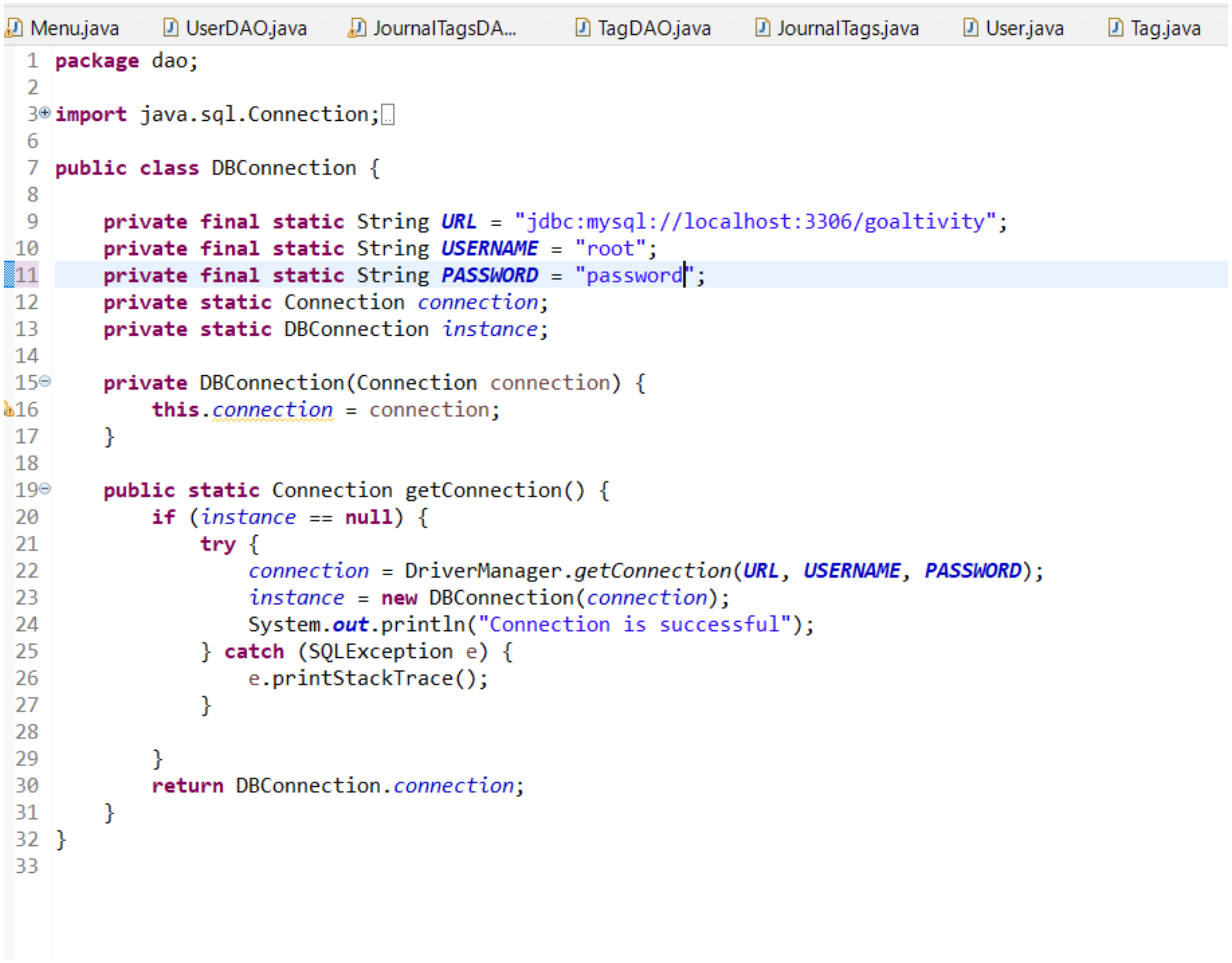
```java
        for (Journal journal : journals) {
            System.out.println(journal.getId() + ": " + journal.getDate() + " " + journal.getTitle() + "\n\t"
                    + journal.getContent() + "\t");
            JournalTags journalTag = journalTagsDAO.getJournalTagByJournalId(journal.getId());
            Tag tag = tagDAO.getTagById(journalTag.getTagId());
            System.out.println("\t TAG: " + tag.getId() + ": " + tag.getName() + "\n");
        }
    }
    private void createUser() throws SQLException {
        System.out.println("Enter your first name: ");
        String firstname = scanner.nextLine();
        System.out.println("Enter your last name: ");
        String lastname = scanner.nextLine();
        System.out.println("Enter your email address: ");
        String emailaddress = scanner.nextLine();
        userDAO.createNewUser(firstname, lastname, emailaddress);
    }
    private void deleteUser() throws SQLException {
        System.out.println(
                "Please delete all journal entries before attempting to delete a user. Enter -1 to go back if needed. \n Enter user ID for the account you want to remov
        displayAllUsers();
        int id = Integer.parseInt(scanner.nextLine());
        userDAO.deleteUser(id);
            }
    private void printUserOptionsMenu() {
        System.out.println("Select an Option:  \n -----------------------------------");
        for (int i = 0; i < userOptions.size(); i++) {
            System.out.println(i + 1 + ") " + userOptions.get(i));
        }
    }
    private void printTagOptionsMenu() {
        System.out.println("Select an Option:  \n -----------------------------------");
        for (int i = 0; i < tagOptions.size(); i++) {
            System.out.println(i + 1 + ") " + tagOptions.get(i));
        }
    }

}
```

Application



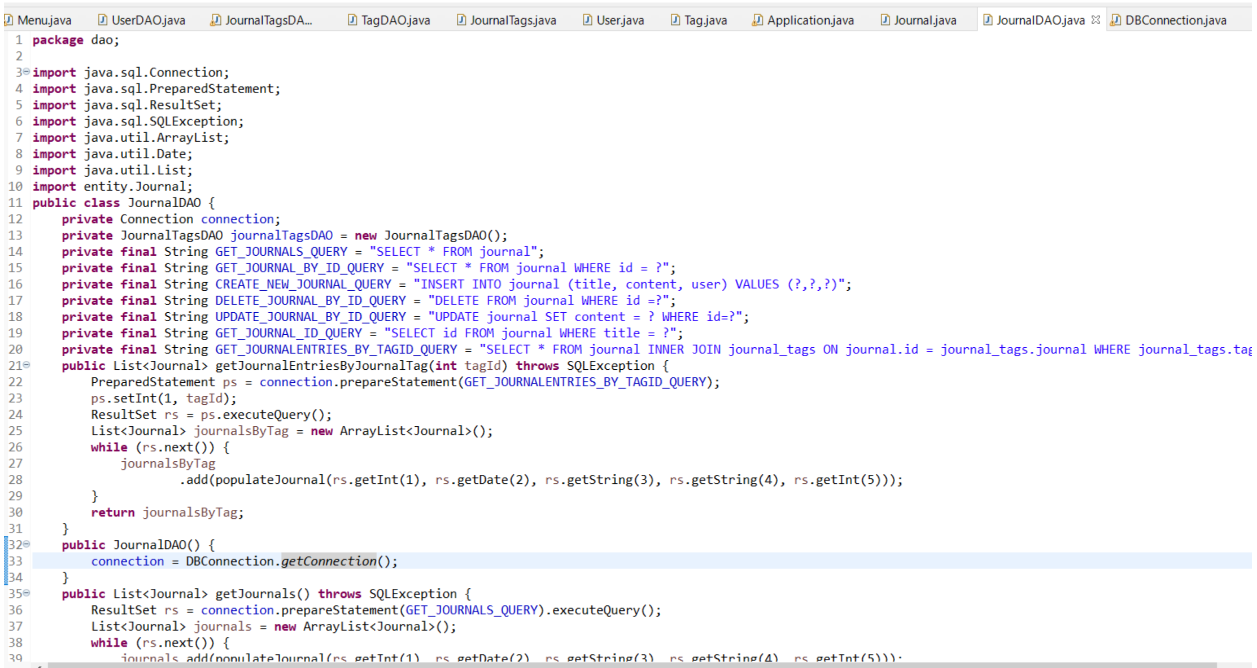
```
package application;

import java.util.List;
import dao.TagDAO;
import entity.Tag;
public class Application {

	public static void main(String[] args) {
		Menu menu = new Menu();
		menu.start();

	}
}

```

(Dao) DBConnection

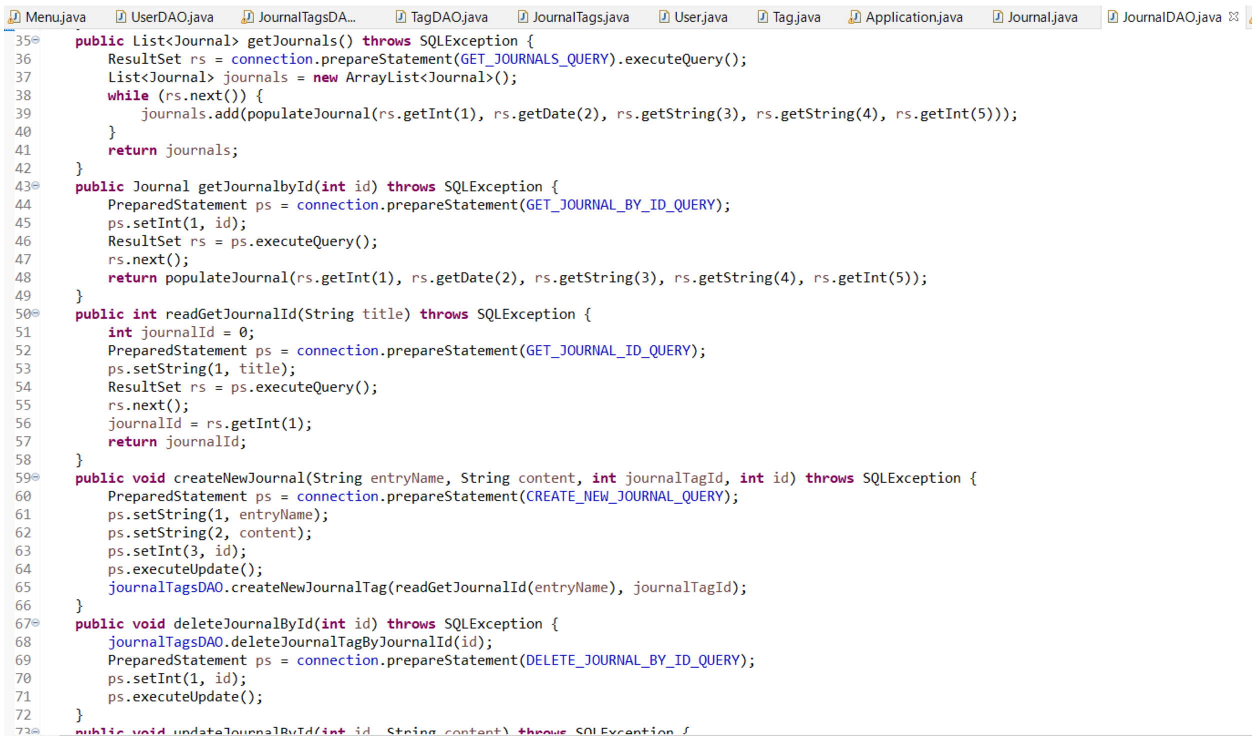


```java
package dao;

import java.sql.Connection;

public class DBConnection {

    private final static String URL = "jdbc:mysql://localhost:3306/goaltivity";
    private final static String USERNAME = "root";
    private final static String PASSWORD = "password";
    private static Connection connection;
    private static DBConnection instance;

    private DBConnection(Connection connection) {
        this.connection = connection;
    }

    public static Connection getConnection() {
        if (instance == null) {
            try {
                connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
                instance = new DBConnection(connection);
                System.out.println("Connection is successful");
            } catch (SQLException e) {
                e.printStackTrace();
            }

        }
        return DBConnection.connection;
    }
}
```

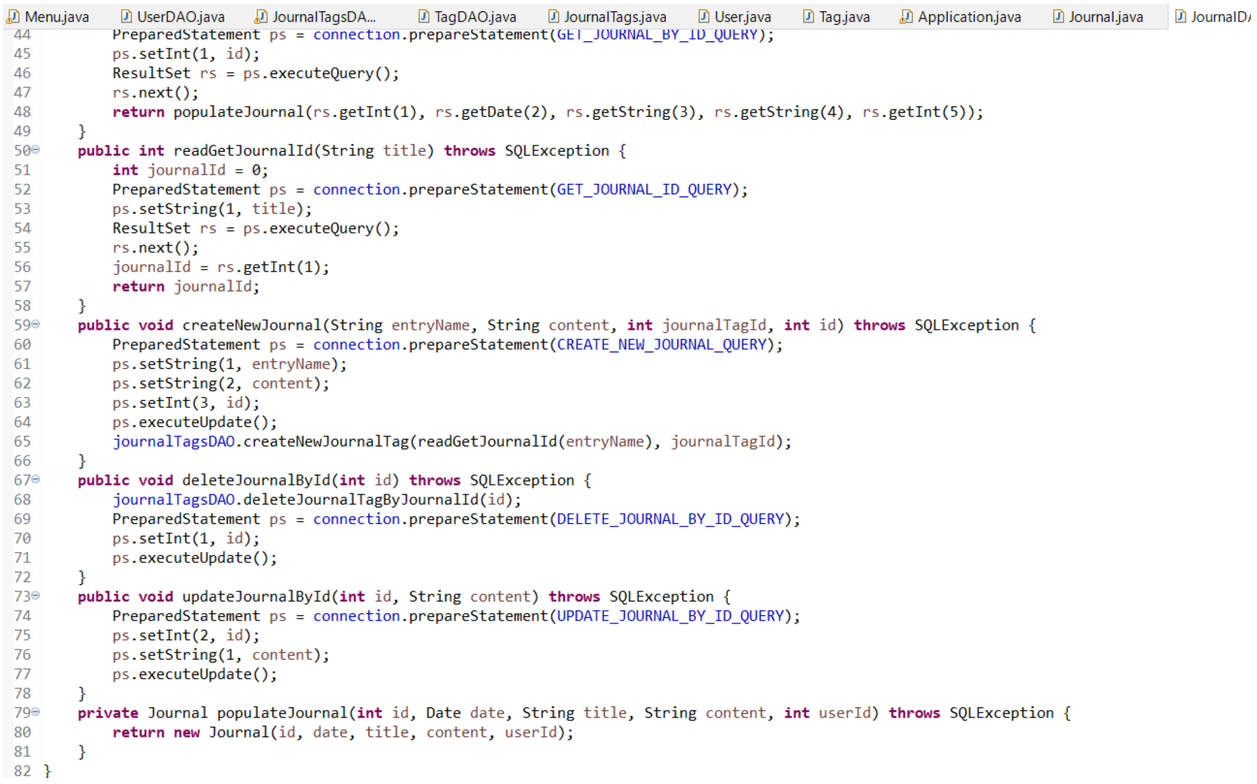
JournalDAO



```java
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import entity.Journal;
public class JournalDAO {
    private Connection connection;
    private JournalTagsDAO journalTagsDAO = new JournalTagsDAO();
    private final String GET_JOURNALS_QUERY = "SELECT * FROM journal";
    private final String GET_JOURNAL_BY_ID_QUERY = "SELECT * FROM journal WHERE id = ?";
    private final String CREATE_NEW_JOURNAL_QUERY = "INSERT INTO journal (title, content, user) VALUES (?,?,?)";
    private final String DELETE_JOURNAL_BY_ID_QUERY = "DELETE FROM journal WHERE id =?";
    private final String UPDATE_JOURNAL_BY_ID_QUERY = "UPDATE journal SET content = ? WHERE id=?";
    private final String GET_JOURNAL_ID_QUERY = "SELECT id FROM journal WHERE title = ?";
    private final String GET_JOURNALENTRIES_BY_TAGID_QUERY = "SELECT * FROM journal INNER JOIN journal_tags ON journal.id = journal_tags.journal WHERE journal_tags.tag
    public List<Journal> getJournalEntriesByJournalTag(int tagId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_JOURNALENTRIES_BY_TAGID_QUERY);
        ps.setInt(1, tagId);
        ResultSet rs = ps.executeQuery();
        List<Journal> journalsByTag = new ArrayList<Journal>();
        while (rs.next()) {
            journalsByTag
                    .add(populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
        }
        return journalsByTag;
    }
    public JournalDAO() {
        connection = DBConnection.getConnection();
    }
    public List<Journal> getJournals() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_JOURNALS_QUERY).executeQuery();
        List<Journal> journals = new ArrayList<Journal>();
        while (rs.next()) {
            journals.add(populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
```

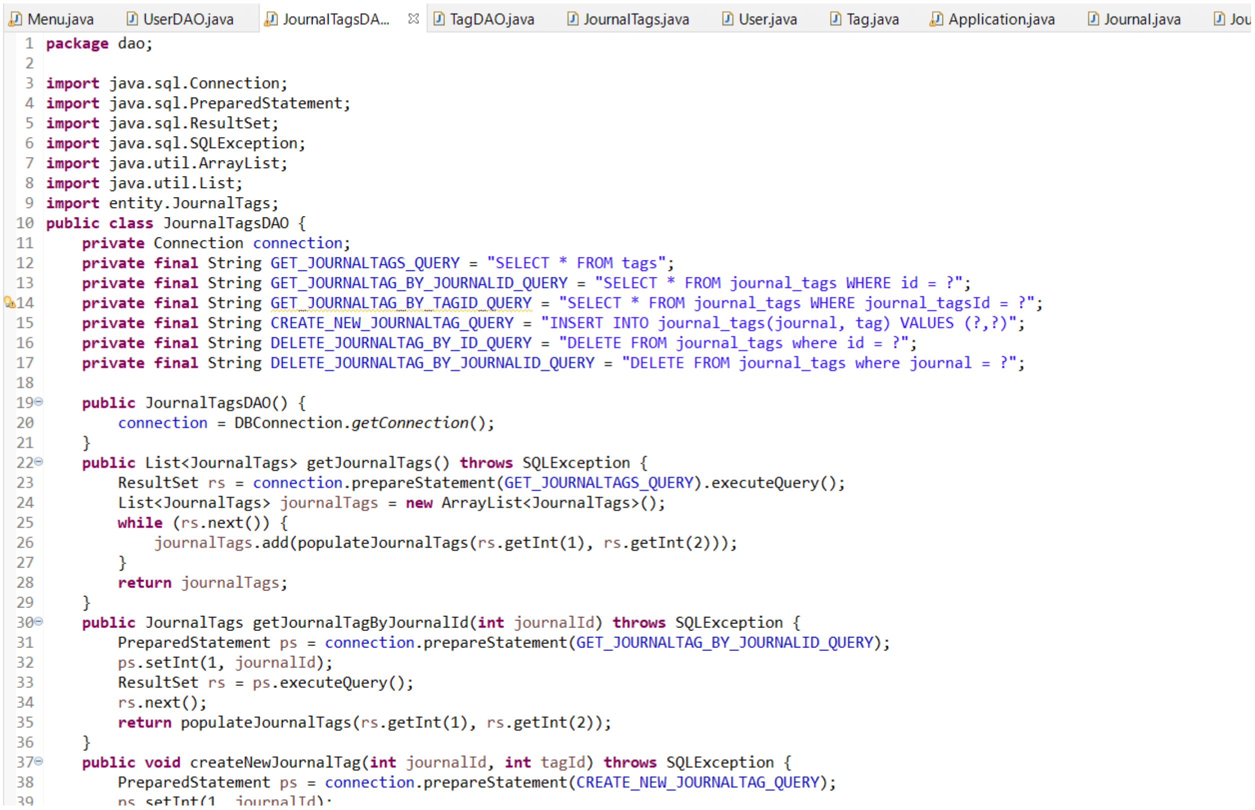


```java
    public List<Journal> getJournals() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_JOURNALS_QUERY).executeQuery();
        List<Journal> journals = new ArrayList<Journal>();
        while (rs.next()) {
            journals.add(populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5)));
        }
        return journals;
    }
    public Journal getJournalbyId(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_BY_ID_QUERY);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5));
    }
    public int readGetJournalId(String title) throws SQLException {
        int journalId = 0;
        PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_ID_QUERY);
        ps.setString(1, title);
        ResultSet rs = ps.executeQuery();
        rs.next();
        journalId = rs.getInt(1);
        return journalId;
    }
    public void createNewJournal(String entryName, String content, int journalTagId, int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNAL_QUERY);
        ps.setString(1, entryName);
        ps.setString(2, content);
        ps.setInt(3, id);
        ps.executeUpdate();
        journalTagsDAO.createNewJournalTag(readGetJournalId(entryName), journalTagId);
    }
    public void deleteJournalById(int id) throws SQLException {
        journalTagsDAO.deleteJournalTagByJournalId(id);
        PreparedStatement ps = connection.prepareStatement(DELETE_JOURNAL_BY_ID_QUERY);
        ps.setInt(1, id);
        ps.executeUpdate();
    }
    public void updateJournalById(int id, String content) throws SQLException {
```

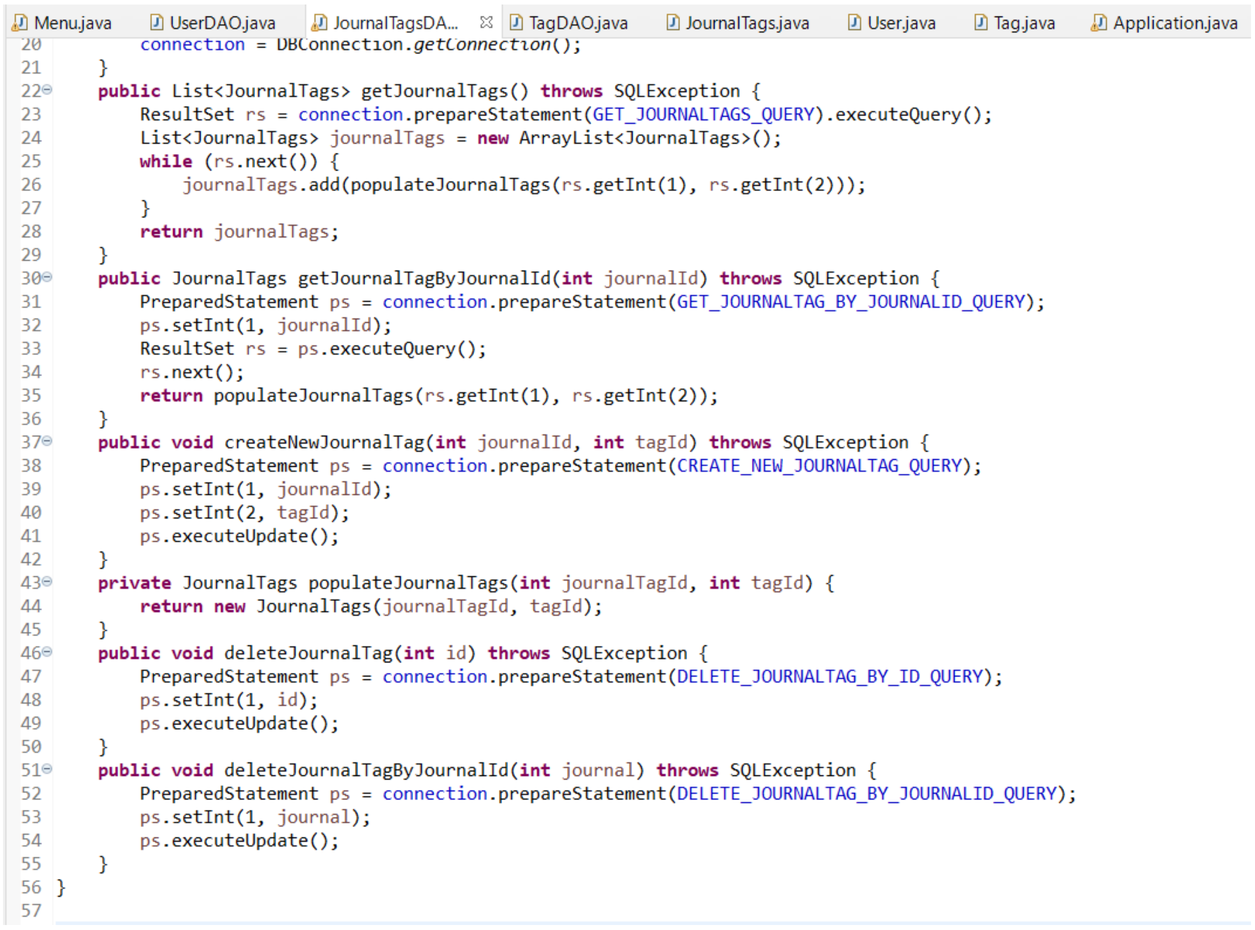


```java
        PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_BY_ID_QUERY);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateJournal(rs.getInt(1), rs.getDate(2), rs.getString(3), rs.getString(4), rs.getInt(5));
    }
    public int readGetJournalId(String title) throws SQLException {
        int journalId = 0;
        PreparedStatement ps = connection.prepareStatement(GET_JOURNAL_ID_QUERY);
        ps.setString(1, title);
        ResultSet rs = ps.executeQuery();
        rs.next();
        journalId = rs.getInt(1);
        return journalId;
    }
    public void createNewJournal(String entryName, String content, int journalTagId, int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNAL_QUERY);
        ps.setString(1, entryName);
        ps.setString(2, content);
        ps.setInt(3, id);
        ps.executeUpdate();
        journalTagsDAO.createNewJournalTag(readGetJournalId(entryName), journalTagId);
    }
    public void deleteJournalById(int id) throws SQLException {
        journalTagsDAO.deleteJournalTagByJournalId(id);
        PreparedStatement ps = connection.prepareStatement(DELETE_JOURNAL_BY_ID_QUERY);
        ps.setInt(1, id);
        ps.executeUpdate();
    }
    public void updateJournalById(int id, String content) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_JOURNAL_BY_ID_QUERY);
        ps.setInt(2, id);
        ps.setString(1, content);
        ps.executeUpdate();
    }
    private Journal populateJournal(int id, Date date, String title, String content, int userId) throws SQLException {
        return new Journal(id, date, title, content, userId);
    }
}
```

JournalTagsDAO

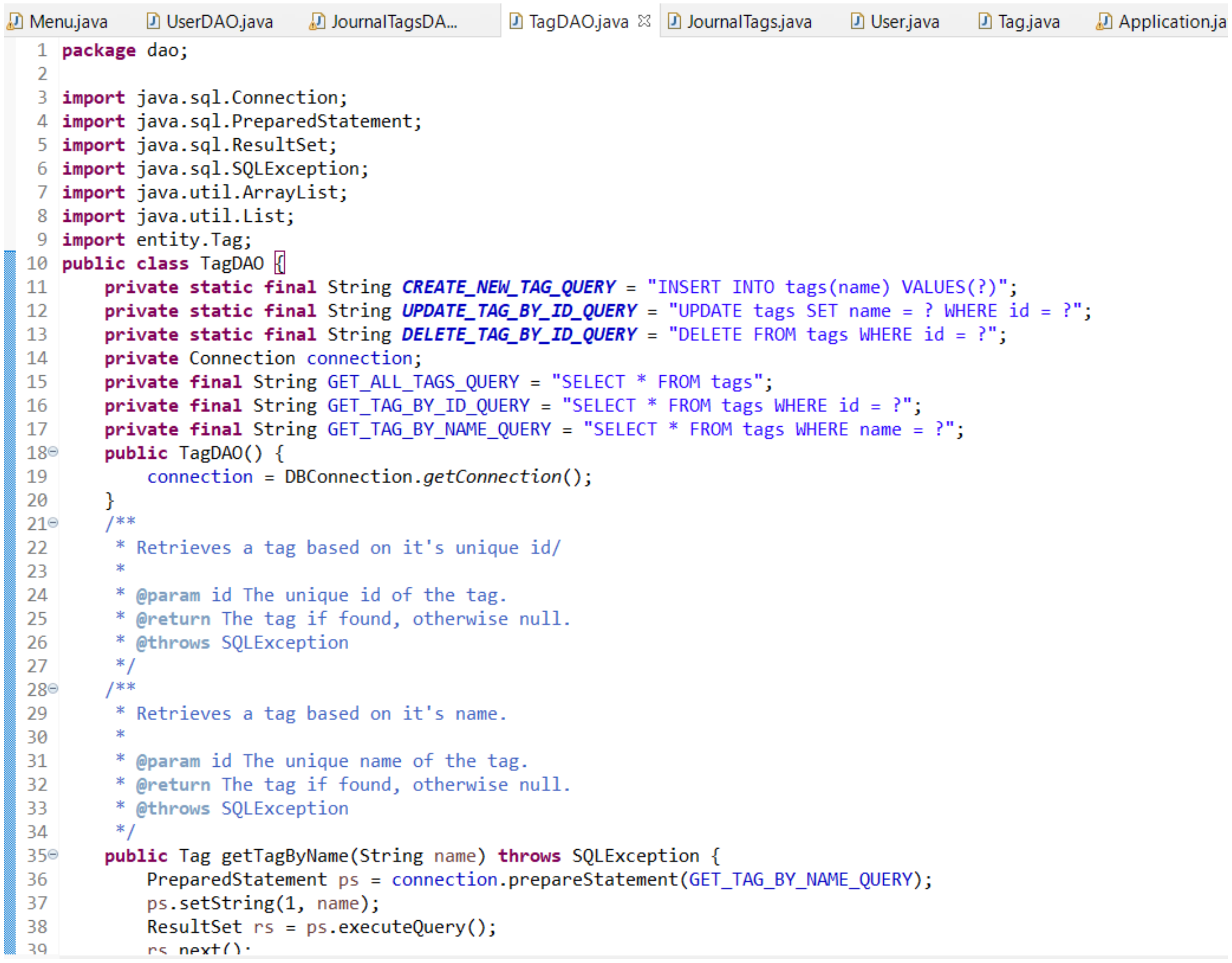


```java
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import entity.JournalTags;
public class JournalTagsDAO {
    private Connection connection;
    private final String GET_JOURNALTAGS_QUERY = "SELECT * FROM tags";
    private final String GET_JOURNALTAG_BY_JOURNALID_QUERY = "SELECT * FROM journal_tags WHERE id = ?";
    private final String GET_JOURNALTAG_BY_TAGID_QUERY = "SELECT * FROM journal_tags WHERE journal_tagsId = ?";
    private final String CREATE_NEW_JOURNALTAG_QUERY = "INSERT INTO journal_tags(journal, tag) VALUES (?,?)";
    private final String DELETE_JOURNALTAG_BY_ID_QUERY = "DELETE FROM journal_tags where id = ?";
    private final String DELETE_JOURNALTAG_BY_JOURNALID_QUERY = "DELETE FROM journal_tags where journal = ?";

    public JournalTagsDAO() {
        connection = DBConnection.getConnection();
    }
    public List<JournalTags> getJournalTags() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_JOURNALTAGS_QUERY).executeQuery();
        List<JournalTags> journalTags = new ArrayList<JournalTags>();
        while (rs.next()) {
            journalTags.add(populateJournalTags(rs.getInt(1), rs.getInt(2)));
        }
        return journalTags;
    }
    public JournalTags getJournalTagByJournalId(int journalId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_JOURNALTAG_BY_JOURNALID_QUERY);
        ps.setInt(1, journalId);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateJournalTags(rs.getInt(1), rs.getInt(2));
    }
    public void createNewJournalTag(int journalId, int tagId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNALTAG_QUERY);
        ps.setInt(1, journalId);
```

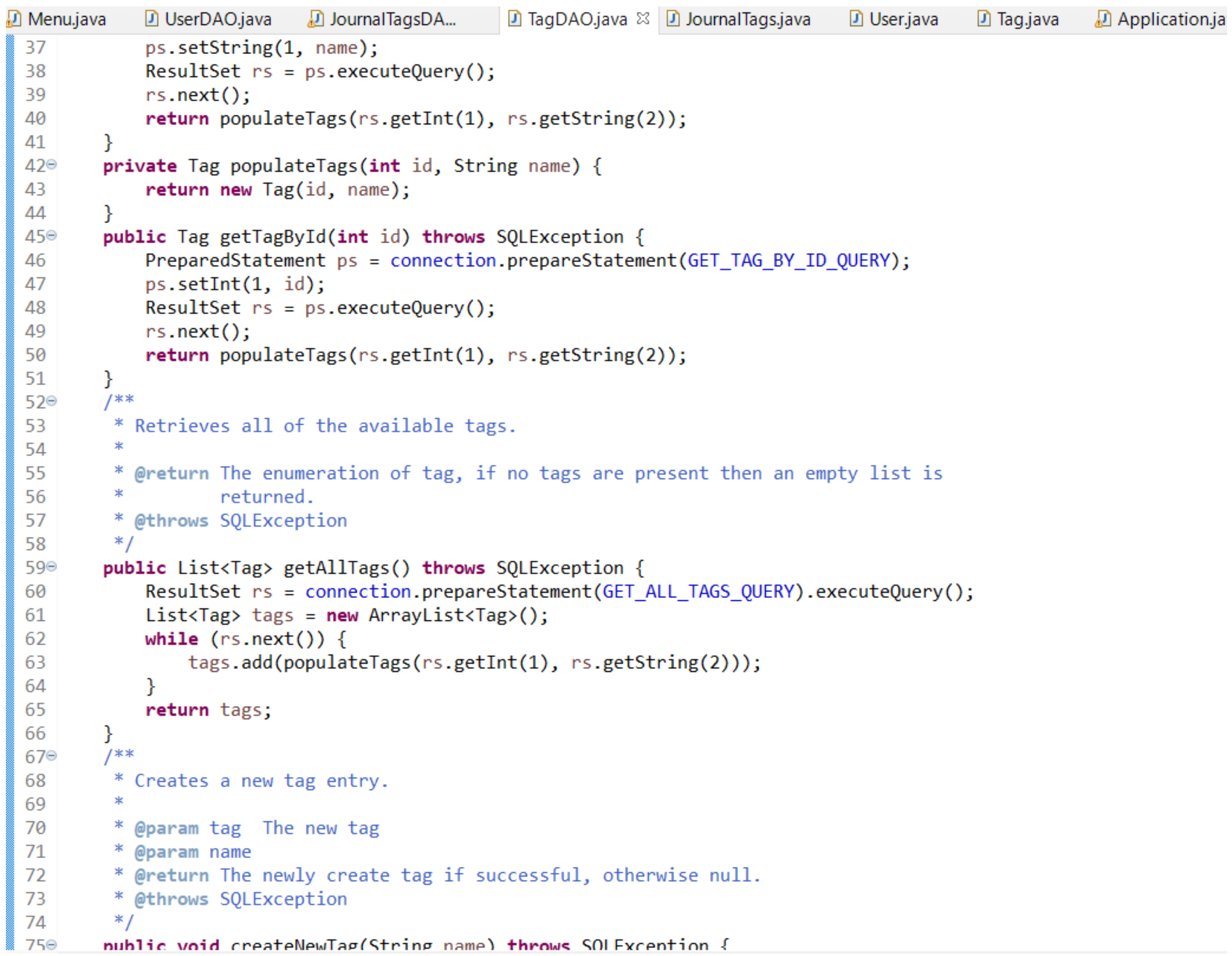


```
        connection = DBConnection.getConnection();
    }
    public List<JournalTags> getJournalTags() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_JOURNALTAGS_QUERY).executeQuery();
        List<JournalTags> journalTags = new ArrayList<JournalTags>();
        while (rs.next()) {
            journalTags.add(populateJournalTags(rs.getInt(1), rs.getInt(2)));
        }
        return journalTags;
    }
    public JournalTags getJournalTagByJournalId(int journalId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_JOURNALTAG_BY_JOURNALID_QUERY);
        ps.setInt(1, journalId);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateJournalTags(rs.getInt(1), rs.getInt(2));
    }
    public void createNewJournalTag(int journalId, int tagId) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_JOURNALTAG_QUERY);
        ps.setInt(1, journalId);
        ps.setInt(2, tagId);
        ps.executeUpdate();
    }
    private JournalTags populateJournalTags(int journalTagId, int tagId) {
        return new JournalTags(journalTagId, tagId);
    }
    public void deleteJournalTag(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_JOURNALTAG_BY_ID_QUERY);
        ps.setInt(1, id);
        ps.executeUpdate();
    }
    public void deleteJournalTagByJournalId(int journal) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_JOURNALTAG_BY_JOURNALID_QUERY);
        ps.setInt(1, journal);
        ps.executeUpdate();
    }
}
```

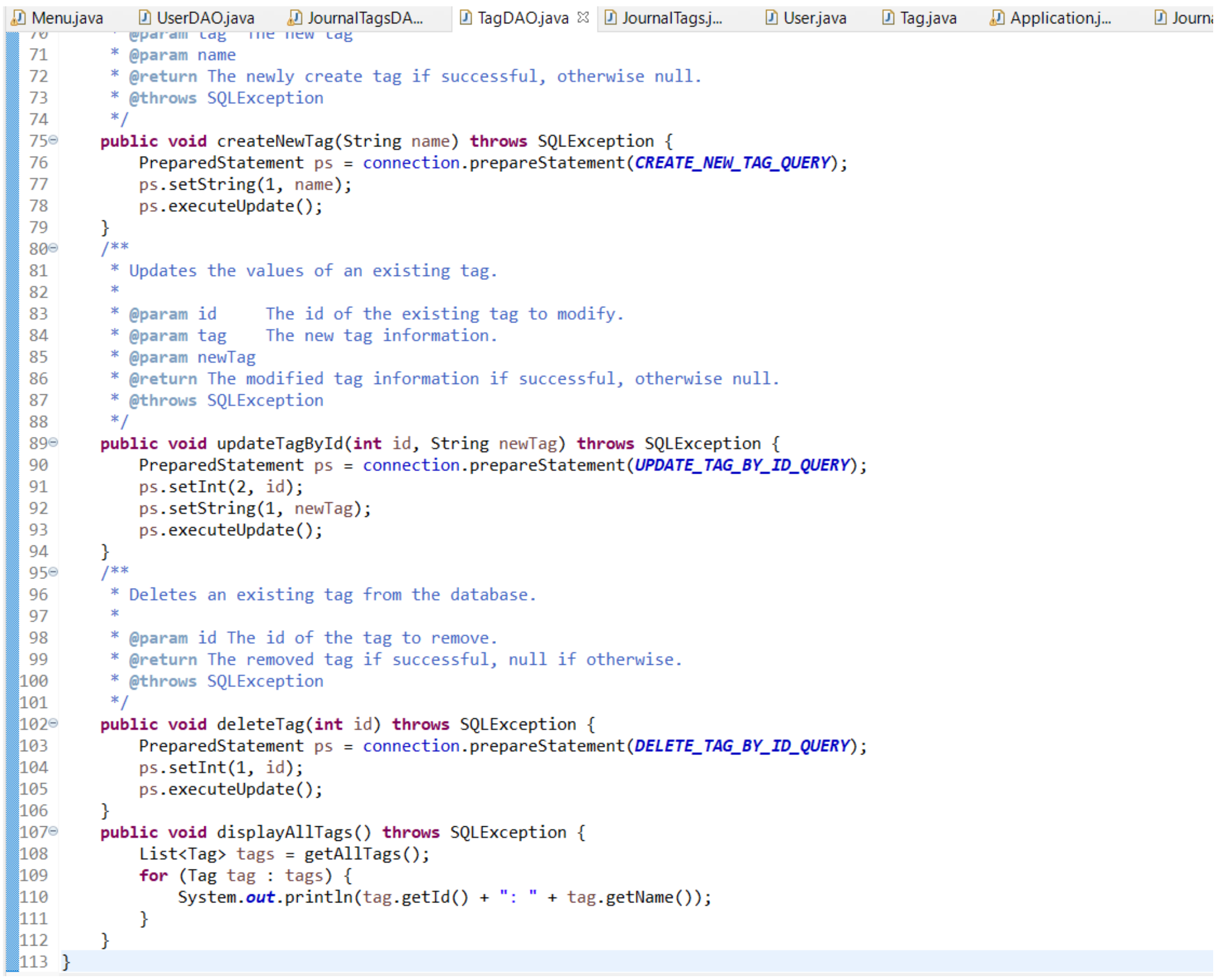
TagDAO



```java
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import entity.Tag;
public class TagDAO {
    private static final String CREATE_NEW_TAG_QUERY = "INSERT INTO tags(name) VALUES(?)";
    private static final String UPDATE_TAG_BY_ID_QUERY = "UPDATE tags SET name = ? WHERE id = ?";
    private static final String DELETE_TAG_BY_ID_QUERY = "DELETE FROM tags WHERE id = ?";
    private Connection connection;
    private final String GET_ALL_TAGS_QUERY = "SELECT * FROM tags";
    private final String GET_TAG_BY_ID_QUERY = "SELECT * FROM tags WHERE id = ?";
    private final String GET_TAG_BY_NAME_QUERY = "SELECT * FROM tags WHERE name = ?";
    public TagDAO() {
        connection = DBConnection.getConnection();
    }
    /**
     * Retrieves a tag based on it's unique id/
     *
     * @param id The unique id of the tag.
     * @return The tag if found, otherwise null.
     * @throws SQLException
     */
    /**
     * Retrieves a tag based on it's name.
     *
     * @param id The unique name of the tag.
     * @return The tag if found, otherwise null.
     * @throws SQLException
     */
    public Tag getTagByName(String name) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_TAG_BY_NAME_QUERY);
        ps.setString(1, name);
        ResultSet rs = ps.executeQuery();
        rs.next();
```

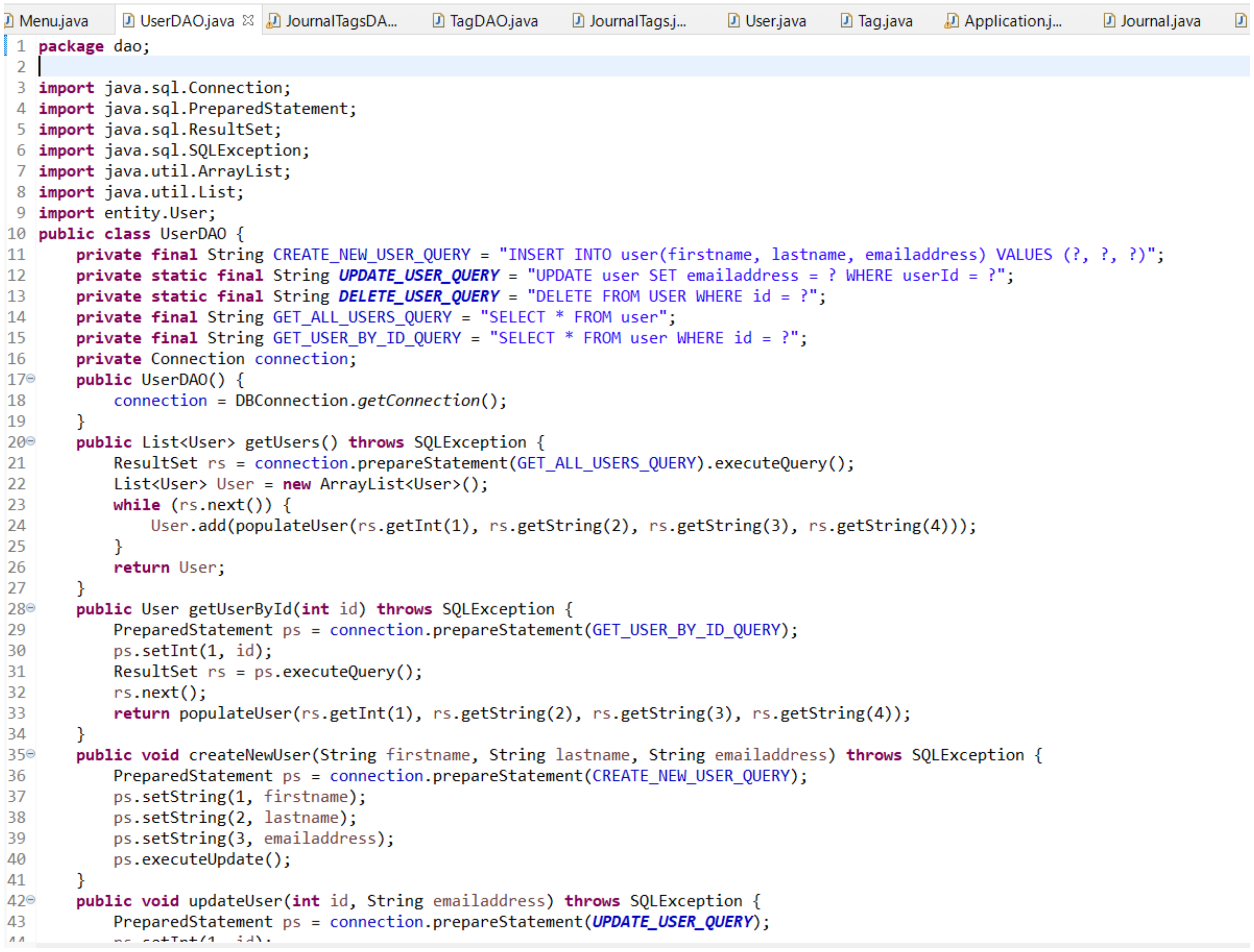

Menu.java | UserDAO.java | JournalTagsDA... | TagDAO.java | JournalTags.java | User.java | Tag.java | Application.ja

```java
        ps.setString(1, name);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateTags(rs.getInt(1), rs.getString(2));
    }
    private Tag populateTags(int id, String name) {
        return new Tag(id, name);
    }
    public Tag getTagById(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_TAG_BY_ID_QUERY);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateTags(rs.getInt(1), rs.getString(2));
    }
    /**
     * Retrieves all of the available tags.
     *
     * @return The enumeration of tag, if no tags are present then an empty list is
     *         returned.
     * @throws SQLException
     */
    public List<Tag> getAllTags() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_ALL_TAGS_QUERY).executeQuery();
        List<Tag> tags = new ArrayList<Tag>();
        while (rs.next()) {
            tags.add(populateTags(rs.getInt(1), rs.getString(2)));
        }
        return tags;
    }
    /**
     * Creates a new tag entry.
     *
     * @param tag  The new tag
     * @param name
     * @return The newly create tag if successful, otherwise null.
     * @throws SQLException
     */
    public void createNewTag(String name) throws SQLException {
```

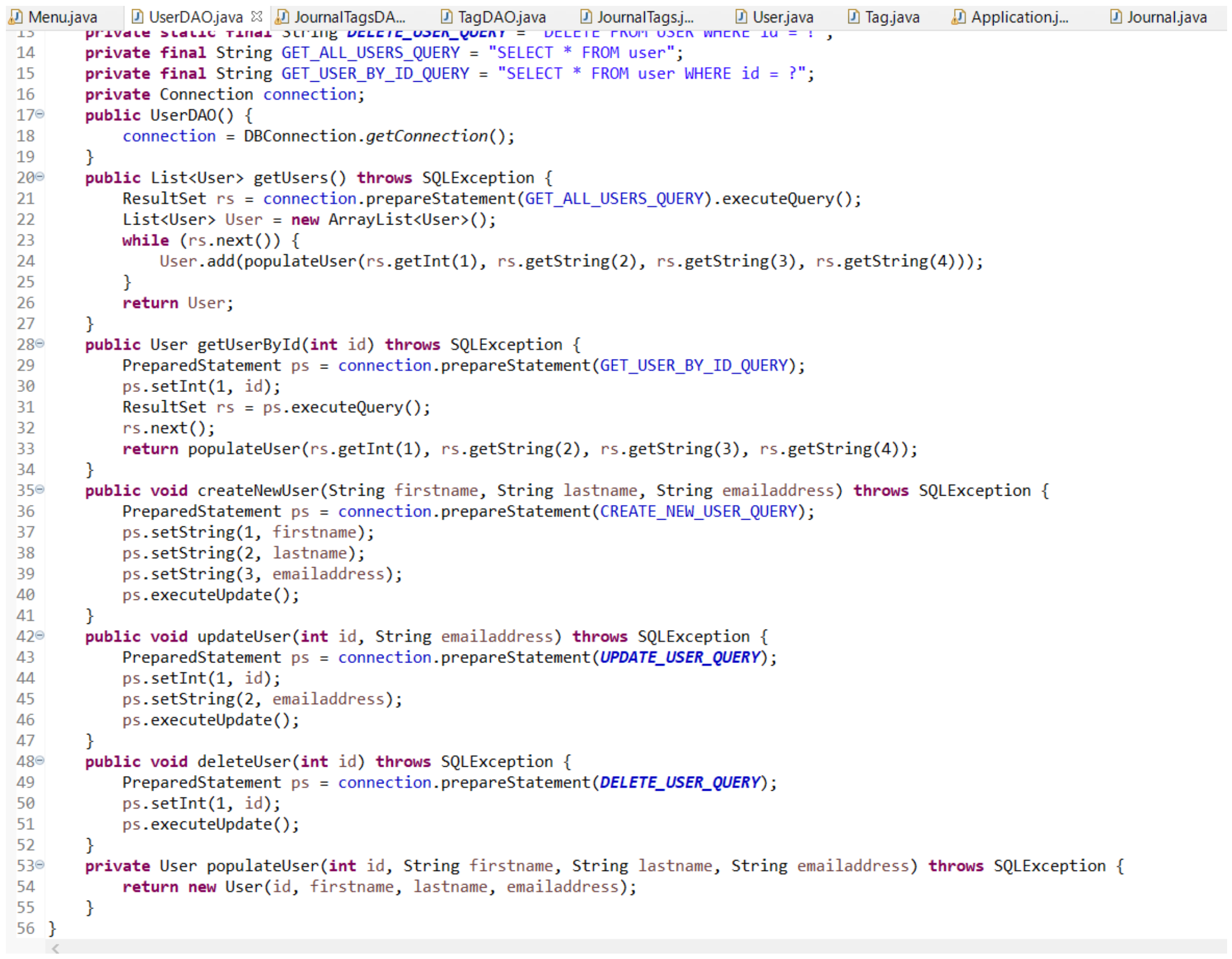


```java
     * @param tag   The new tag
     * @param name
     * @return The newly create tag if successful, otherwise null.
     * @throws SQLException
     */
    public void createNewTag(String name) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_TAG_QUERY);
        ps.setString(1, name);
        ps.executeUpdate();
    }
    /**
     * Updates the values of an existing tag.
     *
     * @param id     The id of the existing tag to modify.
     * @param tag    The new tag information.
     * @param newTag
     * @return The modified tag information if successful, otherwise null.
     * @throws SQLException
     */
    public void updateTagById(int id, String newTag) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_TAG_BY_ID_QUERY);
        ps.setInt(2, id);
        ps.setString(1, newTag);
        ps.executeUpdate();
    }
    /**
     * Deletes an existing tag from the database.
     *
     * @param id The id of the tag to remove.
     * @return The removed tag if successful, null if otherwise.
     * @throws SQLException
     */
    public void deleteTag(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_TAG_BY_ID_QUERY);
        ps.setInt(1, id);
        ps.executeUpdate();
    }
    public void displayAllTags() throws SQLException {
        List<Tag> tags = getAllTags();
        for (Tag tag : tags) {
            System.out.println(tag.getId() + ": " + tag.getName());
        }
    }
}
```

UserDAO

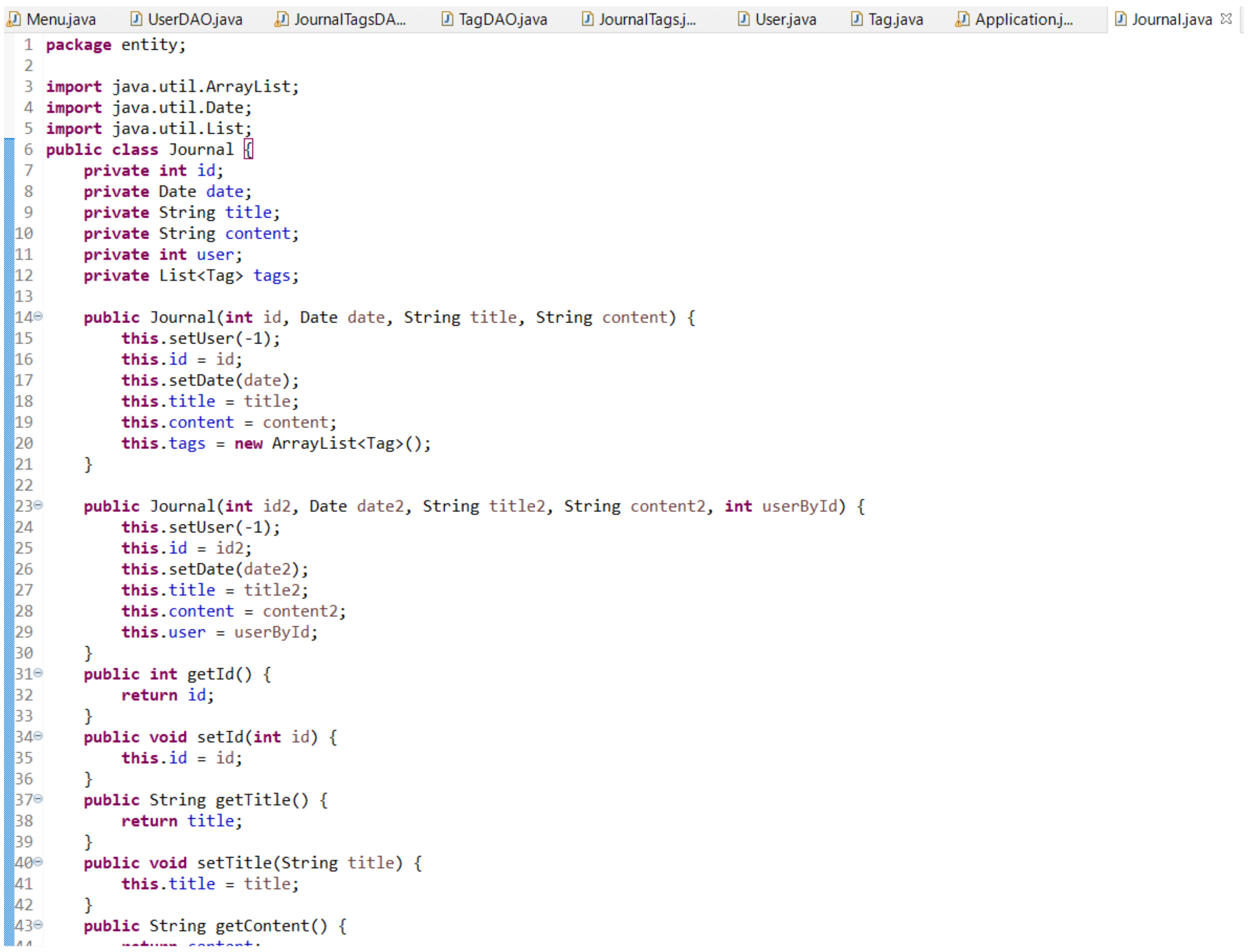


```java
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import entity.User;
public class UserDAO {
    private final String CREATE_NEW_USER_QUERY = "INSERT INTO user(firstname, lastname, emailaddress) VALUES (?, ?, ?)";
    private static final String UPDATE_USER_QUERY = "UPDATE user SET emailaddress = ? WHERE userId = ?";
    private static final String DELETE_USER_QUERY = "DELETE FROM USER WHERE id = ?";
    private final String GET_ALL_USERS_QUERY = "SELECT * FROM user";
    private final String GET_USER_BY_ID_QUERY = "SELECT * FROM user WHERE id = ?";
    private Connection connection;
    public UserDAO() {
        connection = DBConnection.getConnection();
    }
    public List<User> getUsers() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_ALL_USERS_QUERY).executeQuery();
        List<User> User = new ArrayList<User>();
        while (rs.next()) {
            User.add(populateUser(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4)));
        }
        return User;
    }
    public User getUserById(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_USER_BY_ID_QUERY);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateUser(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4));
    }
    public void createNewUser(String firstname, String lastname, String emailaddress) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_USER_QUERY);
        ps.setString(1, firstname);
        ps.setString(2, lastname);
        ps.setString(3, emailaddress);
        ps.executeUpdate();
    }
    public void updateUser(int id, String emailaddress) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_USER_QUERY);
```

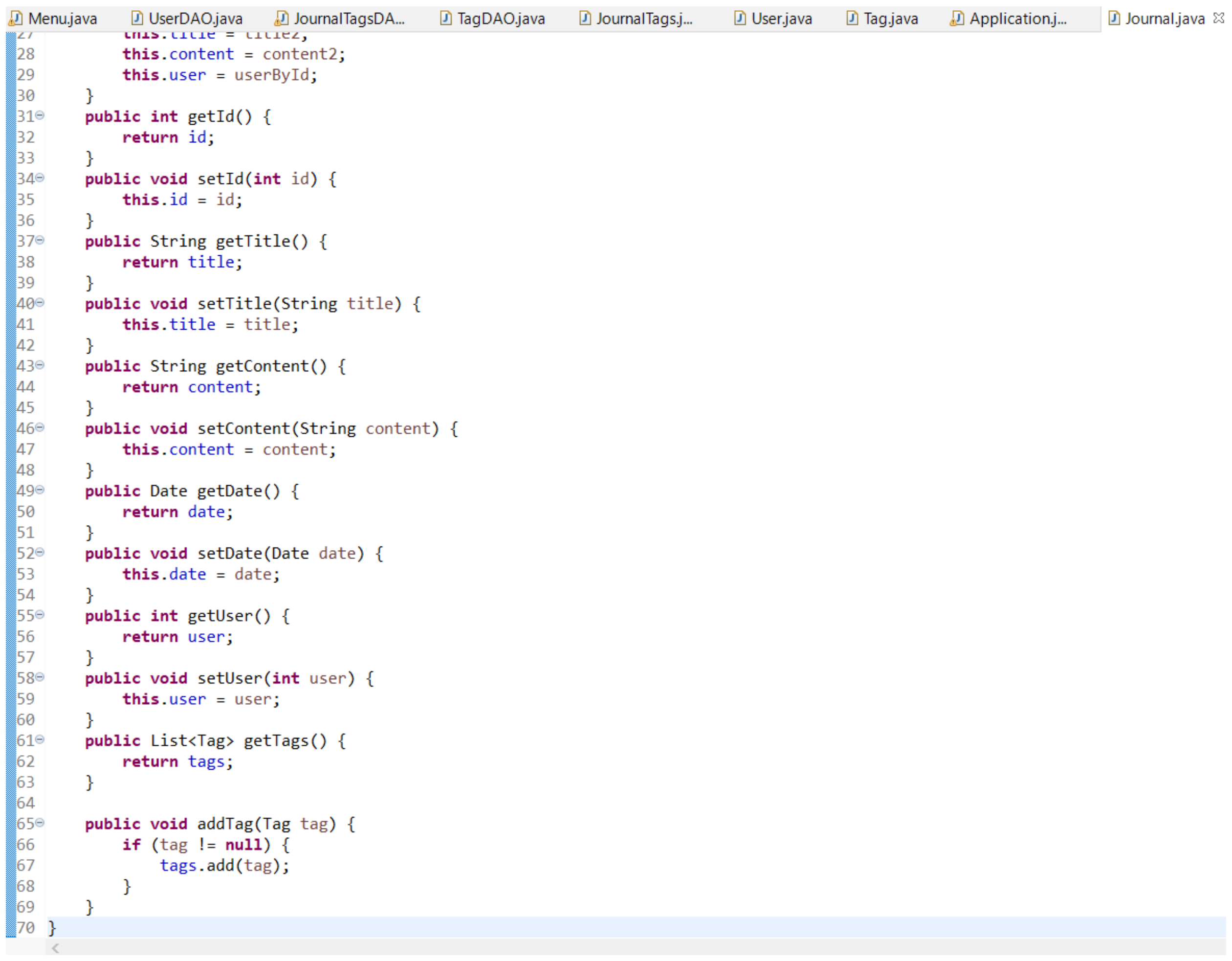


```java
    private static final String DELETE_USER_QUERY = "DELETE FROM USER WHERE id = ?";
    private final String GET_ALL_USERS_QUERY = "SELECT * FROM user";
    private final String GET_USER_BY_ID_QUERY = "SELECT * FROM user WHERE id = ?";
    private Connection connection;
    public UserDAO() {
        connection = DBConnection.getConnection();
    }
    public List<User> getUsers() throws SQLException {
        ResultSet rs = connection.prepareStatement(GET_ALL_USERS_QUERY).executeQuery();
        List<User> User = new ArrayList<User>();
        while (rs.next()) {
            User.add(populateUser(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4)));
        }
        return User;
    }
    public User getUserById(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(GET_USER_BY_ID_QUERY);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        rs.next();
        return populateUser(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4));
    }
    public void createNewUser(String firstname, String lastname, String emailaddress) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(CREATE_NEW_USER_QUERY);
        ps.setString(1, firstname);
        ps.setString(2, lastname);
        ps.setString(3, emailaddress);
        ps.executeUpdate();
    }
    public void updateUser(int id, String emailaddress) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(UPDATE_USER_QUERY);
        ps.setInt(1, id);
        ps.setString(2, emailaddress);
        ps.executeUpdate();
    }
    public void deleteUser(int id) throws SQLException {
        PreparedStatement ps = connection.prepareStatement(DELETE_USER_QUERY);
        ps.setInt(1, id);
        ps.executeUpdate();
    }
    private User populateUser(int id, String firstname, String lastname, String emailaddress) throws SQLException {
        return new User(id, firstname, lastname, emailaddress);
    }
}
```

Journal

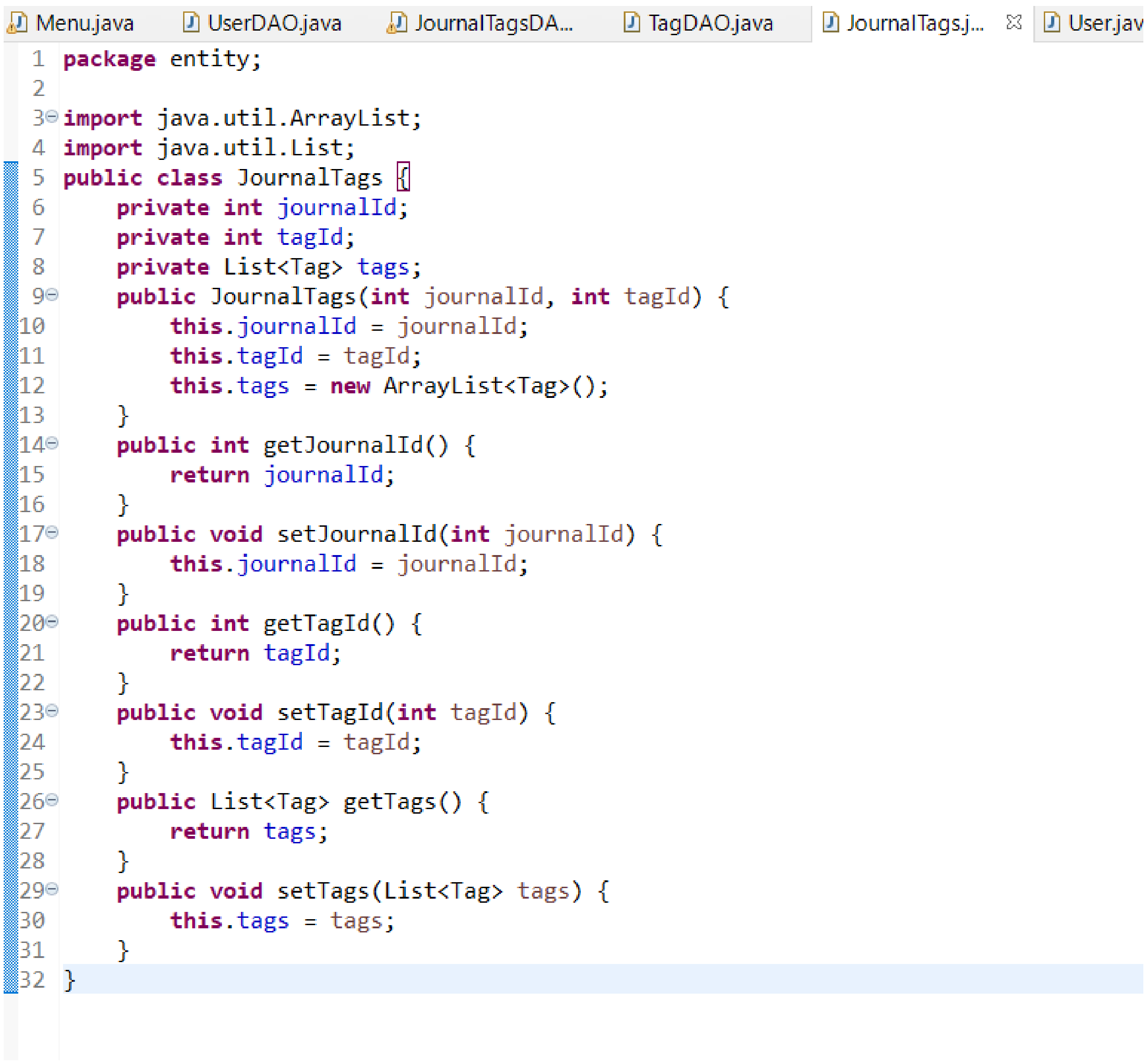


```java
package entity;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
public class Journal {
    private int id;
    private Date date;
    private String title;
    private String content;
    private int user;
    private List<Tag> tags;

    public Journal(int id, Date date, String title, String content) {
        this.setUser(-1);
        this.id = id;
        this.setDate(date);
        this.title = title;
        this.content = content;
        this.tags = new ArrayList<Tag>();
    }

    public Journal(int id2, Date date2, String title2, String content2, int userById) {
        this.setUser(-1);
        this.id = id2;
        this.setDate(date2);
        this.title = title2;
        this.content = content2;
        this.user = userById;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
```



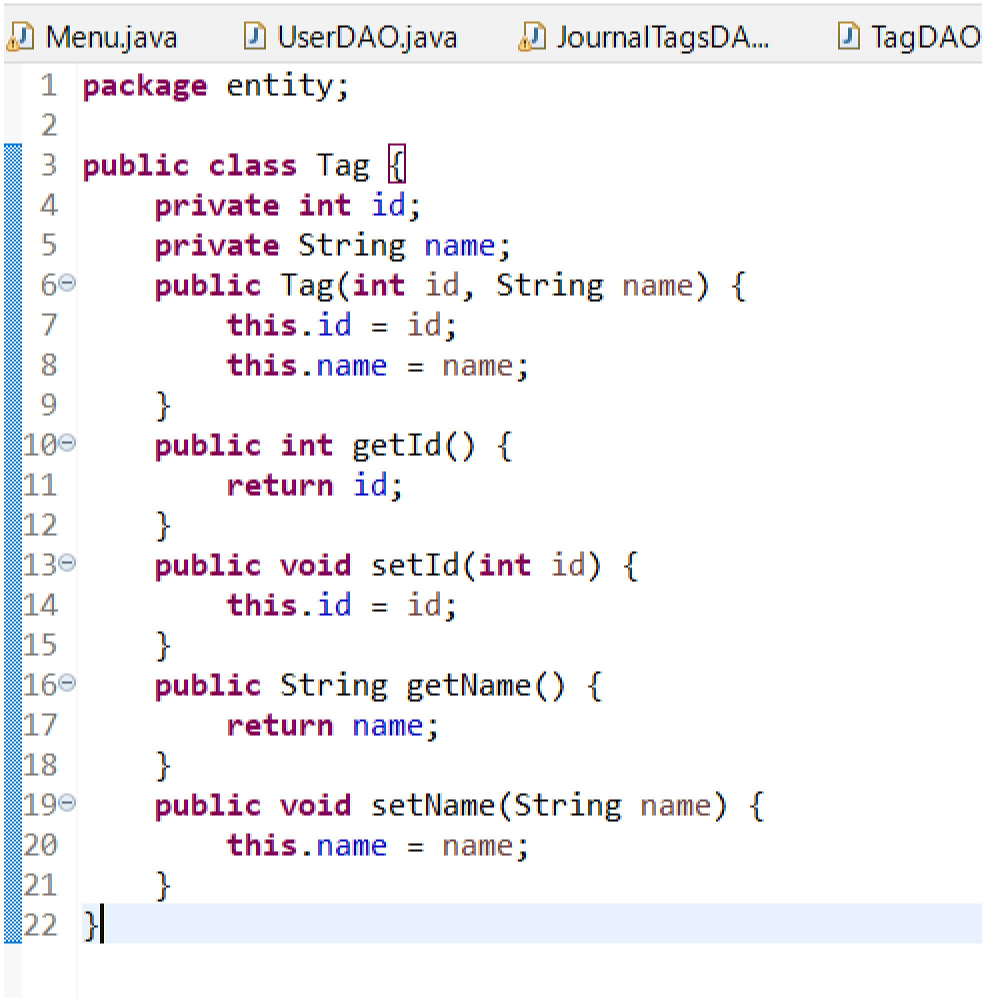
```
        this.title = title2;
        this.content = content2;
        this.user = userById;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
    public int getUser() {
        return user;
    }
    public void setUser(int user) {
        this.user = user;
    }
    public List<Tag> getTags() {
        return tags;
    }

    public void addTag(Tag tag) {
        if (tag != null) {
            tags.add(tag);
        }
    }
}
```

JournalTags


Menu.java   UserDAO.java   JournalTagsDA...   TagDAO.java   JournalTags.j...   User.jav

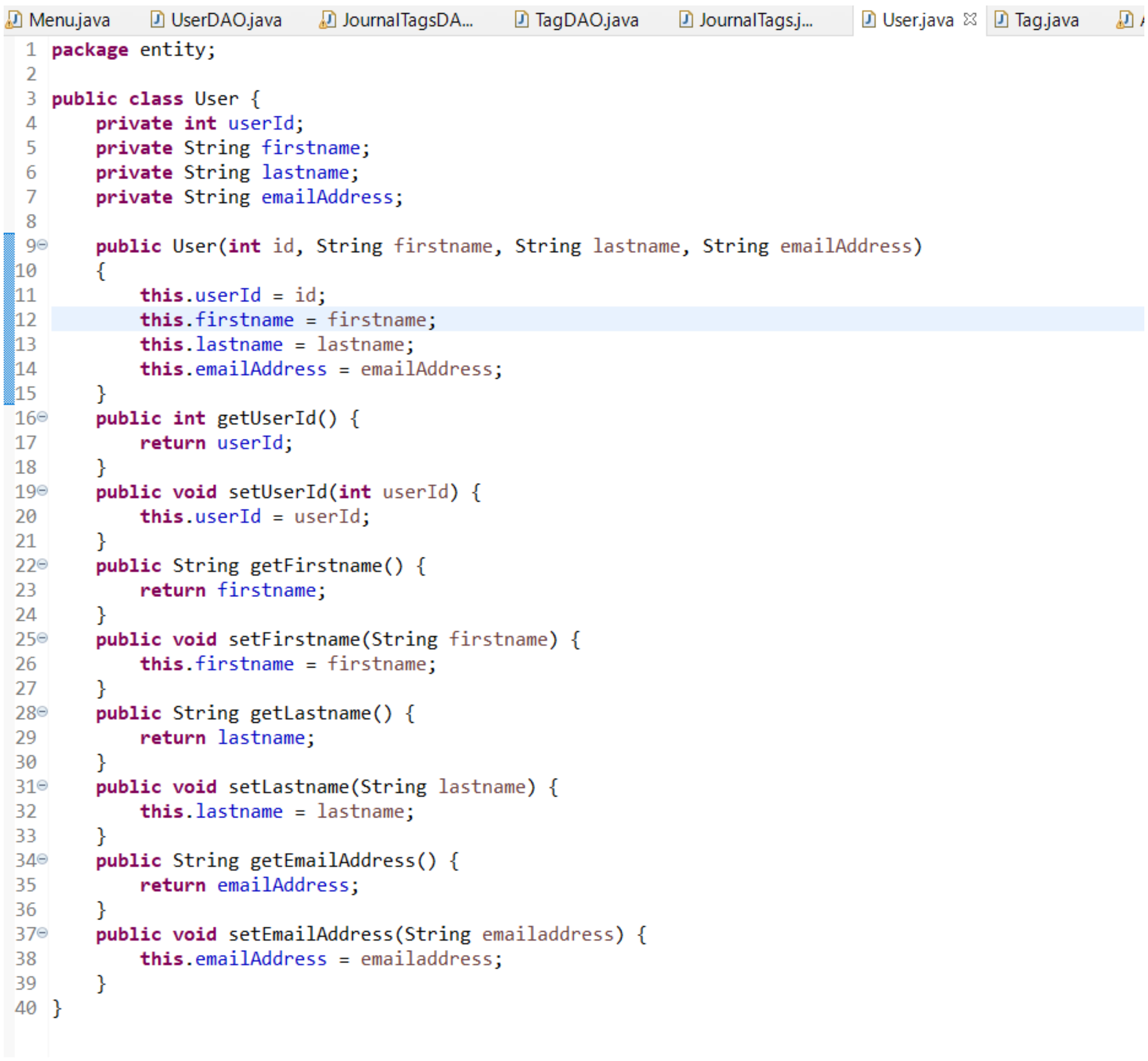
```java
package entity;

import java.util.ArrayList;
import java.util.List;
public class JournalTags {
    private int journalId;
    private int tagId;
    private List<Tag> tags;
    public JournalTags(int journalId, int tagId) {
        this.journalId = journalId;
        this.tagId = tagId;
        this.tags = new ArrayList<Tag>();
    }
    public int getJournalId() {
        return journalId;
    }
    public void setJournalId(int journalId) {
        this.journalId = journalId;
    }
    public int getTagId() {
        return tagId;
    }
    public void setTagId(int tagId) {
        this.tagId = tagId;
    }
    public List<Tag> getTags() {
        return tags;
    }
    public void setTags(List<Tag> tags) {
        this.tags = tags;
    }
}
```

Tag



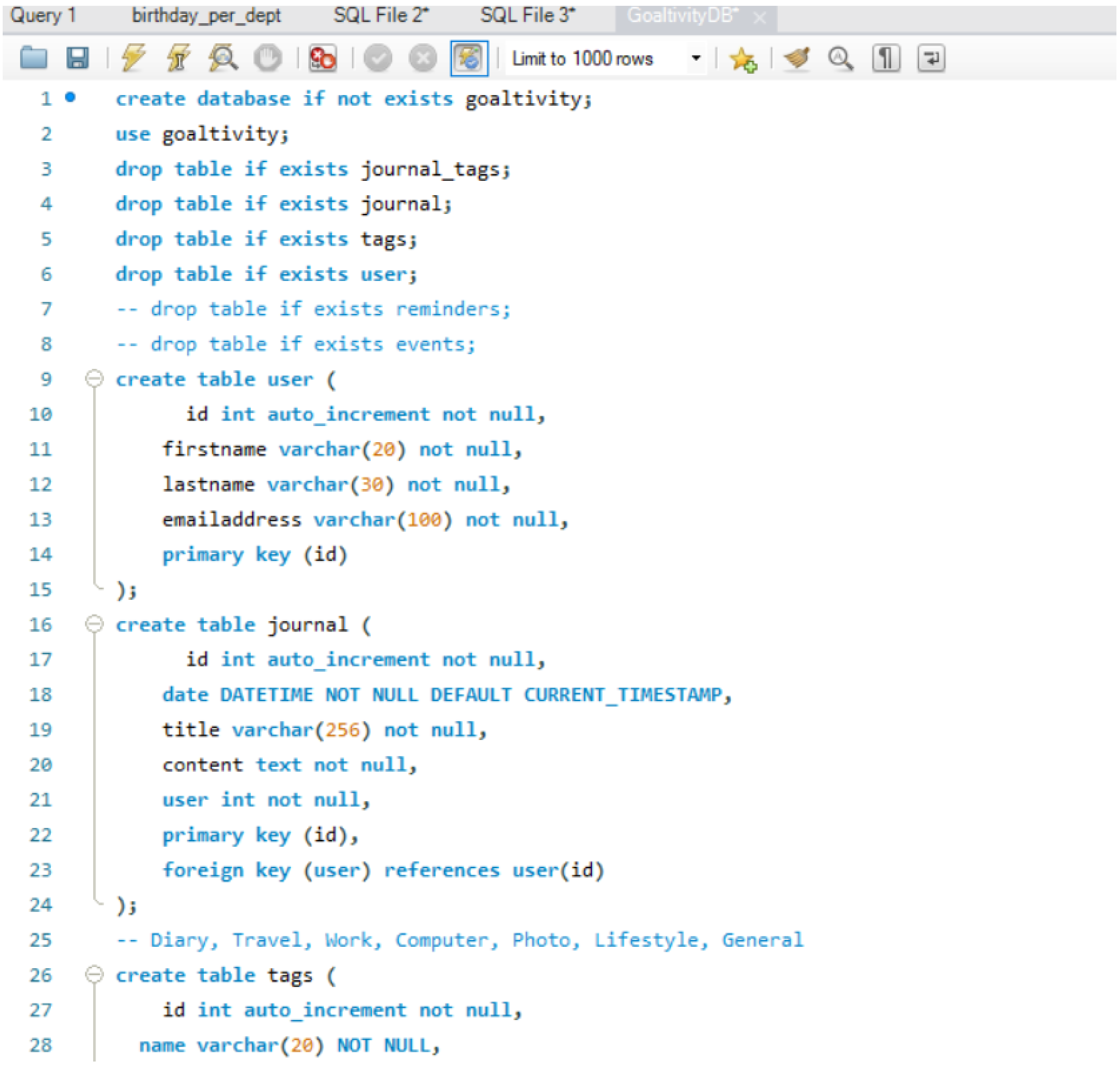
```java
package entity;

public class Tag {
    private int id;
    private String name;
    public Tag(int id, String name) {
        this.id = id;
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

User

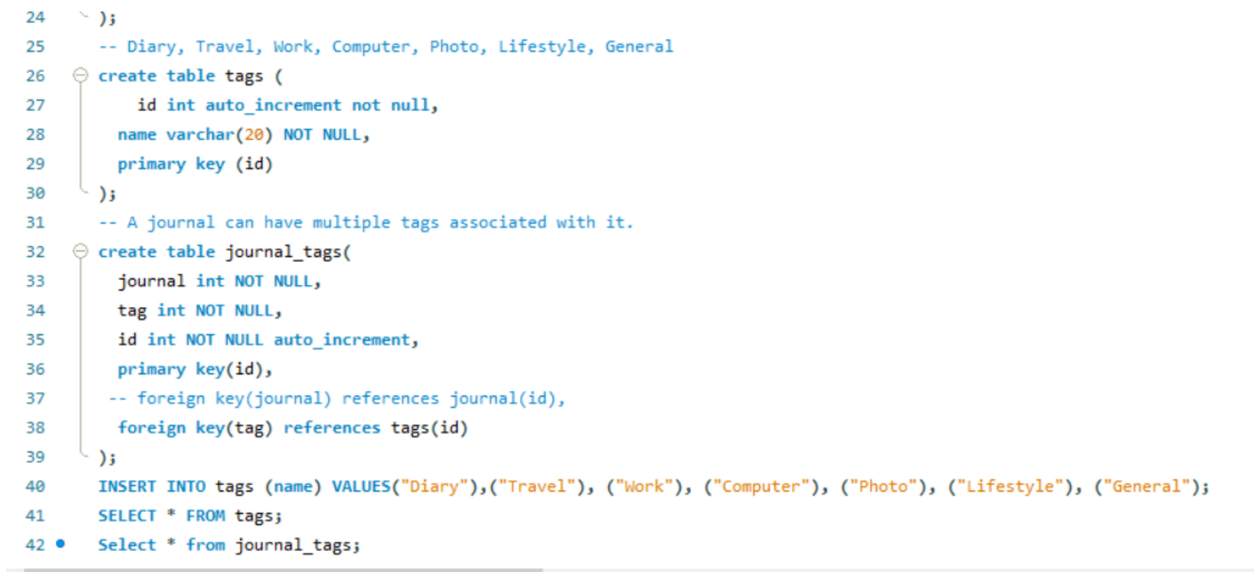


```java
package entity;

public class User {
    private int userId;
    private String firstname;
    private String lastname;
    private String emailAddress;

    public User(int id, String firstname, String lastname, String emailAddress)
    {
        this.userId = id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.emailAddress = emailAddress;
    }
    public int getUserId() {
        return userId;
    }
    public void setUserId(int userId) {
        this.userId = userId;
    }
    public String getFirstname() {
        return firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    public String getLastname() {
        return lastname;
    }
    public void setLastname(String lastname) {
        this.lastname = lastname;
    }
    public String getEmailAddress() {
        return emailAddress;
    }
    public void setEmailAddress(String emailaddress) {
        this.emailAddress = emailaddress;
    }
}
```

SQL Database

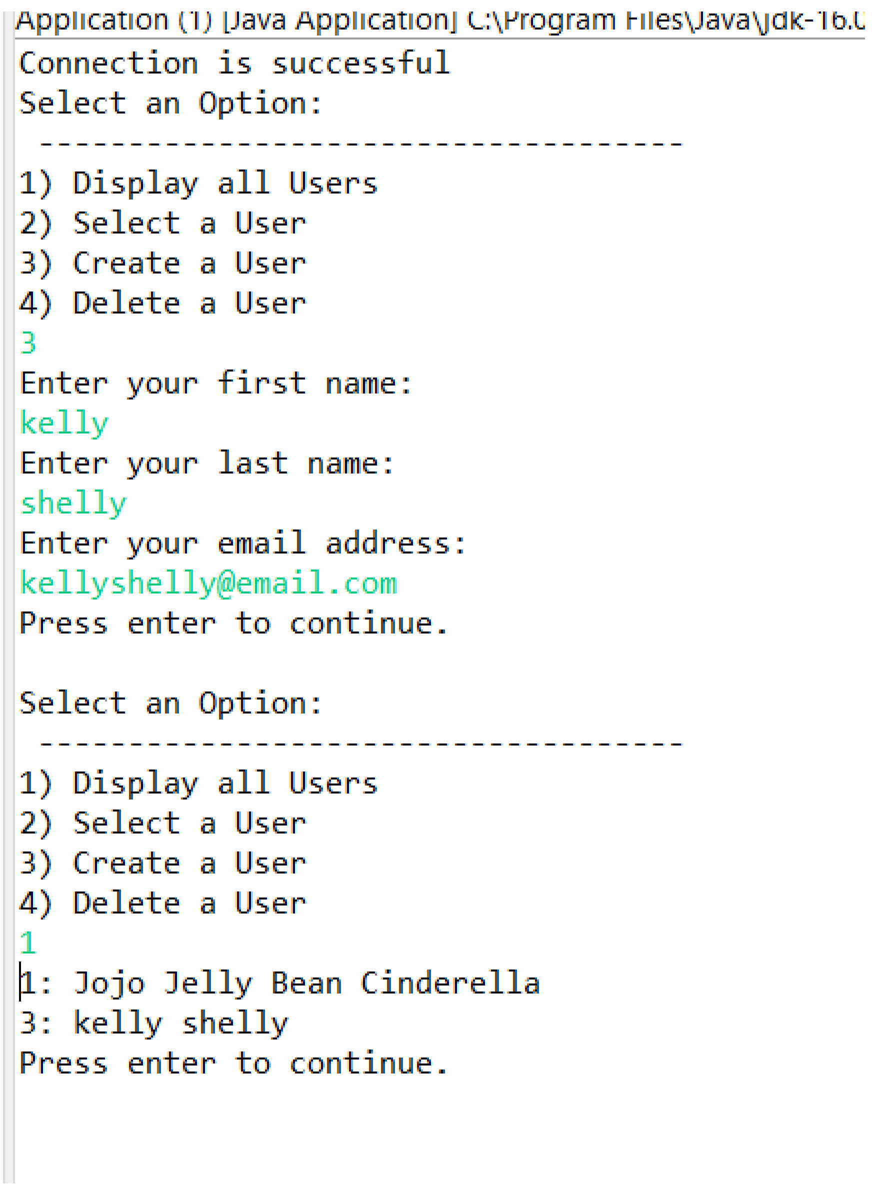


```sql
create database if not exists goaltivity;
use goaltivity;
drop table if exists journal_tags;
drop table if exists journal;
drop table if exists tags;
drop table if exists user;
-- drop table if exists reminders;
-- drop table if exists events;
create table user (
      id int auto_increment not null,
    firstname varchar(20) not null,
    lastname varchar(30) not null,
    emailaddress varchar(100) not null,
    primary key (id)
);
create table journal (
      id int auto_increment not null,
    date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    title varchar(256) not null,
    content text not null,
    user int not null,
    primary key (id),
    foreign key (user) references user(id)
);
-- Diary, Travel, Work, Computer, Photo, Lifestyle, General
create table tags (
    id int auto_increment not null,
  name varchar(20) NOT NULL,
```

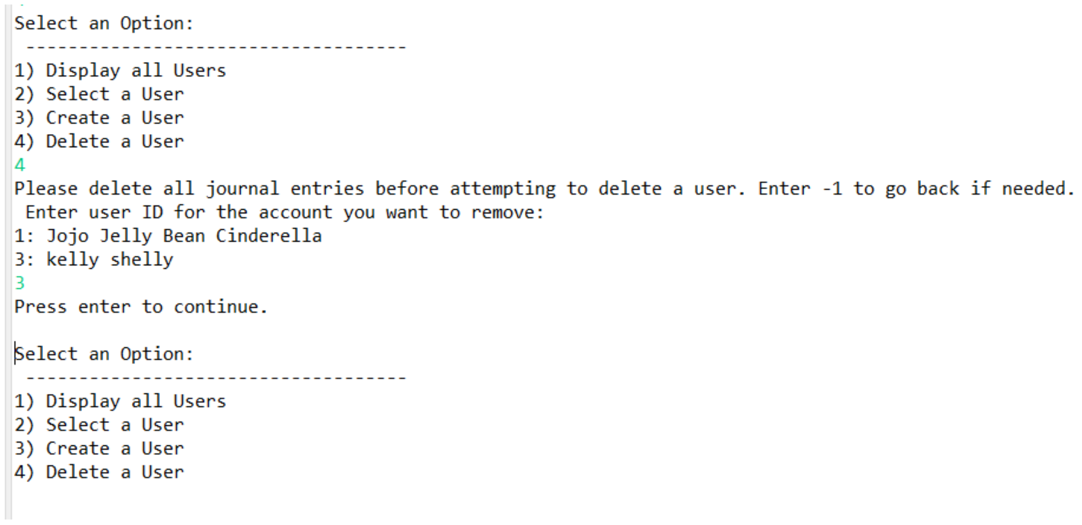


```sql
);
-- Diary, Travel, Work, Computer, Photo, Lifestyle, General
create table tags (
    id int auto_increment not null,
  name varchar(20) NOT NULL,
  primary key (id)
);
-- A journal can have multiple tags associated with it.
create table journal_tags(
  journal int NOT NULL,
  tag int NOT NULL,
  id int NOT NULL auto_increment,
  primary key(id),
 -- foreign key(journal) references journal(id),
  foreign key(tag) references tags(id)
);
INSERT INTO tags (name) VALUES("Diary"),("Travel"), ("Work"), ("Computer"), ("Photo"), ("Lifestyle"), ("General");
SELECT * FROM tags;
Select * from journal_tags;
```

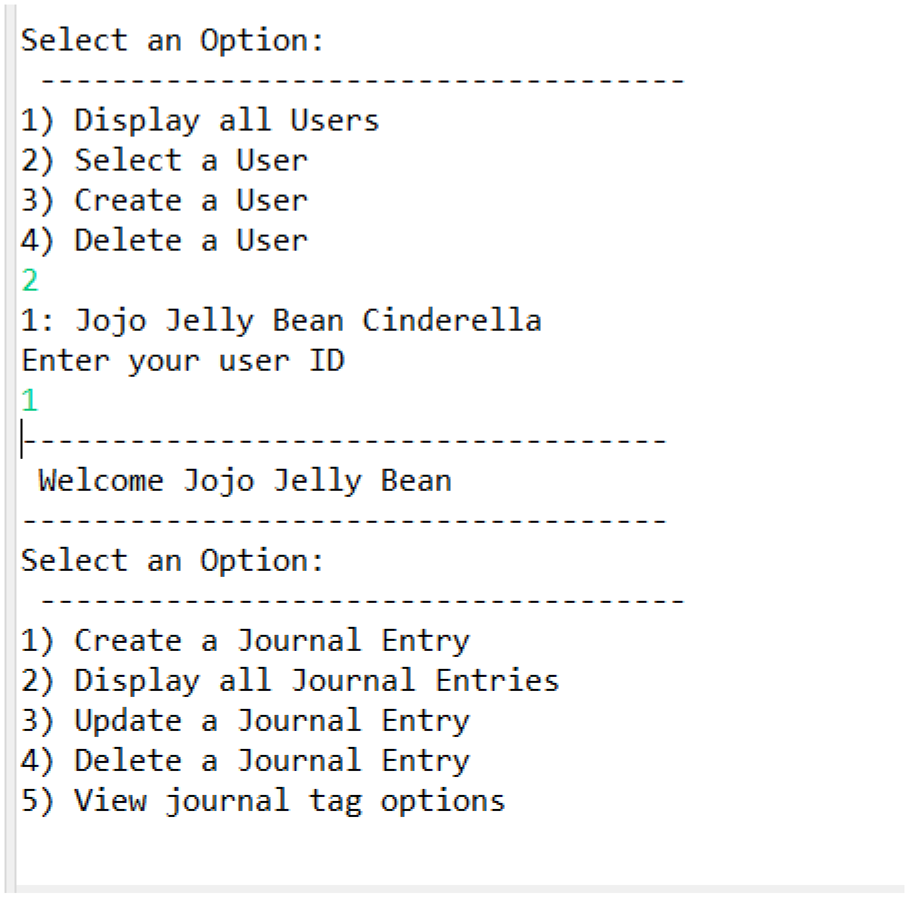
Screenshots of Running Application:



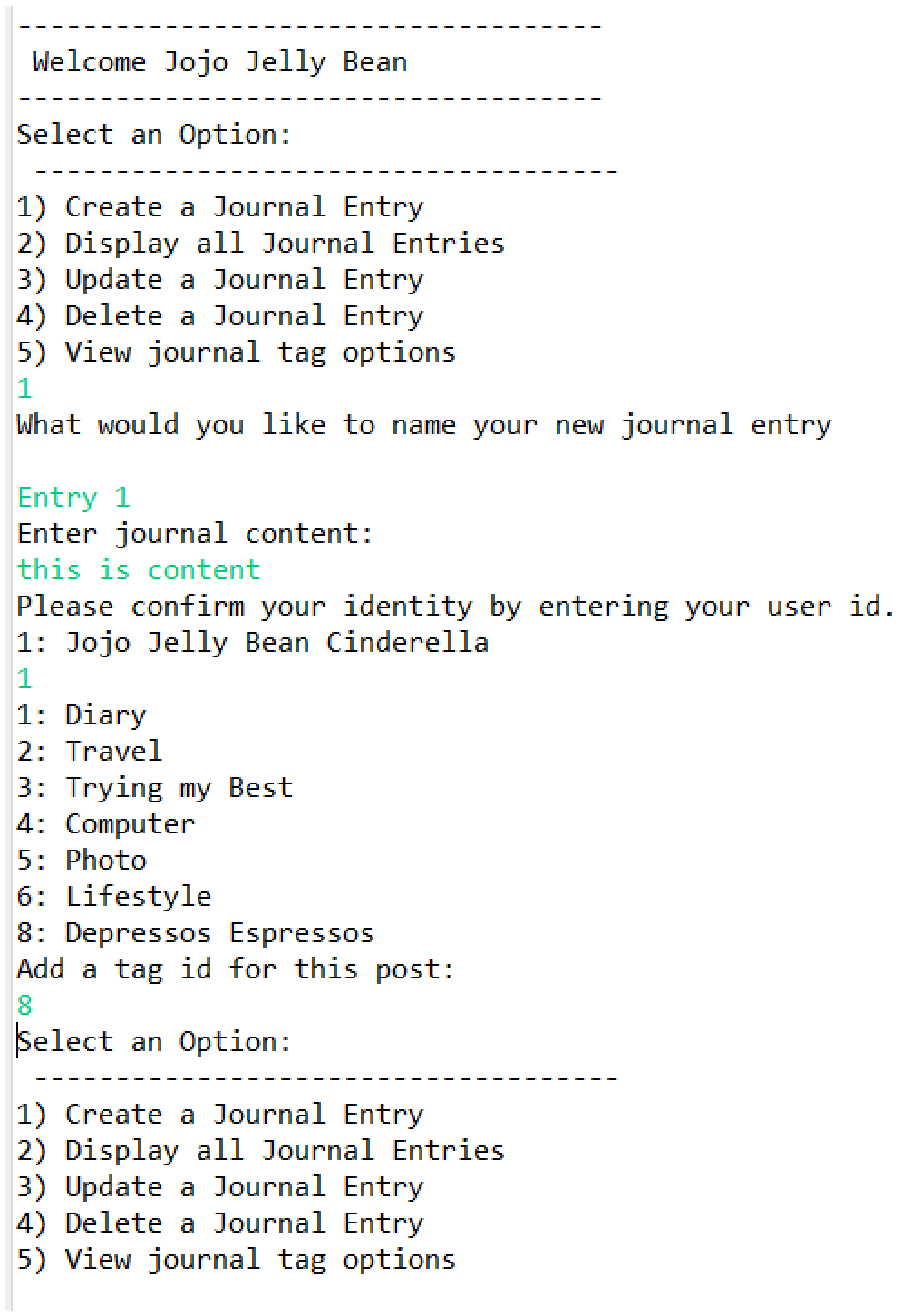
```
Application (1) [Java Application] C:\Program Files\Java\jdk-16.0
Connection is successful
Select an Option:
 ------------------------------------
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
3
Enter your first name:
kelly
Enter your last name:
shelly
Enter your email address:
kellyshelly@email.com
Press enter to continue.

Select an Option:
 ------------------------------------
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
1
1: Jojo Jelly Bean Cinderella
3: kelly shelly
Press enter to continue.
```



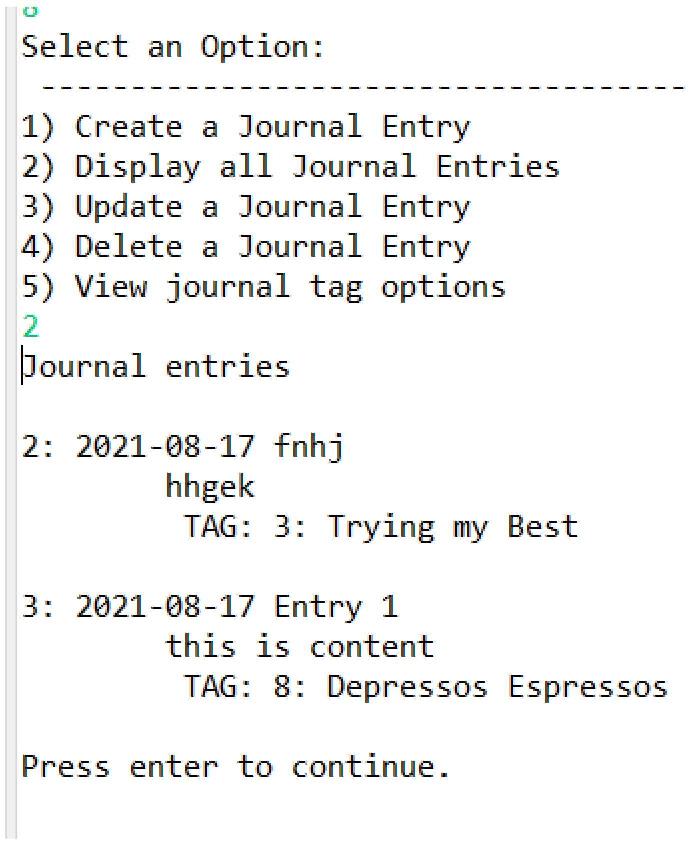
```
Select an Option:
 ------------------------------------
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
4
Please delete all journal entries before attempting to delete a user. Enter -1 to go back if needed.
 Enter user ID for the account you want to remove:
1: Jojo Jelly Bean Cinderella
3: kelly shelly
3
Press enter to continue.

Select an Option:
 ------------------------------------
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
```



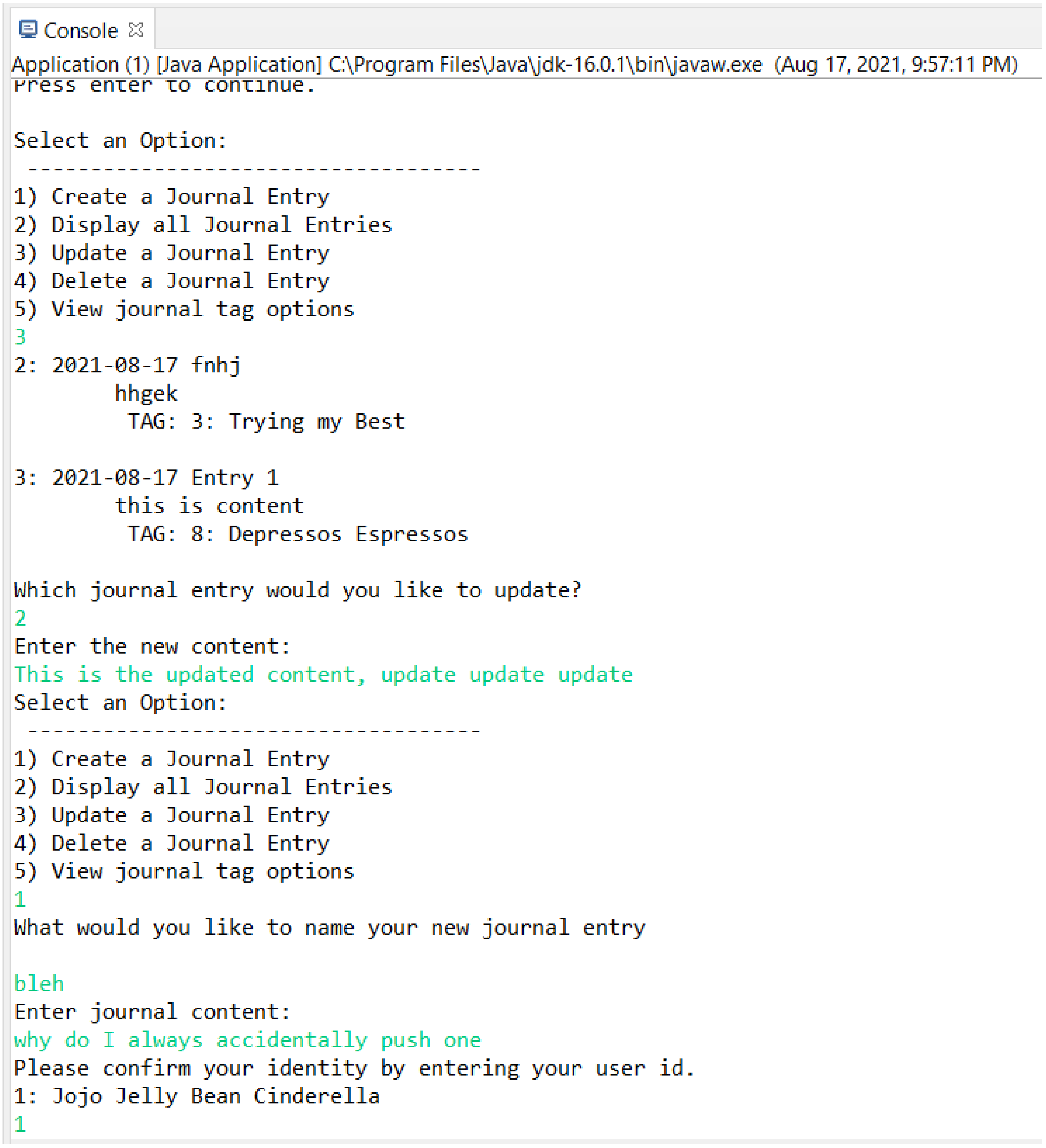
```
Select an Option:
 ------------------------------------
1) Display all Users
2) Select a User
3) Create a User
4) Delete a User
2
1: Jojo Jelly Bean Cinderella
Enter your user ID
1
------------------------------------
 Welcome Jojo Jelly Bean
------------------------------------
Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
```



```
------------------------------------
 Welcome Jojo Jelly Bean
------------------------------------
Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
1
What would you like to name your new journal entry

Entry 1
Enter journal content:
this is content
Please confirm your identity by entering your user id.
1: Jojo Jelly Bean Cinderella
1
1: Diary
2: Travel
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
Add a tag id for this post:
8
Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
```

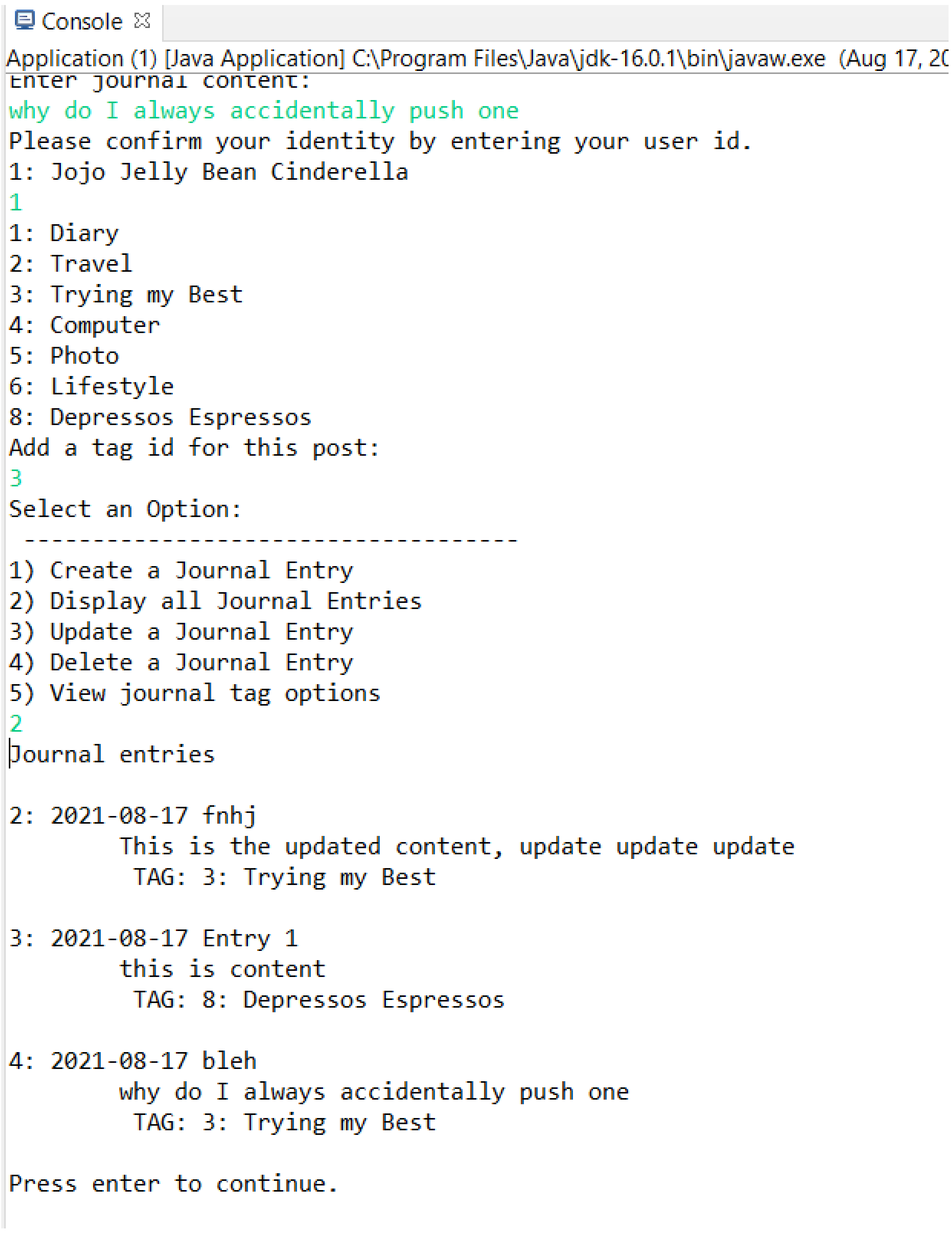


```
8
Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
2
Journal entries

2: 2021-08-17 fnhj
        hhgek
         TAG: 3: Trying my Best

3: 2021-08-17 Entry 1
        this is content
         TAG: 8: Depressos Espressos

Press enter to continue.
```

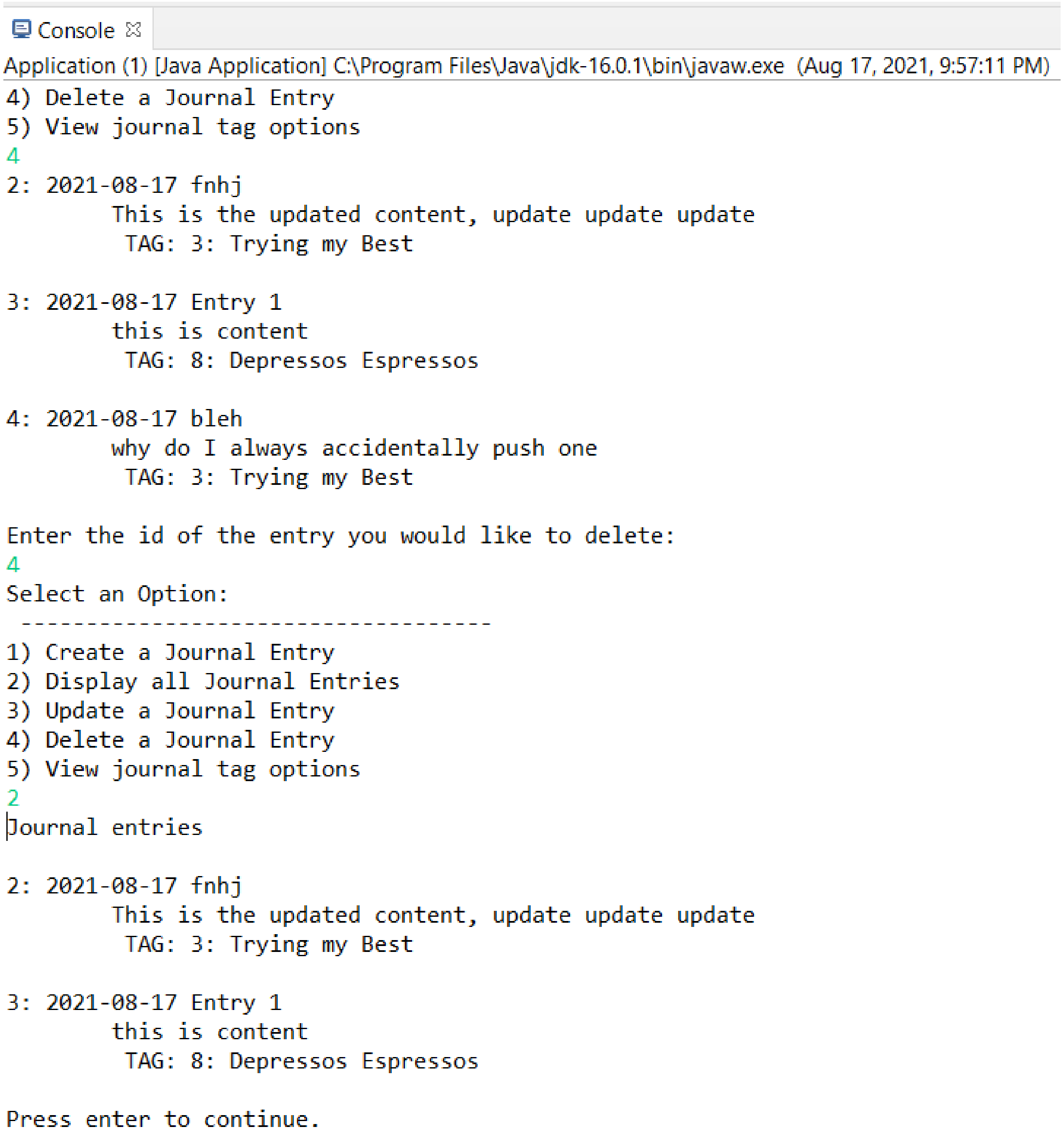


Console

```
Application (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe  (Aug 17, 2021, 9:57:11 PM)
Press enter to continue.

Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
3
2: 2021-08-17 fnhj
        hhgek
         TAG: 3: Trying my Best

3: 2021-08-17 Entry 1
        this is content
         TAG: 8: Depressos Espressos

Which journal entry would you like to update?
2
Enter the new content:
This is the updated content, update update update
Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
1
What would you like to name your new journal entry

bleh
Enter journal content:
why do I always accidentally push one
Please confirm your identity by entering your user id.
1: Jojo Jelly Bean Cinderella
1
```



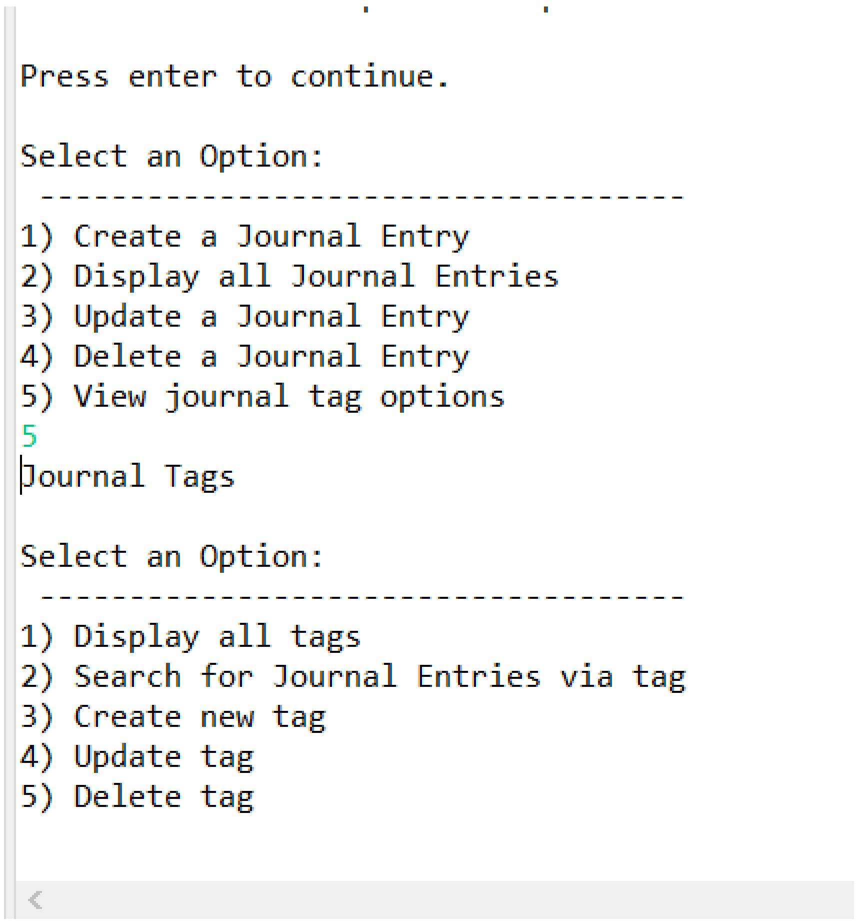
```
Application (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe  (Aug 17, 20
Enter journal content:
why do I always accidentally push one
Please confirm your identity by entering your user id.
1: Jojo Jelly Bean Cinderella
1
1: Diary
2: Travel
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
Add a tag id for this post:
3
Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
2
Journal entries

2: 2021-08-17 fnhj
        This is the updated content, update update update
         TAG: 3: Trying my Best

3: 2021-08-17 Entry 1
        this is content
         TAG: 8: Depressos Espressos

4: 2021-08-17 bleh
        why do I always accidentally push one
         TAG: 3: Trying my Best

Press enter to continue.
```



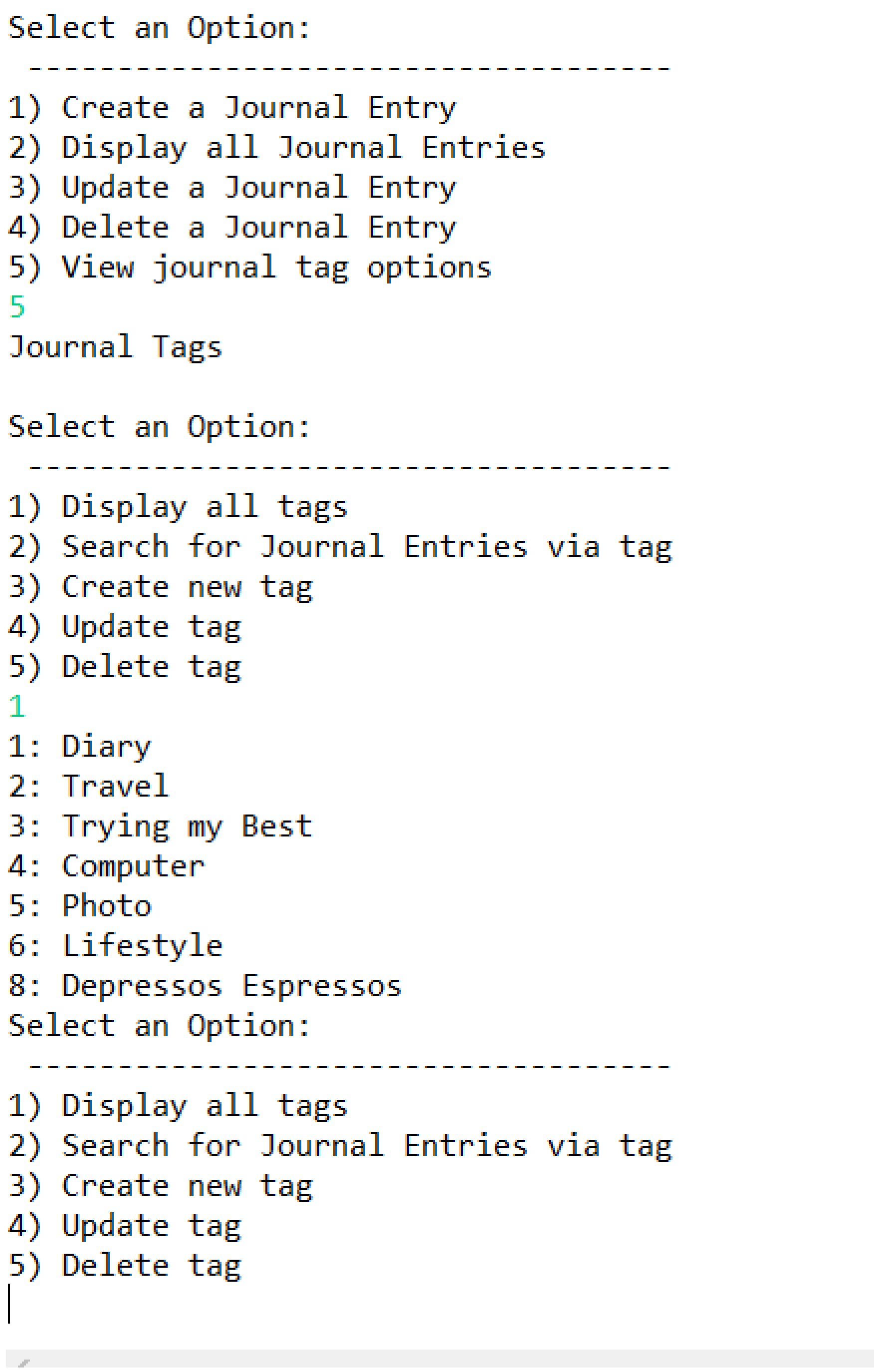
Console ☒

Application (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (Aug 17, 2021, 9:57:11 PM)

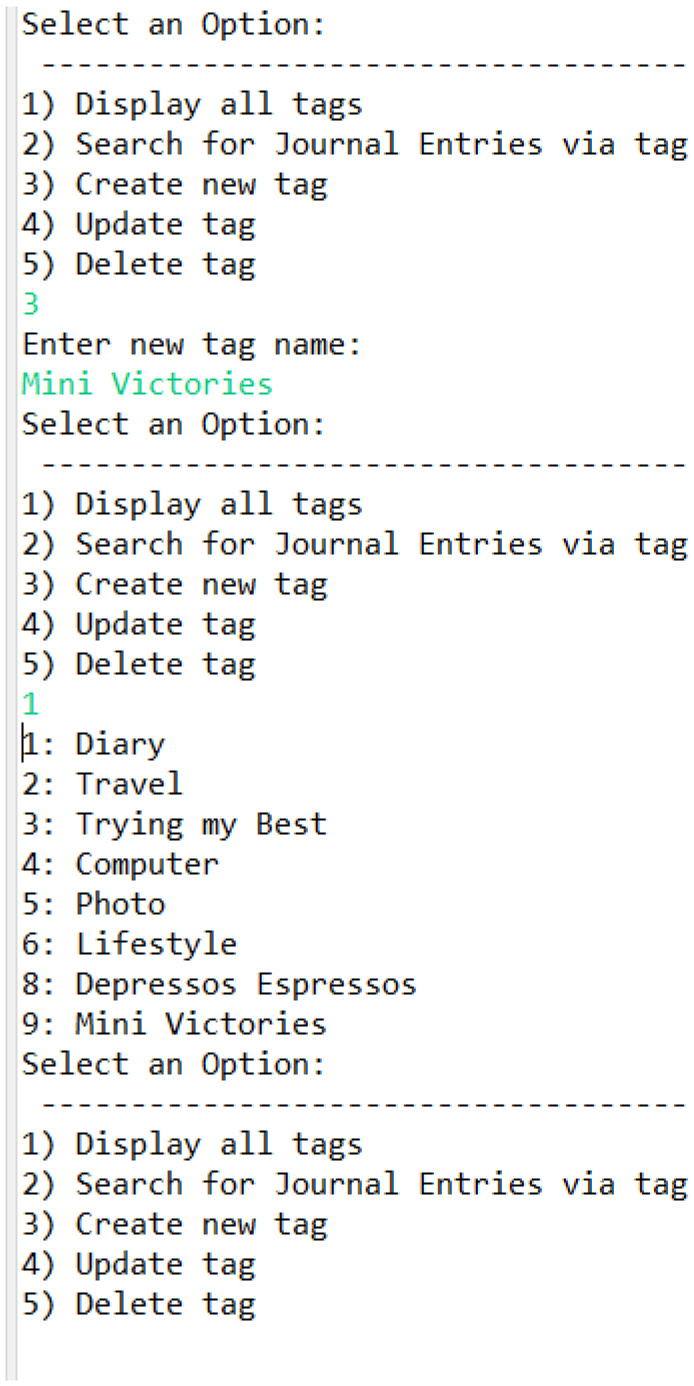
```
4) Delete a Journal Entry
5) View journal tag options
4
2: 2021-08-17 fnhj
        This is the updated content, update update update
         TAG: 3: Trying my Best

3: 2021-08-17 Entry 1
        this is content
         TAG: 8: Depressos Espressos

4: 2021-08-17 bleh
        why do I always accidentally push one
         TAG: 3: Trying my Best

Enter the id of the entry you would like to delete:
4
Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
2
Journal entries

2: 2021-08-17 fnhj
        This is the updated content, update update update
         TAG: 3: Trying my Best

3: 2021-08-17 Entry 1
        this is content
         TAG: 8: Depressos Espressos

Press enter to continue.
```



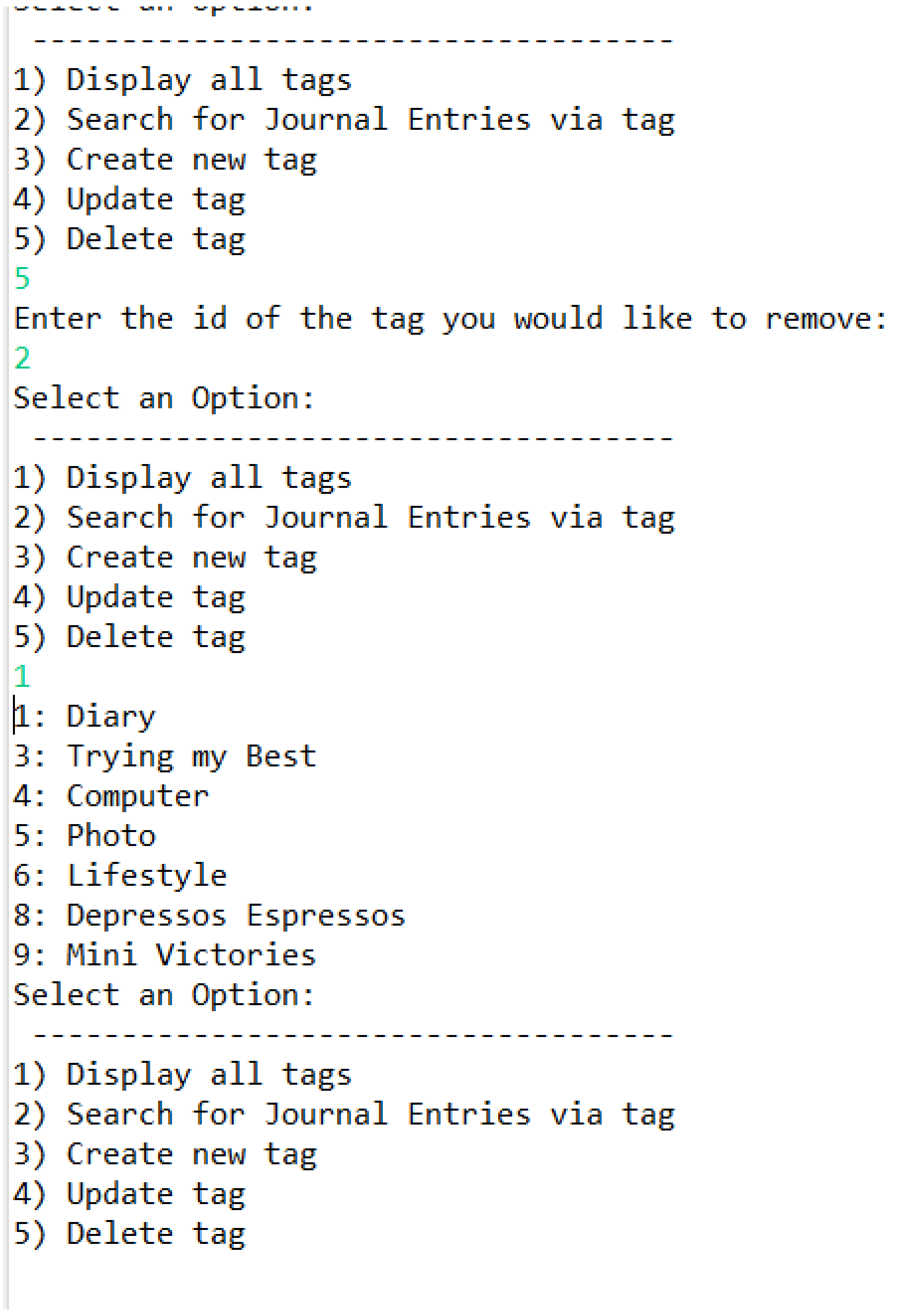
```
Press enter to continue.

Select an Option:
 ------------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
5
Journal Tags

Select an Option:
 ------------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
```



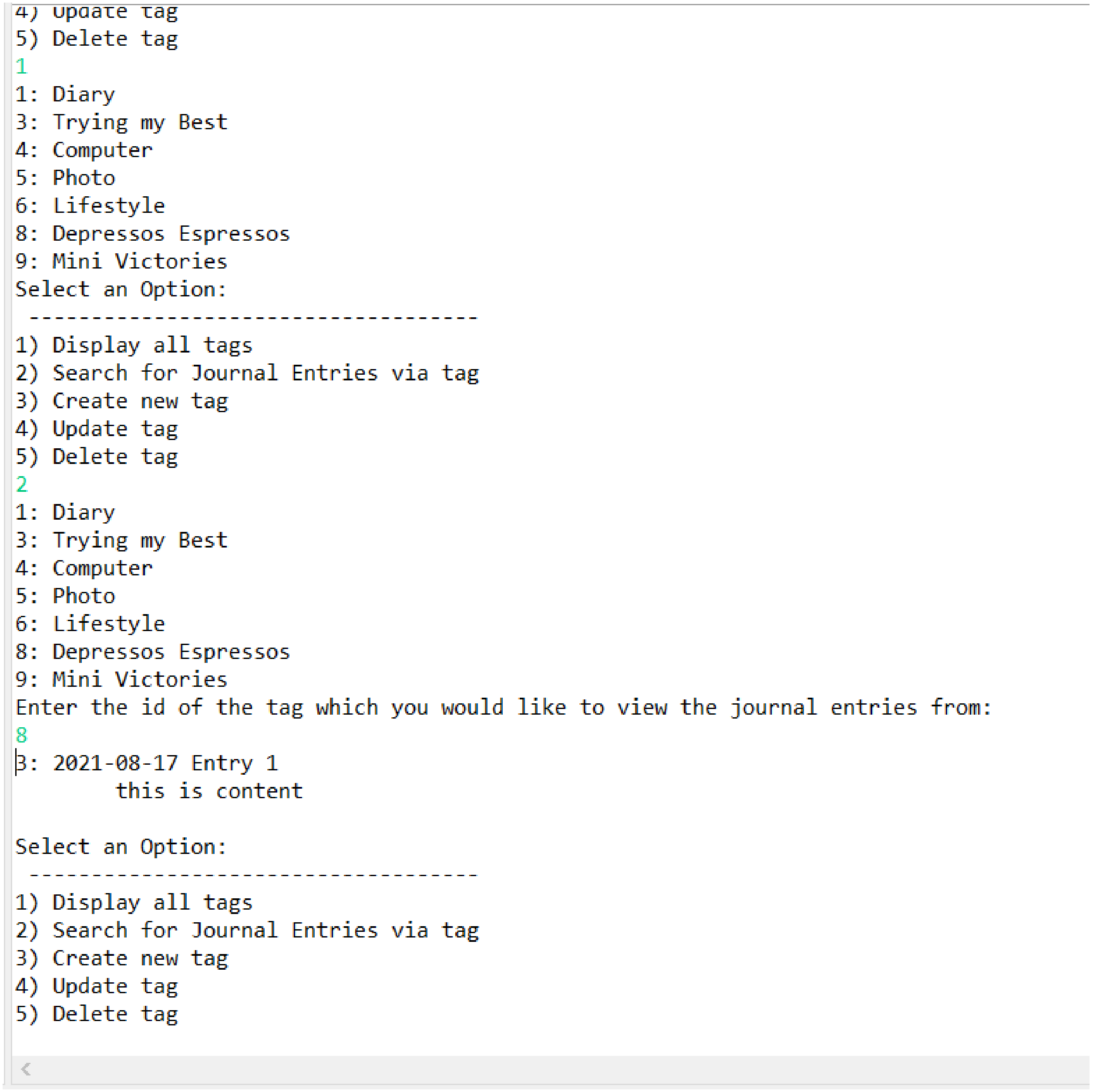
```
Select an Option:
 -----------------------------------
1) Create a Journal Entry
2) Display all Journal Entries
3) Update a Journal Entry
4) Delete a Journal Entry
5) View journal tag options
5
Journal Tags

Select an Option:
 -----------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
1
1: Diary
2: Travel
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
Select an Option:
 -----------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
```



```
Select an Option:
 ------------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
3
Enter new tag name:
Mini Victories
Select an Option:
 ------------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
1
1: Diary
2: Travel
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
9: Mini Victories
Select an Option:
 ------------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
```



```
 ------------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
5
Enter the id of the tag you would like to remove:
2
Select an Option:
 ------------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
1
1: Diary
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
9: Mini Victories
Select an Option:
 ------------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
```



```
4) Update tag
5) Delete tag
1
1: Diary
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
9: Mini Victories
Select an Option:
 -----------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
2
1: Diary
3: Trying my Best
4: Computer
5: Photo
6: Lifestyle
8: Depressos Espressos
9: Mini Victories
Enter the id of the tag which you would like to view the journal entries from:
8
3: 2021-08-17 Entry 1
        this is content

Select an Option:
 -----------------------------------
1) Display all tags
2) Search for Journal Entries via tag
3) Create new tag
4) Update tag
5) Delete tag
```

URL to GitHub Repository:

https://github.com/JoleneMel/GoaltivityGroupProject