CS 499 Milestone Three: Software Design and Engineering Enhancement Narrative

Southern New Hampshire University

Matthew Berdecia

CS - 499

Overview of the Artifact

The artifact I selected for enhancement is the To-Do List Application, originally developed during my coursework in CS 250. This application was initially designed as a console-based program that allowed users to add, remove, and display tasks using a simple list structure. While the program functioned correctly, it lacked important software engineering principles such as error handling, scalability, user interface design, and persistent storage.

To improve the artifact, I implemented several enhancements:

Refactoring the console-based version to improve code readability, error handling, and usability.

Adding a graphical user interface (GUI) using JavaFX to make the application more user-friendly.

Integrating persistent storage using file handling (text files) and a SQLite database to save tasks between sessions.

Enhancing user experience by adding features such as task saving, loading, and database-backed operations.

These enhancements significantly improve the functionality, usability, and maintainability of the application, aligning with software engineering best practices and demonstrating full-stack application development skills.

Enhancements and Code Breakdown

1. Original Console-Based Application

Code Snippet (Before Enhancements):

java

CopyEdit

import java.util.ArrayList;

public class ToDoListApp {

public static void main(String[] args) {

ToDoList toDoList = new ToDoList();

toDoList.addTask("Buy groceries");

toDoList.addTask("Finish homework");

toDoList.displayTasks();

toDoList.removeTask(1);

toDoList.displayTasks();

}

}

class ToDoList {

private ArrayList<String> tasks;

public ToDoList() {

tasks = new ArrayList<>();

}

public void addTask(String task) {

tasks.add(task);

}

public void removeTask(int index) {

tasks.remove(index); // No error handling

}

public void displayTasks() {

for (int i = 0; i < tasks.size(); i++) {

System.out.println(i + ": " + tasks.get(i));

}

}

}

Issues with the Original Code

No Error Handling: If an invalid task index is entered, the program crashes.

Lack of Persistent Storage: Tasks disappear when the program closes.

Limited User Interaction: No way for users to interact dynamically.

Poor Code Structure: The program does not separate concerns (business logic and UI are mixed).

2. Enhanced Console-Based Version with Error Handling and File Persistence

Code Snippet (Refactored Console Version):

java

CopyEdit

import java.util.ArrayList;

import java.util.Scanner;

import java.io.\*;

public class ToDoListApp {

public static void main(String[] args) {

ToDoList toDoList = new ToDoList();

toDoList.loadTasksFromFile("tasks.txt");

Scanner scanner = new Scanner(System.in);

String command;

System.out.println("Welcome to the To-Do List App!");

while (true) {

System.out.println("Commands: add, remove, display, save, exit");

System.out.print("Enter a command: ");

command = scanner.nextLine().trim().toLowerCase();

switch (command) {

case "add":

System.out.print("Enter a task: ");

toDoList.addTask(scanner.nextLine());

break;

case "remove":

System.out.print("Enter the index of the task to remove: ");

try {

int index = Integer.parseInt(scanner.nextLine());

toDoList.removeTask(index);

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid index.");

}

break;

case "display":

toDoList.displayTasks();

break;

case "save":

toDoList.saveTasksToFile("tasks.txt");

System.out.println("Tasks saved to file.");

break;

case "exit":

toDoList.saveTasksToFile("tasks.txt");

System.out.println("Exiting the app. Goodbye!");

scanner.close();

return;

default:

System.out.println("Invalid command. Try again.");

}

}

}

}

Enhancements in this Version

File Handling Added: Tasks are saved and loaded from a text file.

Error Handling: Prevents crashes when removing tasks.

Interactive Menu: Users can continuously add, remove, and save tasks dynamically.

3. JavaFX GUI Version

Code Snippet (GUI Implementation):

java

CopyEdit

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.io.\*;

public class ToDoListAppGUI extends Application {

private ToDoList toDoList = new ToDoList();

private ListView<String> listView = new ListView<>();

@Override

public void start(Stage primaryStage) {

TextField taskInput = new TextField();

taskInput.setPromptText("Enter a new task");

Button addButton = new Button("Add Task");

Button removeButton = new Button("Remove Task");

addButton.setOnAction(e -> {

String task = taskInput.getText();

if (!task.isEmpty()) {

toDoList.addTask(task);

updateListView();

taskInput.clear();

}

});

removeButton.setOnAction(e -> {

int selectedIndex = listView.getSelectionModel().getSelectedIndex();

if (selectedIndex >= 0) {

toDoList.removeTask(selectedIndex);

updateListView();

}

});

VBox layout = new VBox(10, taskInput, addButton, removeButton, listView);

Scene scene = new Scene(layout, 300, 400);

primaryStage.setTitle("To-Do List App");

primaryStage.setScene(scene);

primaryStage.show();

}

private void updateListView() {

listView.getItems().setAll(toDoList.getTasks());

}

public static void main(String[] args) {

launch(args);

}

}

Enhancements in this Version

Graphical User Interface (GUI): Users can manage tasks through buttons instead of text commands.

Better User Experience: No need to manually type commands.

ListView Integration: Displays tasks dynamically.

4. Database Integration (SQLite)

Code Snippet (Database Integration with SQLite):

java

CopyEdit

import java.sql.\*;

public class ToDoListDatabase {

private Connection connection;

public ToDoListDatabase() {

try {

connection = DriverManager.getConnection("jdbc:sqlite:tasks.db");

Statement statement = connection.createStatement();

statement.execute("CREATE TABLE IF NOT EXISTS tasks (id INTEGER PRIMARY KEY, description TEXT)");

} catch (SQLException e) {

System.out.println("Error connecting to the database: " + e.getMessage());

}

}

public void addTask(String taskDescription) {

try (PreparedStatement statement = connection.prepareStatement("INSERT INTO tasks (description) VALUES (?)")) {

statement.setString(1, taskDescription);

statement.executeUpdate();

} catch (SQLException e) {

System.out.println("Error adding task: " + e.getMessage());

}

}

}

Enhancements in this Version

Data Persistence with SQLite: Tasks are stored in a database rather than a text file.

More Scalable Solution: Suitable for larger applications.

Conclusion

This enhancement project transformed a simple console-based application into a scalable, user-friendly, and database-backed system. The improvements showcase software engineering best practices, including modular design, persistent storage, GUI development, and database management. These skills align with professional software development standards and make the artifact a strong addition to my ePortfolio.