Elliot Niemann

Aaren Patel

Johan Jaramillo

Orlando Osorio Garcia

I pledge my honor that I have abided by the Stevens Honor System

**User Story:**

Name: Task List

Description: As a user I want to be able to be able to add to and manage a to do list

Acceptance Test: I should be able to add my tasks to the list, be able to mark them as complete, mark them as to do again if need be, edit them, and delete them

**Manual Test Cases:**

Manual Test Case #1:

-----------------------------

Test Case ID: 1

Description: Verify that a task can be added, is marked as incomplete as default, can be marked as complete, and then verified as complete.

Preconditions: Empty task list.

Steps:

1. Start the program

2. Select "Add Task" from the menu.

3. Enter "Study for exam" as the description.

4. Verify the task list contains "Study for exam" as an incomplete task.

5. Mark the task as complete.

6. Verify that "Study for exam" is now marked as complete.

Expected Result:

1. After step 4, task should be marked as incomplete.

2. After step 6, task should be marked as complete.

Manual Test Case #2:

-----------------------------

Test Case ID: 2

Description: Verify that a task can be added, and the task description can be updated successfully.

Preconditions: Empty task list.

Steps:

1. Start the program.

2. Select "Add Task" from the menu.

3. Enter "Study for Exam" as the description.

4. Verify the task list contains "Study for Exam" as an incomplete task.

5. Select "Update Task" from the menu.

6. Choose the task "Study for Exam" for editing.

7. Enter the new description "Study for CS-555 Exam".

8. Verify the task list now contains "Study for CS-555 Exam" instead of "Study for Exam".

Expected Result:

1. After step 4, the task list should contain "Study for Exam" as an incomplete task.

2. After step 8, the task list should contain "Study for CS-555 Exam" as an incomplete task, verifying that the description was updated successfully.

Manual Test Case #3:

-----------------------------

Test Case ID: 3

Description: Verify that multiple tasks can be added and then deleted one by one, ensuring each deletion occurs in the correct order.

Preconditions: Empty task list.

Steps:

1. Start the program.

2. Select "Add Task" from the menu.

3. Enter "Study for Exam" as the first task description.

4. Enter "Go for a run" as the second task description.

5. Enter "Complete project" as the third task description.

6. Verify that all three tasks are present in the task list in the order they were added.

7. Select "Delete Task" from the menu.

8. Choose "Go for a run" for deletion.

9. Verify the task list now contains two tasks in the correct order: "Study for Exam" and "Complete project".

10. Select "Delete Task" again.

11. Choose "Complete project" for deletion.

12. Verify the task list now contains only one task: "Study for Exam".

13. Delete the last task, "Study for Exam".

14.Verify the task list is now empty.

Expected Result:

1. After step 6, the task list should contain three tasks in the correct order.

2. After step 9, the task list should contain "Study for Exam" and "Complete project".

3. After step 12, the task list should contain only "Study for Exam".

4. After step 14, the task list should be empty, verifying that each deletion occurred correctly and in sequence without affecting the remaining tasks.

**Test Code:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class TaskListTest {

*@Test*

public void testAddAndCompleteTask() {

// Add a task and verify it is incomplete by default

TaskList taskList = new TaskList();

taskList.addTask("Study for exam");

// Verify the task is added and the list contains 1 task

*assertEquals*(1, taskList.tasks.size());

// Verify the task description

*assertEquals*("Study for exam", taskList.tasks.get(0).getTask());

// Verify the task is incomplete by default

*assertFalse*(taskList.tasks.get(0).getStatus());

// Mark the task as complete

////// TASKS IN LIST ARE OFFSET BY 1 ////////////

taskList.completeTask(1);

// Verify the task is now marked as complete

*assertTrue*(taskList.tasks.get(0).getStatus());

}

*@Test*

public void testEditTask() {

// Add a task

TaskList taskList = new TaskList();

taskList.addTask("Study for Exam");

// Verify the task is added

*assertEquals*(1, taskList.tasks.size());

// Verify the task description

*assertEquals*("Study for Exam", taskList.tasks.get(0).getTask());

// Edit the task description

taskList.editTask(1, "Study for CS-555 Exam");

// Verify the updated task description

*assertEquals*("Study for CS-555 Exam", taskList.tasks.get(0).getTask());

}

*@Test*

public void testMultipleTaskDeletion() {

// Add three tasks

TaskList taskList = new TaskList();

taskList.addTask("Study for Exam");

taskList.addTask("Go for a run");

taskList.addTask("Complete project");

// Verify all tasks are added

*assertEquals*(3, taskList.tasks.size());

*assertEquals*("Study for Exam", taskList.tasks.get(0).getTask());

*assertEquals*("Go for a run", taskList.tasks.get(1).getTask());

*assertEquals*("Complete project", taskList.tasks.get(2).getTask());

// Delete the second task

taskList.deleteTask(2);

// Verify the task list contains the correct remaining tasks

*assertEquals*(2, taskList.tasks.size());

*assertEquals*("Study for Exam", taskList.tasks.get(0).getTask());

*assertEquals*("Complete project", taskList.tasks.get(1).getTask());

// Delete the third task

taskList.deleteTask(2);

// Verify only the first task remains

*assertEquals*(1, taskList.tasks.size());

*assertEquals*("Study for Exam", taskList.tasks.get(0).getTask());

// Delete the first task

taskList.deleteTask(1);

// Verify the task list is now empty

*assertEquals*(0, taskList.tasks.size());

}

*@Test*

public void testAddCompleteAndUncompleteTask() {

// Add a task and verify it is incomplete by default

TaskList taskList = new TaskList();

taskList.addTask("Study for exam");

// Verify the task is added and the list contains 1 task

*assertEquals*(1, taskList.tasks.size());

// Verify the task description

*assertEquals*("Study for exam", taskList.tasks.get(0).getTask());

// Verify the task is incomplete by default

*assertFalse*(taskList.tasks.get(0).getStatus());

// Mark the task as complete

////// TASKS IN LIST ARE OFFSET BY 1 ////////////

taskList.completeTask(1);

// Verify the task is now marked as complete

*assertTrue*(taskList.tasks.get(0).getStatus());

// Now, mark task as not complete

taskList.uncompleteTask(1);

//Verify task is now marked as incomplete

*assertFalse*(taskList.tasks.get(0).getStatus());

}

*@Test*

public void testEmptyListErrors() {

TaskList taskList = new TaskList();

//Attempt to complete a non-existing task 1

taskList.completeTask(1);

//Check that task list remains empty afterwards

*assertEquals*(0, taskList.tasks.size());

//Attempt to mark a non-existing task 1 as to do

taskList.uncompleteTask(1);

//Check that task list remains empty afterwards

*assertEquals*(0, taskList.tasks.size());

//Attempt to edit a non-existing task 1

taskList.editTask(1, "Add an item to my to do list");

//Check that task list remains empty afterwards

*assertEquals*(0, taskList.tasks.size());

}

}

**Code:**

Task.java

public class Task {

private String description; //Description of task

private boolean status; //Completion status of task

public Task(String input) {

this.description = input; //Sets description as input

this.status = false; //By default status is false for incomplete

}

public String getTask() {

return description;

}

public void setTask(String input) {

this.description = input;

}

public boolean getStatus() {

return status;

}

public void setStatus(boolean input) {

status = input;

}

}

TaskList.java

import java.util.ArrayList;

import java.util.List;

public class TaskList {

List<Task> tasks;

public TaskList() {

this.tasks = new ArrayList<>(); //Initialize task list

}

public void addTask (String description) { //Function for adding new tasks

Task newTask = new Task(description);

tasks.add(newTask);

System.***out***.println("Task added!");

}

public void completeTask(int taskNum) { //Function for marking task as completed

if(taskNum >= 1 && taskNum <= tasks.size()) { //Check if task number is valid

tasks.get((taskNum - 1)).setStatus(true); //If valid marks completion status as true

System.***out***.println("Task marked as complete!");

} else {

System.***out***.println("Invalid task number");

}

}

public void uncompleteTask(int taskNum) { //Function for marking task as to do

if(taskNum >= 1 && taskNum <= tasks.size()) { //Check if task number is valid

tasks.get((taskNum - 1)).setStatus(false); //If valid marks completion status as false

System.***out***.println("Task marked as to do!");

} else {

System.***out***.println("Invalid task number");

}

}

public void deleteTask(int taskNum) { //Function for deleting tasks

if(taskNum >= 1 && taskNum <= tasks.size()) { //Check if task number is valid

tasks.remove(taskNum - 1); //Removes task

System.***out***.println("Task deleted!");

} else {

System.***out***.println("Invalid task number");

}

}

public void editTask(int taskNum, String newDesc) { //Function for editing task description

if(taskNum >= 1 && taskNum <= tasks.size()) { //Check if task number is valid

tasks.get((taskNum - 1)).setTask(newDesc); //Updates with input description

System.***out***.println("Task updated!");

} else {

System.***out***.println("Invalid task number");

}

}

public void printFullList() { //Function for printing full list

if(tasks.isEmpty()) {

System.***out***.println("You currently have no tasks!");

} else {

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

Task curr = tasks.get(i);

System.***out***.print((i+1)+". " + curr.getTask());

if(curr.getStatus()) { //Checks if task has been completed to print accordingly

System.***out***.println(" - Complete!");

} else {

System.***out***.println(" - To Do");

}

}

}

}

public void printTodo() { //Function for printing ONLY the incomplete tasks

if(tasks.isEmpty()) {

System.***out***.println("You currently have no tasks!");

} else {

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

Task curr = tasks.get(i);

if(!curr.getStatus()) {

System.***out***.println((i+1)+". " + curr.getTask());

}

}

}

}

public void printComplete() { //Function for printing ONLY the complete tasks

if(tasks.isEmpty()) {

System.***out***.println("You currently have no tasks!");

} else {

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

Task curr = tasks.get(i);

if(curr.getStatus()) {

System.***out***.println((i+1)+". " + curr.getTask());

}

}

}

}

}

userInput.java

import java.util.Scanner;

public class userInput {

public static void main(String[] args) {

TaskList taskList = new TaskList();

Scanner scanner = new Scanner(System.***in***);

while (true) {

System.***out***.println("\nChoose an action:");

System.***out***.println("1. Add task");

System.***out***.println("2. Mark task as complete");

System.***out***.println("3. Mark task as incomplete");

System.***out***.println("4. Update task");

System.***out***.println("5. Delete task");

System.***out***.println("6. View all tasks");

System.***out***.println("7. View complete tasks");

System.***out***.println("8. View incomplete tasks");

System.***out***.println("9. Exit");

System.***out***.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.***out***.println();

switch (choice) {

case 1:

System.***out***.print("Enter task description: ");

String description = scanner.nextLine();

taskList.addTask(description);

break;

case 2:

taskList.printTodo();

if(taskList.tasks.size() > 0) {

System.***out***.println("\nWhich task should be marked complete? ");

int taskNum = scanner.nextInt();

scanner.nextLine();

taskList.completeTask(taskNum);

}

break;

case 3:

taskList.printComplete();

if(taskList.tasks.size() > 0) {

System.***out***.println("\nWhich task should be marked incomplete? ");

int taskNum = scanner.nextInt();

scanner.nextLine();

taskList.uncompleteTask(taskNum);

}

break;

case 4:

taskList.printFullList();

if(taskList.tasks.size() > 0) {

System.***out***.println("\nWhich task should be updated? ");

int taskNum = scanner.nextInt();

scanner.nextLine();

System.***out***.println("\nInput new description: ");

String newDesc = scanner.nextLine();

taskList.editTask(taskNum, newDesc);

}

break;

case 5:

taskList.printFullList();

if(taskList.tasks.size() > 0) {

System.***out***.println("Which task should be deleted? ");

int taskNum = scanner.nextInt();

scanner.nextLine();

taskList.deleteTask(taskNum);

}

break;

case 6:

System.***out***.println("Task list:");

taskList.printFullList();

break;

case 7:

System.***out***.println("Complete Tasks:");

taskList.printComplete();

break;

case 8:

System.***out***.println("Incomplete Tasks:");

taskList.printTodo();

break;

case 9:

System.***out***.print("Exiting");

scanner.close();

return;

default:

System.***out***.println("Invalid action");

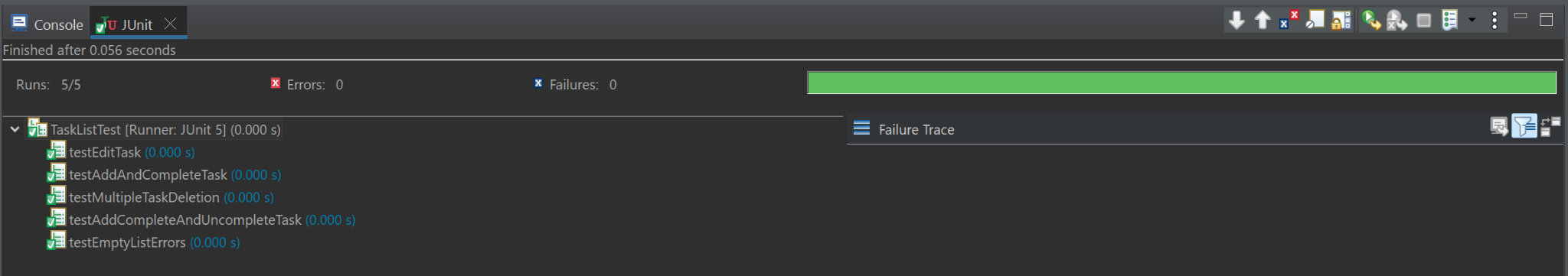
}

}

}

}

**Test Results:**

****