```
package todolist;

import java.util.LinkedList;
import java.util.Stack;
import javax.swing.JOptionPane;

public class Todolist {

    static LinkedList<String[]> list = new LinkedList();
    static LinkedList<String[]> completedTasks = new LinkedList<>();
    static Stack<String[]> undoStack = new Stack<>();

    public static void main(String[] args) {
        while (true) {

            String[] option = {"Add Task", "Mark Task Done", "View Task", "View Completed
task", "Undo", "Exit"};

            int choice = JOptionPane.showOptionDialog(null, "Select an option:", "To-Do
List",
                    JOptionPane.DEFAULT_OPTION,
JOptionPane.INFORMATION_MESSAGE, null, option, option[0]);

            switch (choice) {
                case 0:
                    addtask();
                    break;

                case 1:
                    markTaskDone();
                    break;

                case 2:
                    viewtask();
                    break;

                case 3:
                    viewCompletedTask();
                    break;

                case 4:
                    undo();
                    break;

                case 5:
                    exit();
                    break;

                default:

                    JOptionPane.showMessageDialog(null, "Click Exit", "Error",
JOptionPane.ERROR_MESSAGE);

            }

        }
    }

    public static void addtask() {

        String description;
        String date;
        String time;

        while (true) {
            description = JOptionPane.showInputDialog(null, "Enter Task Description:", "Add
Task", JOptionPane.INFORMATION_MESSAGE);
            if (description != null && !description.trim().isEmpty()) {
                break;
            }
            JOptionPane.showMessageDialog(null, "Task description cannot be empty.",
"Add Task", JOptionPane.WARNING_MESSAGE);
        }

        while (true) {
            date = JOptionPane.showInputDialog(null, "Enter Due Date (e.g., 2024-10-11):",
"Add Task", JOptionPane.INFORMATION_MESSAGE);
            if (date != null && !date.trim().isEmpty()) {
                break;
            }
            JOptionPane.showMessageDialog(null, "Due date cannot be empty.", "Add Task",
JOptionPane.WARNING_MESSAGE);
        }

        while (true) {
            time = JOptionPane.showInputDialog(null, "Enter Due Time (e.g., 14:00):", "Add
Task", JOptionPane.INFORMATION_MESSAGE);
            if (time != null && !time.trim().isEmpty()) {
                break;
            }
            JOptionPane.showMessageDialog(null, "Due time cannot be empty.", "Add Task",
JOptionPane.WARNING_MESSAGE);
        }

        String[] task = {description, date, time};
        list.add(task);
        JOptionPane.showMessageDialog(null, "Task added successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);

    }

    public static void viewtask() {

        if (list.isEmpty()) {
            JOptionPane.showMessageDialog(null, "No tasks to display.", "View Tasks",
JOptionPane.INFORMATION_MESSAGE);

        } else {

            String tasks = "Your Tasks:\n";

            for (int i = 0; i < list.size(); i++) {
                String[] task = list.get(i);
                tasks += (i + 1) + ". Description: " + task[0]
                        + "\n      Due Date: " + task[1]
                        + "\n      Due Time: " + task[2] + "\n\n";
            }

            JOptionPane.showMessageDialog(null,          tasks,          "View          Tasks",
JOptionPane.INFORMATION_MESSAGE);

        }

    }

    public static void markTaskDone() {
        if (list.isEmpty()) {
            JOptionPane.showMessageDialog(null, "No tasks to mark as done.", "Mark Task
Done", JOptionPane.INFORMATION_MESSAGE);
        } else {

            String tasks = "Your Tasks:\n";
            for (int i = 0; i < list.size(); i++) {
                String[] task = list.get(i);
                tasks += (i + 1) + ". Description: " + task[0]
                        + "\n      Due Date: " + task[1]
                        + "\n      Due Time: " + task[2] + "\n\n";
            }

            String input = JOptionPane.showInputDialog(null, tasks + "Enter task number to
mark as done:", "Mark Task Done", JOptionPane.INFORMATION_MESSAGE);

            if (input != null) {
                try {
                    int index = Integer.parseInt(input) - 1;
                    if (index >= 0 && index < list.size()) {

                        String[] task = list.remove(index);
                        completedTasks.add(task);

                        undoStack.push(new String[]{"done", task[0], task[1], task[2]});
                        JOptionPane.showMessageDialog(null,  "Task  marked  as  done!",
"Success", JOptionPane.INFORMATION_MESSAGE);
                    } else {

                        JOptionPane.showMessageDialog(null,  "Invalid  task  number.",
"Error", JOptionPane.ERROR_MESSAGE);
                    }
                } catch (NumberFormatException e) {

                    JOptionPane.showMessageDialog(null, "Please enter a valid number.",
"Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        }
    }

    public static void viewCompletedTask() {

        if (completedTasks.isEmpty()) {
            JOptionPane.showMessageDialog(null, "No completed tasks to display.",
"Completed Tasks", JOptionPane.INFORMATION_MESSAGE);
        } else {
            String tasks = "Completed Tasks:\n";
            for (int i = 0; i < completedTasks.size(); i++) {
                String[] task = completedTasks.get(i);
                tasks += (i + 1) + ". " + task[0] + " - " + task[1] + " " + task[2] + "\n";
            }
            JOptionPane.showMessageDialog(null, tasks, "Completed Tasks",
JOptionPane.INFORMATION_MESSAGE);
        }

    }

    public static void undo() {

        if (undoStack.isEmpty()) {
            JOptionPane.showMessageDialog(null, "No actions to undo.", "Undo",
JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        String[] lastAction = undoStack.pop();
        String actionType = lastAction[0];
        String[] task = {lastAction[1], lastAction[2], lastAction[3]};

        if ("add".equals(actionType)) {

            for (int i = 0; i < list.size(); i++) {
                String[] currentTask = list.get(i);
                if (currentTask[0].equals(task[0]) && currentTask[1].equals(task[1]) &&
currentTask[2].equals(task[2])) {
                    list.remove(i);
                    JOptionPane.showMessageDialog(null, "Last added task removed.",
"Undo", JOptionPane.INFORMATION_MESSAGE);
                    break;
                }
            }
        } else if ("done".equals(actionType)) {

            for (int i = 0; i < completedTasks.size(); i++) {
                String[] completedTask = completedTasks.get(i);
                if (completedTask[0].equals(task[0]) && completedTask[1].equals(task[1])
&& completedTask[2].equals(task[2])) {
                    completedTasks.remove(i);
                    list.add(task);
                    JOptionPane.showMessageDialog(null, "Marked task moved back to to
-do list.", "Undo", JOptionPane.INFORMATION_MESSAGE);
                    break;
                }

            }

        }

    }
```

```
	public static void exit() {
		JOptionPane.showMessageDialog(null, "Thank you for using our To-Do List.", "EXIT",
JOptionPane.INFORMATION_MESSAGE);
		System.exit(0);

	}

}
```