Source code:

```
import java.util.ArrayList;

import java.util.Scanner;

import java.io.*;

 // Simple To-Do List Manager using ArrayList and File Handling

public class TodoListManager {

   static class Task {

      String description;

      boolean done;

       Task(String description, boolean done) {

         this.description = description;

         this.done = done;

      }

       @Override

      public String toString() {

         return (done ? "[Done] " : "[Pending] ") + description;

      }

   }

  private static ArrayList<Task> tasks = new ArrayList<>();

  private static final String FILE_NAME = "tasks.txt";    public static void main(String[]
args) {

      loadTasksFromFile();

       Scanner scanner = new Scanner(System.in);

      while (true) {

         System.out.println("\n=== To-Do List Manager ===");
```

```
        System.out.println("1. Add Task");
        System.out.println("2. Remove Task");
        System.out.println("3. Mark Task as Done");
        System.out.println("4. View Pending Tasks");
        System.out.println("5. Exit");
        System.out.print("Choose an option (1-5): ");
        String input = scanner.nextLine();
        switch (input) {
          case "1":
            addTask(scanner);
            break;
          case "2":
            removeTask(scanner);
            break;
          case "3":
            markTaskDone(scanner);
            break;
          case "4":
            viewPendingTasks();
            break;
          case "5":
            saveTasksToFile();
            System.out.println("Exiting... Goodbye!");
            scanner.close();
            System.exit(0);
            break;
```

```
            default:
                System.out.println("Invalid option. Please try again.");
                break;
        }
    }
}
private static void addTask(Scanner scanner) {
    System.out.print("Enter task description: ");
    String desc = scanner.nextLine().trim();
    if (desc.isEmpty()) {
        System.out.println("Task description cannot be empty.");
        return;
    }
    tasks.add(new Task(desc, false));
    System.out.println("Task added successfully.");
}
private static void removeTask(Scanner scanner) {
    if (tasks.isEmpty()) {
        System.out.println("No tasks to remove.");
        return;
    }
    viewAllTasks();
    System.out.print("Enter task number to remove: ");
    String input = scanner.nextLine();
    try {
        int index = Integer.parseInt(input);
```

```
        if (index < 1 || index > tasks.size()) {
            System.out.println("Invalid task number.");
            return;
        }
        Task removed = tasks.remove(index - 1);
        System.out.println("Removed task: " + removed.description);
    } catch (NumberFormatException e) {
        System.out.println("Please enter a valid number.");
    }
}
 private static void markTaskDone(Scanner scanner) {
    if (tasks.isEmpty()) {
        System.out.println("No tasks to mark as done.");
        return;
    }
    viewAllTasks();
    System.out.print("Enter task number to mark as done: ");
    String input = scanner.nextLine();
    try {
        int index = Integer.parseInt(input);
        if (index < 1 || index > tasks.size()) {
            System.out.println("Invalid task number.");
            return;
        }
        Task task = tasks.get(index - 1);
        if (task.done) {
```

```
            System.out.println("Task is already marked as done.");
        } else {
            task.done = true;
            System.out.println("Task marked as done: " + task.description);
        }
    } catch (NumberFormatException e) {
        System.out.println("Please enter a valid number.");
    }
}    private static void viewPendingTasks() {
    boolean found = false;
    System.out.println("\nPending Tasks:");
    for (int i = 0; i < tasks.size(); i++) {
        Task task = tasks.get(i);
        if (!task.done) {
            System.out.println((i + 1) + ". " + task.description);
            found = true;
        }     }
    if (!found) {
        System.out.println("No pending tasks found.");
    }
}
 private static void viewAllTasks() {
    System.out.println("\nAll Tasks:");
    for (int i = 0; i < tasks.size(); i++) {
        Task task = tasks.get(i);
```

```
        String status = task.done ? "[Done]" : "[Pending]";        System.out.println((i + 1)
+ ". " + status + " " + task.description);      }
```