Documentation of a Project for the advanced course in 'Object Oriented Programming in Java'

By W.Kidane, V.Ion-Cislaro, D. Konjicija, A. Ganeshkumar, B.Cav

Object Oriented Programming in JAVA

Project Documentation

Lecturer: Robin Müller-Bady

Date: 01-12-2023 to DD-MM-YYYY

TABLE OF CONTENTS

GROUP MEMBERS

Name of Member	**Responsibilities** (entfernen?)
Warsay Kidane 1386765	GUI
Victor Cislaro 1482313	API
Drago Konjicija 1313453	API
Atheesan Ganeshkumar 1397660	GUI
Bahadir Cav 1366358	GUI

INTRODUCTION

This Project‘s task is the creation of a **JAVA application intended to interact with a drone simulation system** that is **operates via a web-based RESTful API**. It is actively generating detailed reports (e.g. current state & flight course) on a fleet of simulated drones that is ready to be given out to the operator. Information such as **manufacturer, battery level, positional alignment, velocity and cargo specifics** can be retrieved.

The drone data is subdivided into these three separate categories:

Drone model
This summarizes static data that is consistent throughout all drones of a specific model, such as top speed or manufacturing details.

Individual Drone
This relates to unique information relevant to an individual drone, distinguished by unique identifiers like serial numbers or specific cargo weights. (COPY PASTE ----_--)

Drone Dynamics
These are the temporal dynamic data points for each drone, capturing real-time specifics like velocity, location, or battery life. (COPY PASTE ÄAÖDÖLDSADA)

REQUIREMENTS ANALYSIS

- How does the Web Server work? What is it giving as an output and taking as input?
  → output’s a JSON format with info such as status and other data, (input??)
- Web Server: **dronesim.facets-labs.com** (presents a snapshot of the drone fleet)
  → further access requires a authentication token (e.g. ”/api/?format=json”)
- Special Requirement for access to the drone fleet webserver:
  → VPN: FortiClient or direct connection to universities network
  → Log In via CIT credentials
  →API is configured as READ-ONLY, means only “HTTP GET” operations

Example of Drone Status in JSON-Format


```
{
"drone": "http://10.18.2.60/api/drones/2/?format=json",
"timestamp": "2023-10-16T17:00:47.341575+02:00", "speed": 32,
"align roll": "0.00",
"align pitch": "0.00",
"align yaw": "0.00",
"longitude": "50.110924000",
"latitude": "8.682127000", "battery status": 380,
"last seen": "2023-10-16T17:00:47.341575+02:00",
"status": "ON"
}
```


dronesim.facets-labs.com/api/?format=json

Marks the gateaway of the API which hosts drone-related data
Makes data accessible so it can be directly fetched in JSON format

Project Description

Objectives
Real-Time Monitoring: To provide a real-time overview of all drones in the fleet, including their location, status, and current activity.
Data Analysis and Reporting: To analyze the data received from drones for insights into performance, usage patterns, and operational efficiency.
User Interface Development: To develop an intuitive and user-friendly interface that allows users to interact seamlessly with the drone fleet.

Individual Parts
API Integration:
Seamlessly connect to the web-based drone simulation API.
Fetch and format all necessary information to make it user friendly.

Drone Dashboard:
A dashboard that displays all active drones on a real-time map. It also gives the user the option to access more information about the drone's current status.

Flight Dynamics:
This overview displays all dynamic values the drones have.

Historical Analysis:
Works as a time-based Drone dashboard where all necessary informations such as location can be viewed by either using a controller or by choosing a custom time.

Technology Stack
Programming Language: Java
Frontend: JavaFX for GUI development

API Communication: Java's HttpURLConnection for RESTful API interaction
Data Parsing: JSON parsing with libraries such as Gson
Database: API/dronedynamics

Milestone 1

Our goal for this Milestone was understanding the Project objects and organizing our workload so we can reach them. The first step was to brainstorm in order to fulfill our requirements. We began by finding out which information we have access to. To do that we visited https://dronesim.facets-labs.com/simulator/ where we found out that some information can be used to create requirements such as putting the drones onto a dynamic map, that changes its position based on the location of drones that are active, viewing new information such as average speed or remaining battery time.

The GUI source codes are simple prototypes without deep function. They do not serve any function yet and do not fetch any data from the web server. The following 3 snippets on the next pages are all GUI prototypes marked as “GUI EXAMPLE CODE PROTOTYPE #XY”.

Team Meetings

07.12.2023 - First Meeting, get to know each other and discuss the project description. All Team members were present.

14.12.2023 - Further discussion of project requirements and exchanged general ideas on project development. Team member roles: Who works on backend and who works on frontend/GUI coding and deployment. All Team members were present.

21.12.2023 - Connecting to API and fetching data in JSON format and how to use that data for GUI development. Got the test Java file on Campuas working. All Team members were present.

22.12.2023 - Worked together on documentation for first milestone submission. All Team members were present.

We went through the exercises on “Campuas” to develop our Java skills. We are currently working on getting started with the project’s backend programming and watching the newly added video lectures on the GUI.

GUI EXAMPLE CODE (PROTOTYPE #1)

```
import javax.swing.*; // library for gui
import java.awt.BorderLayout;


public class GUI extends JFrame // class GUi extended by JFrame which is
in lib
{

    public GUI() //constructor for gui class
    {
        JLabel label = new JLabel("Dronedata: "); // jlabel is only a
text field
        JButton button = new JButton("Fetch Drone Data"); // button for
fetching data

        setLayout(new BorderLayout()); //setss Layoutmanager to
Borderlayout

        add(label, BorderLayout.NORTH); //position of label is top of
window
        add(button, BorderLayout.CENTER); //position of label is center
window

        button.addActionListener( e -> { // e -> for compact little
function without creating new method or class

            //Action code here

            JOptionPane.showMessageDialog(this, "still need some wokring
bruv"); //JOption for Pop-Up
        });

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //standard for
closing app -> closes window
        setSize(400, 300); //400x300 window
        setVisible(true); //make window visible
    }

    public static void main(String[] args)
    {
        SwingUtilities.invokeLater(() -> new GUI());
    }

}
```

GUI EXAMPLE CODE (PROTOTYPE #2):

```
package de.oop.guitest;

import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

//Working on connecting to API to get drone data

public class GUIDroneTest {

	public static void main(String[] args) {
		// New JFrame (main window)
		JFrame mainFrame = new JFrame("Drone Fleet");
		mainFrame.setSize(400, 300);
		mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		// New JPanel to hold components
		JPanel mainPanel = new JPanel(new FlowLayout());

		// Create 10 buttons
		for (int i = 51; i <= 70; i++) {
			JButton button = new JButton("Drone " + i);
			mainPanel.add(button);

			// Add ActionListener to each button (listen for
click)
			button.addActionListener(new ActionListener() {
				@Override
				public void actionPerformed(ActionEvent
e) {
					openNewWindow("Specifics of " +
button.getText());

				}
			});
		}

		// Add the panel to the main frame
		mainFrame.add(mainPanel);

		// Set the main frame to be visible
		mainFrame.setVisible(true);
	}

	// Method to open a new window
	private static void openNewWindow(String message) {
		JFrame newFrame = new JFrame("New Window");
		newFrame.setSize(300, 200);

		// Display the message in a JLabel
```

```
56                  JLabel label = new JLabel(message);
57                  newFrame.add(label);
58
59                  // Set the new frame to be visible
60                  newFrame.setVisible(true);
            }
    }
```