Object-Oriented Programming in Java - Advanced Course

Project Description

Prof. Dr. Robin Müller-Bady

Winter 2023/2024

1 Introduction

The examination of this module is in form of a group project. The resulting program has to be fully functional and documented. Each team should come together and work autonomously and is responsible for distributing tasks fair and equal among all team members. The task must be solved by the team. Help of external people or any fraudulent activities, e.g. copying source code from different sources without a reference will result in, at least, disqualification or legal consequences. The amount of own work must significantly exceed the amount of code copied from various sources even if referenced. This does not apply for the use of external libraries for extending own program features in a reasonable ratio of own code vs. external code. Any suspicious activity of that kind will be reported to the faculty examination board for further investigation. All further conditions of the examination regulation apply.

The following constraints apply to the module examination:

Start Date December 1st 2023, 00:01 AM

Submission Deadline February 2nd 2024, 11:59 PM via Moodle (campuas) platform

Team size 4-5 team members

Topic The presented project from Section 3

Submission Details The following things are required for a successful submission

1. A working program as runnable *.jar file including all necessary libraries and the source code (be careful with the discussed jar-in-jar problem) and the (fully functional and configured) zipped project (e.g. eclipse) including all necessary files e.g. project files for importing. The source code must be commented as mentioned in the clean code lecture. Make sure that your program runs at least under Linux with JDK 17.

Please notice that a dysfunctional program, i.e., a program that does not start or execute properly under the given requirements, will be marked 0 points!

2. Documentation as separate document (PDF file) including technical details of the program, user documentation and declaration of the individual team member performance. A milestone document has to be submitted every 3 weeks (after week 3, after week 6) via moodle. Preferably, the milestone document is one developing document over the whole project run time which is finally submitted as project documentation. An example is given in the moodle system.
3. The declaration of authorship (German: “Ehrenwörtliche Erklärung”) must be signed by each team member and added to the submitted documentation. You find an example text below.
4. Submit the software and required documents as archive in *.zip/*.tar.gz format via the moodle platform in the specific section (will be announced). The archive must contain the runnable *.jar file, the full source code, the documentation, the milestone documents, and the signed/scanned declaration of authorship.

Presentation After the submission each individual (person) has to present the team solution and the participation to it as part of a group presentation. The presentation should take around 15 to 20 minutes per group.

Milestone presentations All groups have to give a short presentation (around 10 minutes) about each individual milestone during the exercises. Details will be announced during the semester.

Use of external tools You must be able to explain each detail of your code during the presentation if asked. In case you use external tools, e.g., AI-based tools, you have to (1) declare where and what you used it for and (2) must be able to to explain in detail what happens during execution of code snippets. Points may be deducted in case the relation between own work and external help is not adequate.

Hints It is also possible to implement only parts of the given requirements. In case you do that, please make sure that your program is running even in that case. If parts of the specified program are missing, points will be deducted.

It is necessary to successfully submit every required item by the closing date. In addition, a successful participation in the final presentation is necessary.

Declaration of Authorship

Example for a declaration of authorship ("Ehrenwörtliche Erklärung"):

I hereby declare that the submitted project is my own unaided work or the unaided work of our team. All direct or indirect sources used are acknowledged as references.

I am aware that the project in digital form can be examined for the use of unauthorized aid and in order to determine whether the project as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future projects submitted. Further rights of reproduction and usage, however, are not granted here.

This work was not previously presented to another examination board and has not been published.

First and last name	City, date and signature

2 Grading Criteria

The grading of the projects is based on the given lectures and the individual autonomous learning of the topic. The module is passed with a total of 50 points (50%). See Table 1 for details of the grading structure. For a detailed view of the necessary requirements see Table 2.

Table 1: Points vs. Grades

Grade	5,0	4,0	3,7	3,3	3,0	2,7	2,3	2,0	1,7	1,3	1,0
Points	<50	50	55	60	65	70	75	80	85	90	95^+

Table 2: Requirements

Feature	**Points**	**Description**
Runnable Program	30	The running program is covering the described basic functionalities as described in Section 3.
Milestones	20	Milestones are submitted as required. The requirements of the milestones are specified in Section 3.
Object-Orientation	5	The program was developed using an adequate object orientation. At least one abstract method and one interface must be developed and used.
Collections	5	Information will be stored in suitable Java Collections or appropriate alternative data structures.
Error Handling	5	The program contains adequate error handling. At least one own exception (extending any type of Java Exception/Throwable) must be present.
Streams and Files	5	The program works with streams and files. At least one reading and writing file access has to be made.
Threads	5	The program contains at least two threads (of which one is “main”).
Clean Code	20	The program is developed using the clean code standard(s) as presented in the lecture.
Documentation	5	The documentation inside and outside the code is present as described in Section 3. Appropriate logging is provided using java.util.Logging or comparable.
Additional Features	5	The program can contain additional features to gather extra points. Additional features are to be discussed with the lecturer in advance in order to be graded.
Total	**105**	

3 Project - Communication Network Analysis

3.1 Java Application with Drone Simulation Interface - Description

This task involves the creation of a Java application designed to interact with a drone simulation system. The system in question operates via a web-based RESTful API, actively generating and dispensing detailed reports on a fleet of simulated drones, inclusive of their current states and trajectories. Data points range from manufacturers, battery levels, and cargo specifics to positional alignments and velocities. The provided drone data is segmented into three distinct categories:

Drone Model This encapsulates static data that is consistent across all drones of a particular model, such as maximum speed or manufacturing details.

Individual Drone This pertains to unique information relevant to an individual drone, distinguished by unique identifiers like serial numbers or specific cargo weights.

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

For your development tasks, the following endpoints are crucial:

dronesim.facets-labs.com/ Presents a rudimentary snapshot of the current drone fleet, serving primarily for verification of server accessibility. The site can be reached via http or https.

/api/ Marks the gateway to the API. It hosts the entirety of drone-related data accessible via a user-friendly web interface when unmodified, or when using the query "?format=api". To directly fetch JSON formatted data, append "?format=json" to the endpoint, such as "dronesim.facets-labs.com/api/?format=json".

/api/dronetypes/ Dedicated to retrieving information pertaining to various drone models.

/api/drones/ Dedicated to accessing data specific to individual drones.

/api/dronedynamics/ Provides access to real-time dynamic data of each drone.

/redoc/ Hosts the comprehensive API documentation, essential for understanding the full capabilities and usage of the API.

Please consider the following guidelines when engaging with the API:

- A connection via the university's VPN or an internal university network link is necessary to interact with the server.
- The API is configured for read-only interactions; therefore, only HTTP GET operations are allowed.

- Retrieving data does require authentication. You will receive your username/-password during project ramp-up. However, for requests to the API, please use a Token-based authentication as explained during the lecture/exercises. See Listing 1 for an example on how to use token-based authentification. You can see your token on the dronesim website once you are logged in.
- Adjust URL parameters suitably for your intended application, whether for manual exploration or for automated retrieval within your Java application.
- The API implements pagination, limiting the number of entries returned in a single request. It is vital to navigate through the pagination to accumulate all the available data, as indicated by the "next page" pointers or a "null" value signifying the end of the dataset.
- The API maintains a denial-of-service protection, which limits the amount of requests. In case of abuse, the API will deny the access for a configured amount of time.
- JSON is the preferred way to fetch information from the API. A detailed description on this exchange format can be found on the website[1]. A library that enhances the experience in using JSON in Java will be provided in the elearning system. An example JSON file is shown in Figure 2.

Listing 1: Example: Token-based authentication

```java
final String TOKEN = "Token e77d87cxxxxdcd994";
URL url = new URL("https://...");
HttpURLConnection con;
con = (HttpURLConnection) url.openConnection();
con.setRequestProperty("Authorization", TOKEN);
con.setRequestMethod("GET");
connection.setRequestProperty("User-Agent", "XYZ");
int responseCode = connection.getResponseCode();
// process response etc.
```

The details above outline the framework and resources available for the successful completion of the Java interface to the drone simulation API. An example for an abstract program flow of the Java Rich Drone Simulator desktop application can be found in Figure 1.

[1] https://www.json.org/json-en.html

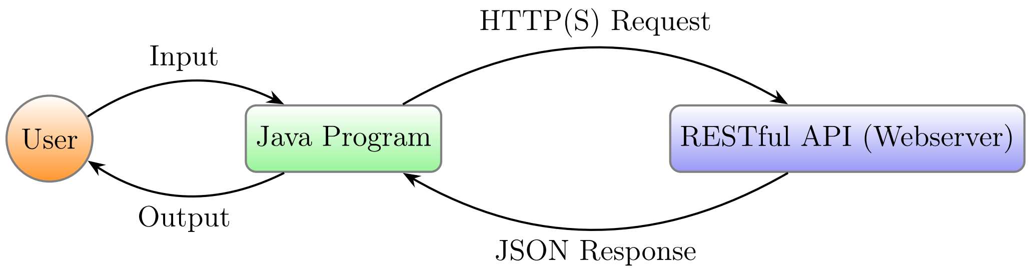

Figure 1: Program flow of the Java Rich Drone Simulator

Listing 2: Example Drone Status JSON Format

```json
{
  "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"
}
```

3.2 Java Application with Drone Simulation Interface - Challenge

This team project aims to elevate the user interaction with the drone simulator by crafting a sophisticated graphical user interface (GUI) using Java. Your mission is to architect a GUI that not only fetches but also presents data from the drone simulator API.

Your GUI should contain these essential functionalities:

API Connectivity Forge a robust link to the drone simulator API, facilitating the seamless retrieval and manipulation of data.

Drone Dashboard Curate a comprehensive and intuitive dashboard, displaying real-time statuses and particulars of all drones in operation. Also include information that is not simply retrievable, e.g., that you have to calculate using other measures.

Drone Catalog Compile a detailed catalog showcasing the various drone models available, along with their specific attributes.

Flight Dynamics Provide a dynamic presentation of each drone's flight parameters and behaviors.

Data Refresh Integrate a mechanism for users to refresh API data, either on-demand (such as by clicking a button) or at regular intervals (like auto-refresh every minute).

Historical Analysis Implement a feature allowing users to retrospectively examine drone statuses, for instance, by reviewing conditions from five minutes prior.

You are afforded a broad spectrum of artistic liberty in how you choose to represent the data. You might consider employing a palette of colors, diverse UI components, interactive tables, navigation menus, among others, to enrich the user experience. While creativity is encouraged, adherence to the specified requirements is paramount.

3.3 Documentation

The requirements for the documentation are

- JavaDoc is applied for each public method inside a class except getter and setter methods, unless they do something else than getting/setting an attribute of a class which requires documentation.
- Other source code documentation/comments are applied as necessary.
- An appropriate logging of information is provided using the java.util.Logging or a comparable library. Logging must at least be applied to each exceptional program flow, e.g., when throwing exceptions or handling error code but it is recommended to also log debug information, e.g., when handling data or reading in files etc. Logging should replace calls to "System.out." completely.
- Documentation outside the code is submitted as PDF file containing user- and technical documentation having adequate descriptions, graphics, diagrams etc. Following elements must be present:
  - User handbook (how to use the program)
  - Technical description
  - UML diagram(s)
  - Work distribution among team members preferably as table
  - Milestone documents
- Milestone documents containing the current state of the project and the participation of the team members. It has to be submitted separately every 3 weeks, i.e., after the 3rd and the 6th week, via the elearning system as one document in PDF format. Submitting the milestones is mandatory. The submission deadline the milestones for each time frame is Sundays at 11:59 PM (23:59). Submission is closed after the deadline and no further submission is possible.