Object-Oriented Programming in Java

Project Description

Prof. Dr. Robin Müller-Bady

Summer 2025

1 Introduction

The examination of this module is in form of a group project. The resulting program must be fully functional and well-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 from external sources or any fraudulent activities, such as copying source code from various sources without proper reference, will result in, at a minimum, disqualification or legal consequences. The amount of original work must significantly exceed the amount of code copied from external sources, even if properly 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 02.05.2025

Submission Deadline 12.07.2024, 17:59 via the elearning system

Milestone Deadlines 23.05.2025 and 13.06.2025

Team size 4 to 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 including all necessary libraries and the source code and the (fully functional and configured) zipped project (e.g. eclipse, IntelliJ) including all necessary files e.g. project files for importing. The source code must be commented as mentioned in the clean code lecture. Ensure that your program runs on JDK 17 or higher.

   Note that a dysfunctional program, i.e., one that does not start or execute properly under the given requirements, will receive 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 the elearning system. 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 elearning 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 full source code, the documentation, the milestone documents, and the signed/scanned declaration of authorship.
5. Mark the implemented parts of the requirements in the documentation. Only explicitly declared features can be considered for grading.

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 general conditions for the presentations will be announced during the semester.

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

Use of external tools You must be prepared to explain every 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, (2) must be able to to explain in detail what happens during execution of code snippets and (3) give the exact prompt that you used in case you use AI generated content. Points may be deducted if the balance between original work and external assistance is deemed inadequate.

Additional Software and Frameworks You are free to use any available IDE and GUI framework you like, as long as it fulfills the requirements. In the lecture, however, Linux/Unix-based Eclipse is used as the IDE, and Java Swing is used as the GUI framework. Thus, if you decide to deviate from the discussed software, there will be no support during the project regarding this software.

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.

For the submission, please provide exemplary details on where you fulfilled individual requirement including class and line number, e.g., "Threads in class HelloWorld.java:123".

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 and presented 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. Not using appropriate collections will require justification.
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/or writing file access has to be made, apart from the basic requirement.
Threads	5	The program contains at least two threads (of which one is "main").
Clean Code	15	The program is developed using the clean code standard(s) as presented in the lecture.
Documentation	10	The documentation inside and outside the code is present as described in Section 3. Appropriate logging is provided using java.util.Logging or comparable.
Total	**100**	

3 Project - Drone Simulation Interface

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 is accessible 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. Watch out for other helpful endpoints.

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 requires 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 includes denial-of-service protection, which limits the number 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

```
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

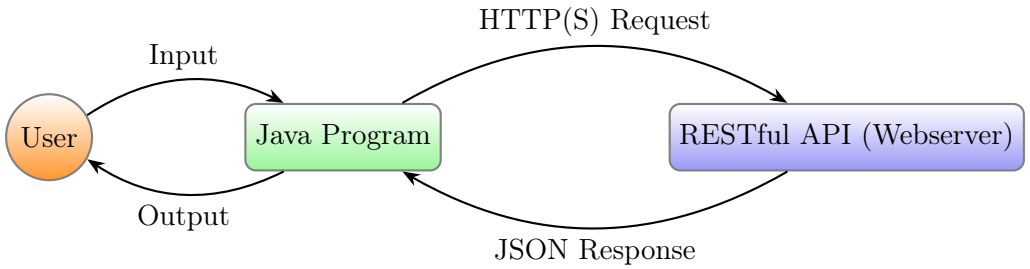

Figure 1: Program flow of the Java Rich Drone Simulator

Listing 2: Example Drone Status JSON Format. Note that this is not a complete response, some information may be missing or of a different format.

```
{
  "drone": "http://10.18.2.60/api/drones/2/?format=json",
  "timestamp": "2028-10-16T17:00:47.341575+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": "2028-10-16T17:00:47.341575+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.

The general idea of your project is to receive information from the server, process it and present it to the user via a GUI. Your Program must contain these basic functional requirements:

API Connectivity Forge a robust link to the drone simulator API, facilitating the seamless retrieval and manipulation of data. It is important that **all** information from the API must be (somehow) retrievable, e.g., using paging or different GUI elements. Especially for the retrieval of DroneDynamics, you must implement a paging mechanism, i.e., not fetching all the data at once but only the information you require at a specific time point.

Login There is no explicit login screen required. However, prior to fetching any data, it is required to input the security token and the target URL of the server into the program. It may be saved for future uses of the program but it must be possible to change it in the GUI, at least after a restart of the program.

Drone Catalog Compile a detailed catalog showcasing the various drone models available, along with their specific attributes. This might be a separate window, a separate Panel or another separate element in your GUI.

Drone Dashboard Create a comprehensive and intuitive dashboard, displaying real-time statuses and information of all drones available from the API. Also include information that is not simply retrievable, e.g., that you have to calculate using other measures. Create at least four additional features or metrics for a possible end-user that requires composition and/or calculation of data. Note them down in your final documentation.

Flight Dynamics Provide a dynamic presentation of each drone's flight parameters and behaviors. Data must be shown by drone. Do not fetch all data at once but use a pagination feature, i.e., only fetch the n elements, which are currently presented on the screen. Hint: There is a built-in pagination feature available in the API.

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).

Important Note Make sure you gather all information from the API that is provided, not just the excerpt that is presented on the example website! Use the provided API documentation available.

You have a high degree of freedom in how you choose to represent the data. While creativity is encouraged, adherence to the specified requirements is most important. You are also free to choose whatever GUI framework you use. However, if you choose one that is not presented in the lecture, support during exercise classes cannot be guaranteed.

3.3 Documentation and Non-Functional Features

The requirements for the documentation are

- JavaDoc must be applied to each public method within a class, except for getter and setter methods, unless these methods perform additional functions requiring 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 must replace calls to the systems output stream completely, if used for debugging purposes.
- Documentation outside the code is submitted as PDF file containing user- and technical documentation having adequate descriptions, graphics, diagrams etc. The following elements must be included:
  - User handbook (how to use the program), e.g., create screenhots and showcase your program flow to a possible (non-technical) user
  - Technical description, e.g., create a description that helps technical people to understand your project.
  - UML diagram(s), at least a class diagram is helpful to understand the interconnection of your classes
  - Work distribution among team members, preferably as table
  - Milestone documents

  All documents may be aggregated into one PDF file, e.g., organized as individual chapters, and may also be maintained throughout the semester and submitted as milestone document, i.e., living documentation.
- 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 announced in the elearning system. Submission is closed after the deadline and no further submission is possible.