# Java Programming

##### Objective

Design a java-based RESTful service for managing satellite events. Each event belongs to a satellite and has a creation date, description and priority. The events are stored by the service so that they can be retrieved later on. You’re free to pick any Java framework of your preference.

##### Requirements

* The service provides a REST interface supporting the following actions:
  + Get a list of events
  + Get a list of events that belong to a certain satellite.
  + Create a new event
* It should be possible to find all events that belong to a certain satellite by providing a satellite name, e.g. “ISS” or “Sentinel-1A”.
* All events are stored in some kind of data store so that they can be retrieved later on.

##### Example output

The RESTful service is expected to return output comparable to the examples below. As an example, a basepath of <http://localhost:8080/restfulservice> is used. You’re free to pick a basepath of your preference.

Getting all events:

GET <http://localhost:8080/restfulservice/events/>

// response:

[

{

"date": "2024-01-01T00:13:37",

"description": "This is some description",

"priority": "high",

"satelliteName": "SENTINEL-1A"

},{

"date": "2024-01-03T00:10:37",

"description": "This is another description",

"priority": "low",

"satelliteName": "ISS"

}

// etc…

]

Getting all events for a certain satellite:

GET <http://localhost:8080/restfulservice/events/SENTINEL-1A>

// response:

[

{

"date": "2024-01-01T00:13:37",

"description": "This is some description",

"priority": "high",

"satelliteName": "SENTINEL-1A"

},

// etc…

]

Create an event:

POST <http://localhost:8080/restfulservice/events/>

// body:

{

"date": "2024-01-02T00:13:37",

"description": "This is a new event",

"priority": "low",

"satelliteName": "SENTINEL-9"

}

// response:

[

{

"date": "2024-01-02T00:13:37",

"description": "This is a new event",

"priority": "low",

"satelliteName": "SENTINEL-9"

}

]

# Object-Oriented Programming

Explain which OOP principles you applied in the previous exercise and what their purpose is.

# Front-end

##### Objective

Using your favourite front-end framework, implement a webclient that interfaces with the RESTful service you developed in exercise 1. With this webclient, users can enter events and view events per satellite. In addition, the webclient should display information on the satellite’s orbit. This can be done by using TLE information from Celestrak (<https://celestrak.org>). TLE stands for “Two-Line Element”, a common data format for sharing orbit information. Celestrak is a public API for retrieving TLE information.

##### Requirements

* The webclient offers a form with which the user can create a new event
  + The new event should be POSTed to the RESTful service
* A table shows all events stored in the RESTful service
* The table allows the user to search for a satellite name. I.e. when the user enters “SENTINEL-1A”, only events are returned whose “satelliteName” parameter matches this search value.
* When clicking on an event, TLE information from Celestrak is shown matching the satellite name related to that event. For this, the webclient directly interfaces with the Celestrak API as a source for TLE information.
  + See the public Celestrak API on how to query the Celestrak API: <https://celestrak.org/NORAD/documentation/gp-data-formats.php>
  + Example request from Celestrak:

GET <https://celestrak.org/NORAD/elements/gp.php?NAME=sentinel-1a>

// response:   
SENTINEL-1A

1 39634U 14016A 24120.94488617 .00000293 00000+0 71860-4 0 9992

2 39634 98.1814 128.7374 0001377 88.9986 271.1371 14.59198725536520

# Docker

##### Objective

Dockerize both the RESTful service and front-end you wrote in the previous exercises. You can create one or multiple Dockerfiles. A docker-compose.yml file may be provided, with which all components can be automatically started.