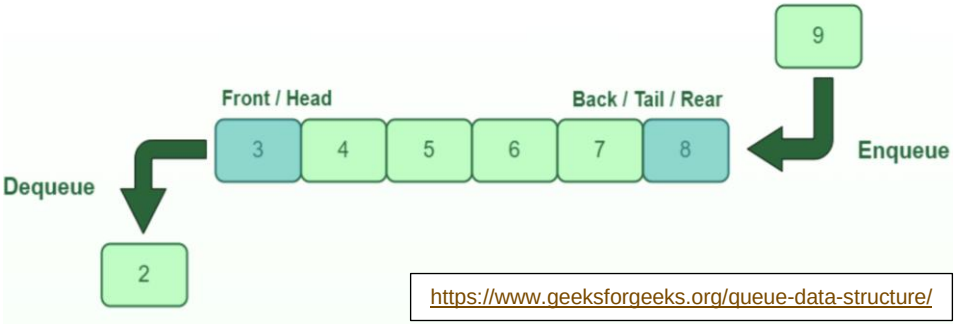
Exercise 02: Queue using Linked List

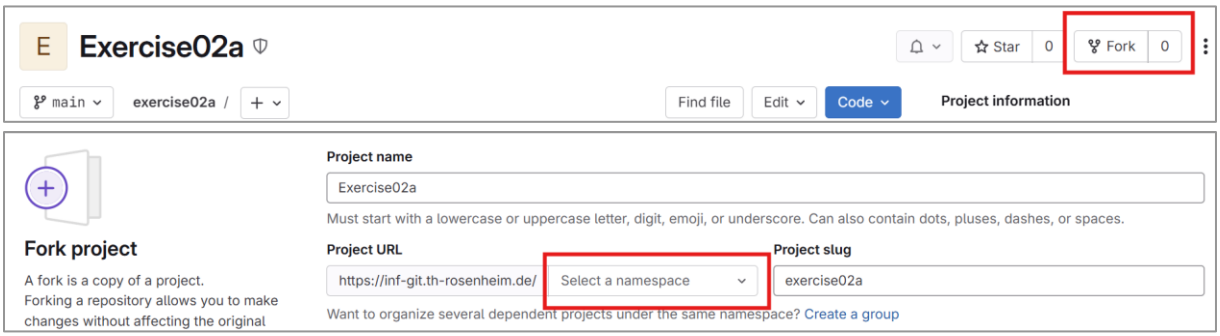
In this exercise, we will be implementing the *Queue* data structure (see `PersonQueue` interface)



Preparation

From now on we are going to use following git process for the exercises:

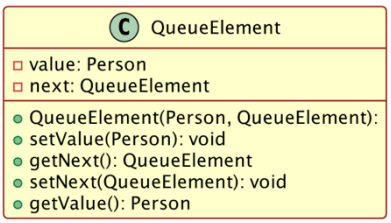
- Fork the exercise project under your own namespace (your username)



- Clone the forked repository, by using IntelliJ to import the project from Git, or by doing so "manually" in the console:
  `git clone` *`LINK TO YOUR FORKED REPOSITORY`*
- After finishing the exercise `add`, `commit` and `push` your solution.

Task 1: QueueElement Class

- Implement a class called `QueueElement` with the following private fields:
  - `value`: A `Person` object representing the data stored in the element.
  - `next`: A `QueueElement` reference pointing to the next element in the queue.



- Provide the following methods:
  - Getters and setters for both fields.
  - A constructor to initialize the fields (`value` and `next`) during instantiation.

Task 2: PersonQueueImpl Class

- Use a linked list of `QueueElement` objects as the underlying data structure in `PersonQueueImpl`.
- Include these private fields:
  - `front`: A reference to the front element in the queue.
  - `back`: A reference to the back of the queue. Wrap each `Person` object in a `QueueElement` when adding it to the queue.
- Implement the following methods:
  - `enqueue`(Person person): Add a v to the end of the queue by wrapping the `Person` in a new `QueueElement` and linking it to the current back.
  - `dequeue`(): Remove and return the `Person` from the first `QueueElement`. Throw a `NoSuchElementException` if the queue is empty.
  - `peek`(): Return (without removing) the `Person` in the first `QueueElement`. Throw a `NoSuchElementException` if the queue is empty.
  - `size`(): Return the number of elements in the queue.

Task 3: Tests

Verify that the `PersonQueueTest` tests run without errors.

How to upload your solution to the Git:

- Add any additional source files you may have to git.
  - In IntelliJ with right click on the file -> Git -> Add, or
  - In the console: `git add file1.java file2.java`
- Commit your changes
  - In IntelliJ: Git → Commit
  - In the console: `git commit -m "Your commit message"`
- Push your commit to your repository
  - In IntelliJ: Git → Push
  - In the console: `git push`