# Mcq - Persistency

MCQ Java application with GUI, persistency part.

In this part, the goal is to implement the application **persistency mechanism** (allowing to load/save questionnaires and submissions) from:

- UML class diagram, located in `./part3-Persistency/uml` directory.
- Generated *Javadoc*, located in `./part3-Persistency/docs` directory.
  - **Javadoc comments are not to be rewritten**
- Following guidelines.

**Package structure** is updated:

- `fr.iutvalence.info.but.s2_01.mcq.storage.core` contains storage core classes,
- `fr.iutvalence.info.but.s2_01.mcq.storage.exceptions` contains storage exception classes.

## Table of contents

## Implementation guidelines

The next section details tasks to be performed.

### 1. Project configuration

**Doing it**

1. **Configure project** so that:

- `/part3-Persistency/src` is considered as *source*,
- any other directory is considered as *excluded*.

2. **Copy/paste** `Main` from `/part0-Preamble/src/.../preamble` to `/part3-Persistency/src/.../validation` .

**Checking it**

Check that project builds successfully and that `Main` execution is as expected.

**Committing/Pushing it**

- **Commit** changes with `3-Persistency-Configuration` as message brief.
- **Push** immediately.
- **Check** that remote repository has been updated.

## 2. Parsing methods added in model

(See **UML class diagram** in `/part3-Persistency/uml/Model-ClassDiagram.png` .)

Some **methods have to be added to model classes** in order to **parse object from strings or text streams** :

- `parseQuestionnaireId` in `QuestionnaireId` , to parse an object from a string,
- `parseSubmissionId` in `SubmissionId` , to parse an object from a string,
- `parseQuestion` in `Question` , to parse an object from a text stream,
- `parseQuestionnaire` in `Questionnaire` , to parse an object from a text stream,
- `parseSubmission` in `Submission` , to parse an object from a text stream,

**Doing it, committing/pushing it**

With the help of UML class diagram and *Javadoc*,

1. **Complete** the source code of all parsing methods described above, as well as source code of all missing exceptions (in `part3-Persistency/src/.../exceptions` ).
2. **Commit** changes with `3-Persistency-Parsing` as message brief (and details if issues).
3. **Push** immediately.
4. **Check** that remote repository has been updated.

## 3. Storage interface

(See **UML class diagram** in `/part3-Persistency/uml/Storage-ClassDiagram.png` .)

`Storage` interface defines the **general contract for a questionnaire/submission storage service**, allowing to:

- **load** all available questionnaires or submissions,
- **save** a single or a collection of questionnaires or submissions.

**Doing it, committing/pushing it**

With the help of UML class diagram and *Javadoc*,

1. **Complete** `Storage` source code, as well as `StorageAccessException` (in `./part3-Persistency/src/.../exceptions` ).
2. **Commit** changes with `3-Persistency-Interface` as message brief (and details if issues).

3. **Push** immediately.
4. **Check** that remote repository has been updated.

## 4. File system storage

(See **UML class diagram** in `/part3-Persistency/uml/Storage-ClassDiagram.png`.)

`FileSystemStorage` class is a **storage service implementation using files** to store questionnaires and submissions.

The **directory structure** is depicted below.

```
- (root directory)
  |____ questionnaires
  |     |____ Myself#My questionnaire.qst
  |
  |____ submissions
        |____ A filler-Myself#My questionnaire.sbm
        |____ Serial filler-Myself#My questionnaire.sbm
```

**Questionnaires** are stored in a `questionnaires` sub-directory:

- each questionnaire is stored in a **separate file**
  - **file name** is the text representation of questionnaire id
  - **file extension** is `.qst`
- file contains (see below), in order:
  - title,
  - author name,
  - question count,
  - questions,
    - text,
    - answer count,
    - answers (one per line) ,
    - correct answer id.

```
My questionnaire
Myself
2
What is the answer to life, universe and everything?
4
42
32768
There is no answer
Kamoulox
0
Another question?
4
I don't know
```

```
No
For sure
Maybe
1
```

**Submissions** are stored in a `submissions` sub-directory:

- each submission is stored in a **separate file**
  - **file name** is the text representation of submission id
  - **file extension** is `.sbm`
- file contains (see below), in order:
  - filler name,
  - questionnaire id,
  - answer count,
  - answer ids (one per line),

```
A filler
A filler—Myself#My questionnaire
2
1
3
```

### Doing it

With the help of UML class diagrams and *Javadoc*,

1. **Complete** `FileSystemStorage` source code.
2. **Complete** `StorageMain` application source code, following guidelines given in source code.

### Checking it

Check that `StorageMain` execution output is as below:

```
Adding a questionnaire
Title:  My questionnaire
Author: Myself
2 question(s)
Question 0
? -> What is the answer to life, universe and everything?
0 -> (o) 42
1 -> (x) 32768
2 -> (x) There is no answer
3 -> (x) Kamoulox

Question 1
? -> Another question?
0 -> (x) I don't know
1 -> (o) No
2 -> (x) For sure
3 -> (x) Maybe
```

```
Adding a submission
Filler: A filler
Questionnaire: Myself#My questionnaire
Question 0: 1
Question 1: 3

Adding a submission
Filler: Serial filler
Questionnaire: Myself#My questionnaire
Question 0: 0
Question 1: 2

1 questionnaire(s)
2 submission(s)
Saving all
Replacing manager with a new one using same storage
0 questionnaire(s)
0 submission(s)
Loading all
1 questionnaire(s)
2 submission(s)
Title:  My questionnaire
Author: Myself
2 question(s)
Question 0
? -> What is the answer to life, universe and everything?
0 -> (o) 42
1 -> (x) 32768
2 -> (x) There is no answer
3 -> (x) Kamoulox

Question 1
? -> Another question?
0 -> (x) I don't know
1 -> (o) No
2 -> (x) For sure
3 -> (x) Maybe


Filler: A filler
Questionnaire: A filler-Myself#My questionnaire
Question 0: 1
Question 1: 3

Filler: Serial filler
Questionnaire: Serial filler-Myself#My questionnaire
Question 0: 0
Question 1: 2
```

### Committing/Pushing it

- **Commit** changes with `3-Persistency-FileSystemStorage` as message brief.
    - If checking step has failed, give details about issues

- **Push** immediately.
- **Check** that remote repository has been updated.