Projet Ocaml - Java ILU1

Décembre 2022

Ce document a pour vocation de donner quelques détails supplémentaires pour l'architecture de votre projet, sur les conseils de rendu et les critères d'évaluation dont nous tiendrons compte.
Soyez bien conscient que chaque correcteur ou correctrice aura beaucoup de projets à évaluer. Il ne leur sera pas possible de perdre du temps à chercher trop d'informations dans votre rendu si elles ne sont pas présentées comme espérées.

Conseils d'architecture

Architecture globale du projet

Comme nous l'avons dit précédemment, la partie Java doit servir pour réaliser :

- l'interface textuelle (par exemple à la manière de la machine à café vue en cours et en TP GL),
- la gestion des objets de votre domaine métier, le sujet de votre projet personnel
- éventuellement, un objet médiateur entre l'interface textuelle et les objets du domaine métier pour diminuer le couplage entre les deux
- éventuellement, un objet spécifique qui réalise les appels au programme OCaml pour réaliser la persistance des données, leur enregistrement dans un ou des fichiers `csv`.

La partie OCaml doit servir à réaliser la persistance des données avec, par exemple, des fonctions CRUD :

- C pour Creation, l'enregistrement initiale d'une donnée dans un fichier `csv`
- R pour Read, la lecture d'une donnée à partir d'un fichier `csv`
- U pour Update, la modification ou la mise à jour d'une donnée déjà présente dans un fichier `csv`
- D pour Delete, la suppression d'une donnée présente dans un fichier `csv`.

En créant un projet Java avec Eclipse que vous complétez par la partie Ocaml, cela pourrait donner la structure suivante de votre projet :

- un répertoire `src` pour les classes Java dans lequel se trouvent vos paquetages/répertoires Java.
- un répertoire `bin` pour les fichiers `.class`
- un répertoire `ocaml` contenant tous les fichiers pour la persistance

Dans tous les cas, quelle que soit la structure de projet que vous choisissez, nous vous conseillons de la décrire en fin de fichier `README.md` de votre projet ainsi que dans la vidéo. Ainsi, vos enseignants de TP ne seront pas perdus dans l'ensemble des fichiers de votre projet.

Architecture partie Ocaml

Le projet Github suivant déjà diffusé (https://github.com/UPS-ILU/ups-ilu1-ocaml-examples) vous donne quelques éléments de base pour la gestion de la persistance :

- un répertoire `examples`

- un répertoire `src` contenant :
    - un fichier `main.ml` donnant l'exemple de deux fonctionnalités activées à partir de la ligne de commande
    - un fichier `parse_cli.ml` à reprendre tel quel pour extraire les arguments de la ligne de commande (avec d'abord les options, le séparateur d'options puis les autres arguments, supposés non optionnels)
    - un fichier `libunix.ml` pouvant servir de modèle pour la gestion des chemins en Ocaml
    - un fichier `libcsv.ml` à reprendre tel quel pour gérer les fichiers `csv` comme des listes de type `string list list`.
    - un fichier (ici `sum_avg.ml`) contenant les fonctions utiles au programme
- un fichier `Makefile` définissant quels sont les fichiers `.ml` à inclure dans la fabrication du programme exécutable (obtenu par la commande `make`)
- d'autres fichiers à reprendre tels quels

Pour organiser la partie OCaml de votre projet, se trouvant donc par exemple dans le répertoire `ocaml`, vous pouvez adopter la structure suivante :

- un sous-répertoire `data` contenant les fichiers `csv` permettant de stocker les données nécessaires au domaine métier spécifique de votre projet
- un sous-répertoire `src` tel que ci-dessus dans lequel vous avez ajouté vos propres fichiers `.ml` comme par exemple un fichier définissant les opérations CRUD
- les autres fichiers nécessaires dont au moins le fichier `Makefile` mis à jour pour votre projet.

Dans tous les cas, quelle que soit la structure de projet que vous choisissez, nous vous conseillons de la décrire en fin de fichier `README.md` de votre projet ainsi que dans la vidéo. Ainsi, vos enseignants de TP ne seront pas perdus dans l'ensemble des fichiers de votre projet.

Architecture partie Java

Elle regroupe la partie interface textuelle nécessaire pour présenter un éventuel menu et des éventuels sous-menus, les objets du domaine métier que vous gérez spécifiquement dans votre projet, les objets nécessaires au découplage menu/domaine ou domaine/persistance et le main. Cela peut être découpé en plusieurs paquetages inclus dans `src` comme, par exemple :

- `menu` pour l'interface textuelle et l'objet (qu'on appelle *controller*) qui interconnecte l'interface et les autres objets de l'application
- `domain` qui contient toutes vos classes des objets du domaine métier
- `dao` (pour Data Access Object) qui contient les classes des objets qui appellent le programme Ocaml pour la persistance
- `main` qui contient le programme principal de votre partie Java

Dans tous les cas, quelle que soit la structure de projet que vous choisissez, nous vous conseillons de la décrire en fin de fichier `README.md` de votre projet ainsi que dans la vidéo. Ainsi, vos enseignants de TP ne seront pas perdus dans l'ensemble des fichiers de votre projet.

Couplage Java - Ocaml

Une façon simple (mais pas tout à fait réaliste) d'obtenir la persistance et de coupler la partie Java et Ocaml est de lire l'ensemble des données stockées dans les fichiers `csv` en début

de programme (ou de proposer cela comme opération du menu) et de stocker à nouveau toutes les informations contenues dans les objets du domaine métier juste avant que le programme se termine.

- Pour cela, un moyen peut être d'écrire un programme OCaml qui se charge de la persistance. Dans l'exemple de projet proposé (https://github.com/UPS-ILU/ilu1-project-example), ce programme a été nommé `persist` (valeur de `RESULT` à décommenter dans le `Makefile`). Le choix a également été fait d'utiliser une option (`-C`, `-R`, `-U` ou `-D`) pour l'opération CRUD visée et les valeurs optionnelles de l'opération avant le marqueur de fin d'option ("`--`"). Enfin, le nom de fichier utilisé cible quelles sont les données affectées par l'opération CRUD. Ces éléments sont des choix proposés ; vous pouvez choisir d'autres conventions.
- Étant donnés ces choix, un fichier `crud_csv.ml` peut être défini qui précise les implantations des fonctions CRUD.
- Ainsi fait, la classe Java `DataAccessObject` qui s'occupe d'appeler le programme Ocaml pour la persistance définit la commande à appeler et adapte l'exemple de `MainProcess.java` pour appeler et récupérer les résultats des opérations CRUD.

Rendre votre projet

Pour rendre votre projet, nous allons procéder à l'aide d'un outil spécifique de GitHub qui vous permet de créer un dépôt individuel privé auquel chaque enseignant de TP pourra automatiquement accéder. Il s'agit de GitHub ClassRoom. Cela se passe en deux étapes comprenant la création du dépôt avec ClassRoom et la mise à jour des informations de rendu.

Création du dépôt

Pour cela, vous devez :

- accéder à l'adresse suivante qui, dans le jargon de GitHub, est un devoir (*assignment* in english) : https://classroom.github.com/a/TCmJ3kBX
- accepter les permissions demandées (principalement pour se connecter avec votre compte GitHub)
- accepter le devoir. GitHub configure alors votre dépôt (ce qui met un peu de temps) et vous demande de mettre à jour la page web. Il vous donne alors l'URL de ce dépôt qui apparaît dans la "classroom" UPS-ILU

Vous êtes maintenant prêt à travailler comme s'il s'agissait d'un dépôt que vous avez vous-même créé. Vous pouvez cloner ce dépôt distant sur un ou plusieurs dépôts locaux, travailler sur votre projet et synchroniser votre dépôt distant régulièrement.
Ce dépôt initial contient une structure de projet telle que nous l'avons conseillée dans la première partie. Vous n'êtes pas obligés de la respecter. Vous pouvez supprimer cette structure et définir la vôtre. **Dans tous les cas, quelle que soit la structure de projet que vous choisissez, nous vous conseillons de la décrire en fin de fichier `README.md` de votre projet.** Ainsi, vos enseignants de TP ne seront pas perdus dans l'ensemble des fichiers de votre projet.

Mise à jour du rendu

Pour permettre l'évaluation de votre projet et participer à l'épreuve de CC4, **vous devez impérativement effectuer toutes les opérations suivantes**, sans quoi votre note de CC4 vaudra 0 :

- Editer le fichier README.md pour y enregistrer les informations suivantes :
    - ajoutez votre nom sur la fin de ligne 3
    - ajoutez votre prénom sur la fin de ligne 4
    - remplacez `` `[ ]` `` par `` `[x]` `` pour la ligne correspondant à votre groupe de TP
    - donnez le lien vers l'adresse d'hébergement de la vidéo de votre présentation de projet (par exemple hébergée sur Youtube ou tout autre service d'hébergement de vidéo) en fin de ligne 22

Encore une fois, sans ces 4 informations fournies par le moyen ci-dessus, vous ne serez pas évalué pour le CC4.

N'oubliez pas d'historiser et de synchroniser cette modification (git add, git commit, git push).
N'oubliez pas non plus d'historiser et de synchroniser régulièrement vos avancées sur le projet si vous ne l'avez pas encore terminé. Cela permettra à vos enseignants de TP de mesurer vos progrès.

Critères d'évaluation

Pour rappel, vous avez sur Moodle, dans chacun des onglets 'GL', 'POO' et 'PF', la liste des compétences attendues dans chacune des parties de ILU1 et que vous devez démontrer dans le projet. Cela fait 13 tâches pour 'GL', 25 pour 'POO' et 13 pour 'PF' qui peuvent s'appliquer à votre projet.
Probablement que le contexte particulier de votre sujet ne vous permettra pas facilement de démontrer de ces capacités mais vos correcteurs auront d'autant plus de faciliter à vous donner beaucoup de points que vous avez pu démontrer de nombreuses capacités.
En plus de ces capacités, il doit être possible à l'un de vos correcteurs **de télécharger votre projet et de l'exécuter sur sa machine**. Faites très attention à ce point.
Dans la vidéo, vous devez présenter :

- un rappel de votre sujet de projet et des fonctionnalités prévues
- la structure de votre projet, même si elle reprend celle proposée
- le détail de la partie OCaml en présentant :
    - son organisation structurelle en répertoire et fichiers
    - le détail des fichiers csv utilisés
    - le détail du programme principal (structures de données, fonctions, types)
    - le détail des sous-parties que vous avez écrites (structures de données, fonctions, types)
    - un exemple d'usage (ou plusieurs) du programme en ligne de commande et des résultats obtenus
- le détail de la partie Java en présentant :
    - son organisation structurelle en répertoire et fichiers
    - le détail du programme principal (objets créés et leur rôle, les éléments algorithmiques principaux)
    - le détail des classes du domaine métier spécifique à votre projet (rôle des objets, attributs, constructeurs et méthodes pertinents)
    - le détail d'autres classes ajoutées et mettant en valeur vos compétences
- le détail de vos compétences en GL :
    - votre utilisation des dépôts
    - la qualité du code obtenu ou les efforts ponctuels faits en ce sens

    - les activités de programmation facilitant la maintenance
    - les scénarios de test que vous avez mis en oeuvre

Cela fait certainement beaucoup de choses à présenter en 10 minutes. Cela fait partie de l'épreuve de la vidéo que de savoir synthétiser ce qui est le plus important à présenter. **Ne dépassez pas les 10 minutes de façon outrageuse au risque que vos correcteurs ne regardent pas la fin.**