

# Aide Correction

Delhomme Fabien

26 novembre 2022

## 1 Objectif

Avoir une interface qui permette de m'aider pour la correction de contrôle.

J'aimerais notamment :

- comptage des points
- statistiques de réussite
- grille de réussite pour chaque question (“propreté”, “justification”)
- Détection automatique d’erreur systématique chez les élèves, pour les notifier efficacement.
- exporter en csv pour analyser les données dans un tableur.

## 2 Étape 1

### 2.1 On définit l'ensemble des élèves dans une variable globale

Nous avons donc le code lisp suivant :

---

```
(defparameter *classe* '((ABDOUNE Maxence)
  (AGAMAKOU Axel)
  (ATTYE Charlotte)
  (BEMBA Séréna)
  (BRESSY Jade)
  (CALEJO Alex)
  (CHARLES Maeva)
  (CORREIA Joana)
  (DE-LUISE Enzo)
  (DE-PERCIN Kristopher)
  (FOURDRINIER Thomas)
  (GALLOY Ronan)
  (GARRUZZO Rafael)
  (GERMAIN Willem)
  (GORSSE Clara)
  (LABAYE Elodie)
  (LEOCADIE Enzo)
  (LEVI-VALENSI Margaux)
  (MADAoui Lyma)
  (MAHROUG Walid)
  (MARTIN Emma)
  (MOUROU Aya)
  (POUMAROUX Alyssia)
  (PUPILLI Lola)
  (RACON Kassidi)
  (RAMTANI Cerina)
  (RENZI Ambre)
  (SAHNOUN Halima)
  (SALEM Ibrahim)
  (SALLENAVE Helene)
  (SLIMANI Shirine)
  (TRAN Matthieu)
  (TRAORE Saïd)
  (TSHISELEKA MUTETENU Justino)
  (VAREILLE Mathieu)
  (ZEKA Kerene)))
```

---

\*CLASSE\*

Avoir des cons-listes ou des listes ne changent pas grand chose à mon avis. Autant faire au plus simple avec des listes.

## 2.2 Exemple

On peut parcourir l'ensemble des élèves par :

---

```
(loop for (nom prenom) in *classe* do
  (print nom)
  (princ prenom))

(format t "~%La classe comporte ~S élèves" (length *classe*))
```

---

ABDOUNE MAXENCE  
AGAMAKOU AXEL  
ATTYE CHARLOTTE  
BEMBA SÉRÉNA  
BRESSY JADE  
CALEJO ALEX  
CHARLES MAEVA  
CORREIA JOANA  
DE-LUISE ENZO  
DE-PERCIN KRISTOPHER  
FOURDRINIER THOMAS  
GALLOY RONAN  
GARRUZZO RAFAEL  
GERMAIN WILLEM  
GORSSE CLARA  
LABAYE ELODIE  
LEOCADIE ENZO  
LEVI-VALENSI MARGAUX  
MADAOUY LYMA  
MAHROUG WALID  
MARTIN EMMA  
MOUROU AYA  
POUMAROUX ALYSSIA  
PUPILLI LOLA  
RACON KASSIDI  
RAMTANI CERINA  
RENZI AMBRE  
SAHNOUN HALIMA  
SALEM IBRAHIM  
SALLENAVE HELENE  
SLIMANI SHIRINE  
TRAN MATTHIEU  
TRAORE SAÏD  
TSHISELEKA MUTETENU  
VAREILLE MATHIEU  
ZEKA KERENE  
La classe comporte 36 élèves

### 3 Mise en forme des contrôle dans lisp

#### 3.1 Définition d'un contrôle.

Un contrôle est une évaluation qui comporte un certain nombre de **questions**. Chaque question vérifie que l'élève sait faire un certain nombre de **compétences**. Chaque compétences acquises rapporte un certain nombre de **points**.

Plusieurs pistes pour implémenter un contrôle dans lisp :

- La programmation orientée objet.
- Une liste de liste. On peut utiliser des clé voire des tableaux de hashage.

La solution la plus souple me paraît être la deuxième option. Je ne connais pas assez les objets dans lisp pour m'y fier.

### 3.2 Programmer un exemple de contrôle.

On définit au passage une fonction qui ajoute un contrôle à la liste des contrôles.

Définir une fonction qui ajoute un exercice à un contrôle est plus délicate, puisque techniquement le contrôle en question n'est pas une variable globale. Je repousse ça à plus tard.

La variable `*contrôle-1*` contient deux exercices, le premier contient deux questions, et le suivant qu'une seule.

---

```
(defparameter *contrôles* '())

(defparameter *contrôle-1* '(
  (
    (:enonce "Combien vaut 1 + 1 ?" :competence (:calcul 1
  ↪ :presentation 0.5))
    (:enonce "Montrer que  $\sqrt{2}$  est irrationnel."
  ↪ :competence (:demonstration 2 :proprete 1 :logique 1)))
  (
    (:enonce "Démontre qu'une somme de vecteur est bien
  ↪ définie" :competence (:demonstration 2))))))

(defun ajouter-contrôle (contrôle)
  (append contrôle *contrôles*))

;; (defun ajouter-exercice (exercice contrôle)
;;   (setf contrôle (append exercice contrôle)))

(defun reset-contrôles ()
  (setf *contrôles* '()))

(defparameter *exercice-test* '(:enonce "Citer une identité remarquable" :competence
  ↪ (:cours 1)))

(ajouter-contrôle *contrôle-1*)
```

---

### 3.3 Accéder aux caractéristiques d'un contrôle

On souhaite accéder aux nombres de points de la première question. On définit deux fonctions qui permettent d'accéder aux compétences d'une question d'un contrôle, et une autre qui retourne le nombre de points associés à cette question.

On avait débuté par faire des accès «absolus» aux questions, mais ça sert à rien. Les fonctions acceptent une liste qu'elles comprennent comme une question. Beaucoup plus souple, et pas besoin de traîner des indices partout. On voit ça notamment dans la ligne 23.

---

```

1 ;; (defun liste-competence-point-question (controle exercice question)
2 ;;   (let ((question-selectionne (nth question (nth exercice controle))))
3 ;;     (getf question-selectionne :competence)))
4
5 (defun liste-competences-points-question (question)
6   (getf question :competence))
7
8 ;; (defun nombre-point-question (controle exercice question)
9 ;;   (let ((ensemble-competence (liste-competence-point-question controle exercice
10 ;;     question)))
11 ;;     (reduce '+ (remove-if-not 'numberp ensemble-competence))))
12
13 (defun nombre-points-question (question)
14   "En donnant une question, retourne la somme des points associés aux compétences."
15   (reduce '+ (remove-if-not 'numberp question)))
16
17 (defun nombre-points-controle (controle)
18   (loop
19     for exercice in controle
20     sum (loop
21       for question in exercice
22       sum (nombre-points-question (liste-competences-points-question question)))))
23
24 ;(nombre-point-question *controle-1* 0 0)
25 (nombre-points-controle *controle-1*)

```

---

7.5

### 3.4 Coder une interface pour ajouter un nouveau contrôle

## 4 Macro

### 4.1 Parcourir toutes les questions d'un contrôle

Une idée à peaufiner pour plus tard.

On notera qu'il peut y avoir des soucis à ne pas avoir utilisé gensym pour le nom d'une variable (ici exercice) qui parcourt les exercices.

Pour l'instant le code ne fonctionne pas.

---

```

(defmacro pour (question controle &body)
  `(loop
    for exercice in ,controle
    do
      (loop
        for ,question in exercice
        do
          &body)))

```

---

## 5 Corriger un contrôle

### 5.1 KILL Chaque élève est assigné à une copie du contrôle

Pour l'instant je ne sais pas créer une copie du modèle contrôle qui ne contiennent que des compétences initialisées à 0 sans modifier la liste des compétences de chaque question.

J'ai testé de définir une liste des compétences pour chercher parmi elles, mais dans ce cas on crée des propriétés pour celle qui n'y était pas, ce n'est pas ce que l'on veut.

Une autre idée : une plist est une liste. Donc, on peut tout simplement tout parcourir et mettre tous les nombres à 0.

Je suis en train d'implémenter cette idée dans la fonction `reset-nombre-zero` mais c'est pas si évident.

Le plan après c'est de :

- On parcourt chaque élève,
- On crée une copie blanche
- On parcourt la copie et on peut éditer les points
- On fait le résumé de cet élève et tout est enregistré à chaque étape pour éviter les pertes.

---

```
;; Il faudrait une copie neuve du contrôle  
;; qui représente la copie de l'élève.
```

```
(defparameter *liste-competences* '(:calcul :raisonement :logique) "Liste des  
↳ compétences d'une question.")  
  
(defun reset-nombre-zero (liste)  
  (let ((element (car liste)) (suite-vide? (cdr liste)))  
    (cond  
      ((numberp element) (progn  
                           (setf element 0)  
                           (unless suite-vide? (reset-nombre-zero (cdr liste)))))  
      ((listp element) (reset-nombre-zero (car element))))))  
  
(defun reset-notes (question)  
  (loop)  
  for competence in *liste-competences*  
  do  
  (let ((champ-competence (getf question :competence)))  
    (when (getf champ-competence competence) (setf (getf champ-competence competence)  
↳ 0))  
    (print champ-competence))  
  question)  
  
(defun copie-eleve (controle)  
  "Donne une copie vierge du contrôle, prête à être corrigée"  
  ;; C'est là que les objets aurait pu être utile...  
  (loop)  
  for exercice in controle  
  collect (loop  
           for question in exercice  
           (reset-notes (copy-list question))))
```

---

COPIE-ELEVE

## 5.2 DONE On reconstruit le contrôle pour chaque élève

C'est bon, ça marche ENFIN

---

```
(defun copie-control-competece-0 (controle)
  (when controle
    (cond
      ((listp controle) (cons (copie-control-competece-0 (car controle))
                              ↪ (copie-control-competece-0 (cdr controle))))
      ((numberp controle) 0)
      (t controle))))
```

---

COPIE-CONTROLE-COMPETENCE-0

## 6 Commandes pour faciliter la correction

### 6.1 En vrac

- search pour chercher un élève dans la liste, et le sélectionner pour modifier ses notes
- stat pour sortir les statistiques par questions du contrôle

\*