LANGAGES DE PROGRAMMATION
IFT 3000 NRC 15997
HIVER 2022

Travail pratique 2 (individuel)

À remettre, par le portail du cours avant 17h00 le mardi 19 avril 2022

1 Objectif

Nous vous demandons dans ce travail pratique d'implanter **le système d'indicateurs** de développement des pays du TP1 en utilisant le paradigme orienté objet. En effet, il s'agit de reprendre le travail demandé pour le TP1, mais en ajoutant d'autres fonctionnalités et en utilisant également une interface graphique.

2 Travail à faire

Le TP consiste en l'implantation de la structure (*module*) Tp2h22 (voir fichier "TP2-H2022.ml") qui a la signature (*module type*) TP2H22 (voir fichier : "TP2-SIG-H2022.mli"). **Il est à noter qu'il ne faut pas modifier la signature TP2H22.** Dans cette signature, il existe 4 classes (*indicateur*, *sysindicateurs*, *sysind_education* et *app_sysindicateurs*). La classe *sysind_education* hérite de la classe *sysindicateurs*. La classe *app_sysindicateurs* permet de démarrer l'application pouvant chercher des indicateurs pour un pays ou une région sélectionné soit dans la console soit dans une fenêtre graphique. Vous pouvez ajouter des fonctions ou des méthodes privées dans le fichier "TP2-H2022.ml". Par contre, votre travail consiste essentiellement à implanter les 10 méthodes suivantes :

1. method afficher_indicateur : unit **(6 points)**
2. method retourner_indicateur : string * string → indicateur **(6 points)**
3. method supprimer_indicateur : string * string → unit **(6 points)**
4. method supprimer_liste_indicateurs : (string * string) list → unit **(6 points)**
5. method afficher_indicateurspays : indicateur list → unit **(6 points)**
6. method afficher_valeur_indicateur : indicateur → int → unit **(7 points)**
7. method retourner_indicateurs_pays : string → indicateur list **(7 points)**

8. method sauvegarder_indicateur : indicateur → out_channel → unit **(6 points)**
9. method lancer_systeme_indicateurs : unit **(15 points)**
10. method lancer_interface_sindicateurs : unit **(25 points)**

Vous connaissez normalement ce que fait la majorité de ces méthodes (veuillez consulter l'énoncé du TP1). Cependant, voici des explications supplémentaires surtout pour les nouvelles méthodes à implanter :

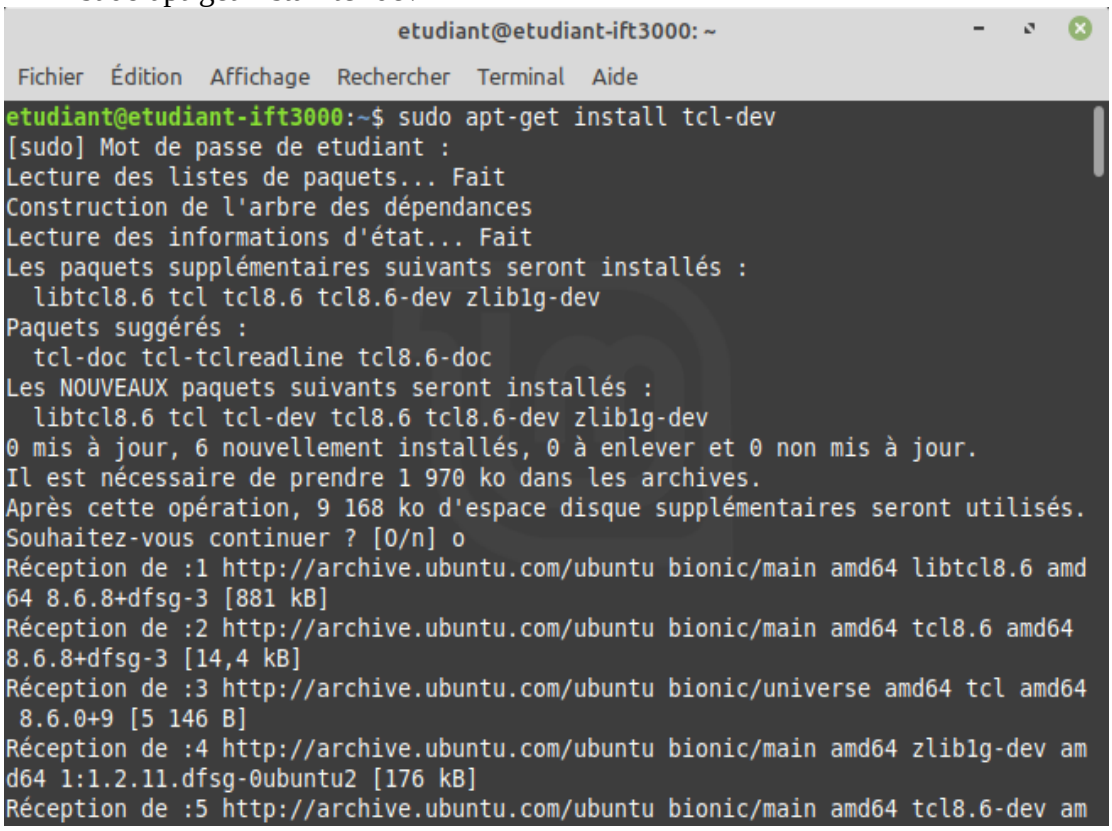
- **sauvegarder_indicateur : indicateur → out_channel → unit** est une méthode qui prend un indicateur (un objet) ainsi qu'un flux déjà ouvert en mode sortie (*out_channel*) et elle écrit les informations de cet indicateur sur ce flux. Cette méthode ne doit pas fermer le flux. C'est à la méthode appelante de le faire. Il s'agit en fait de la méthode lancer_systeme_indicateurs. Nous vous fournissons d'ailleurs la méthode privée retourner_chi() permettant de retourner une chaîne de caractères qui va être utilisée dans la sauvegarde des informations de l'indicateur en questions. Voici un exemple de code permettant d'écrire dans un fichier texte :

  ```
  # let file = open_out "test.txt";;
  val file : out_channel = <abstr>
  # let chaine = "Ceci est un test";;
  val chaine : string = "Ceci est un test"
  # output_string file chaine;;
  - : unit = ()
  # close_out file;;
  - : unit = ()
  ```

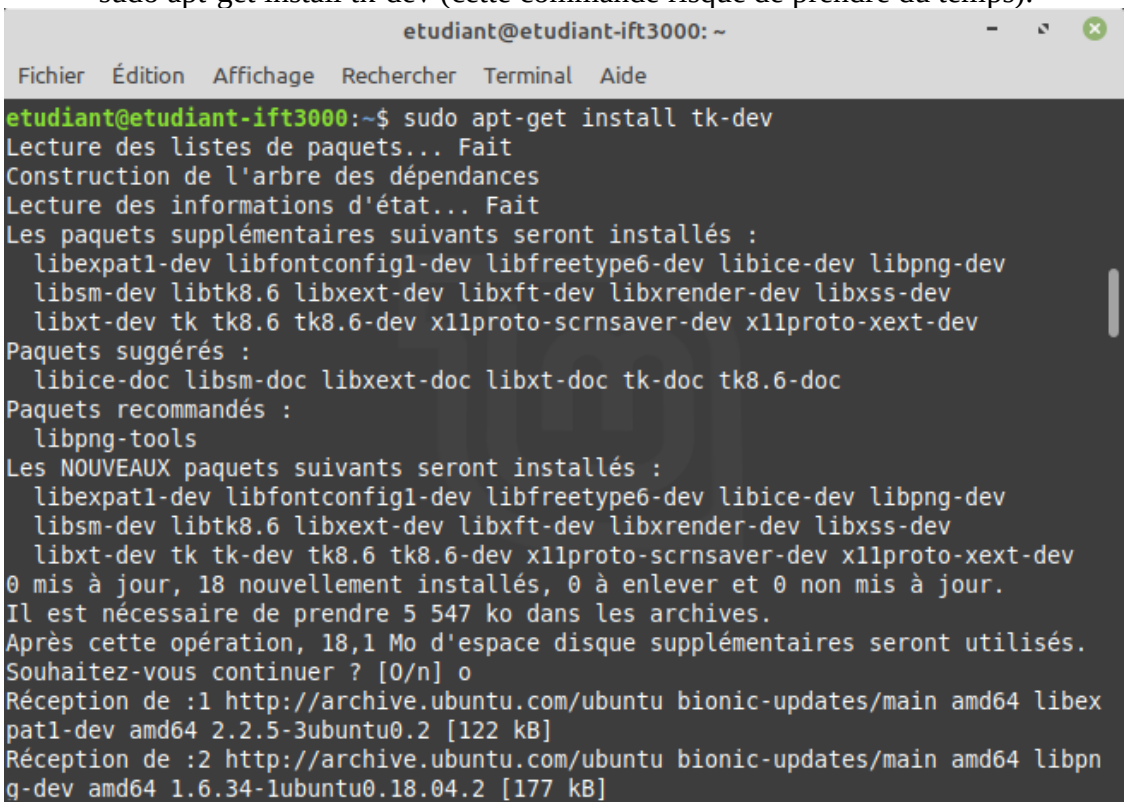
- **lancer_systeme_indicateurs: unit** Vous connaissez normalement l'utilité de cette méthode. Elle permet de charger les données en utilisant le fichier csv. Elle doit chercher les informations d'un pays en particulier selon le choix de l'utilisateur. Cette méthode peut également maintenant sauvegarder le résultat de la recherche dans un fichier texte "Resultat.txt". Veuillez d'ailleurs consulter le fichier "je.ml" pour voir des exemples d'exécution de cette méthode. Il est à noter que cette méthode est lancée automatiquement lors de l'instanciation d'un objet de la classe *app_ sysindicateurs* si l'utilisateur fournit à new un booléen égal à false (voir "je.ml").

- **lancer_interface_sindicateurs : unit** est une méthode qui affiche une interface graphique en utilisant la librairie lablTk. Il faut commencer par contre par installer cette librairie sur la VM du cours. Il suffit d'ouvrir un terminal et entrer les trois commandes suivantes (mot de passe : ift3000):

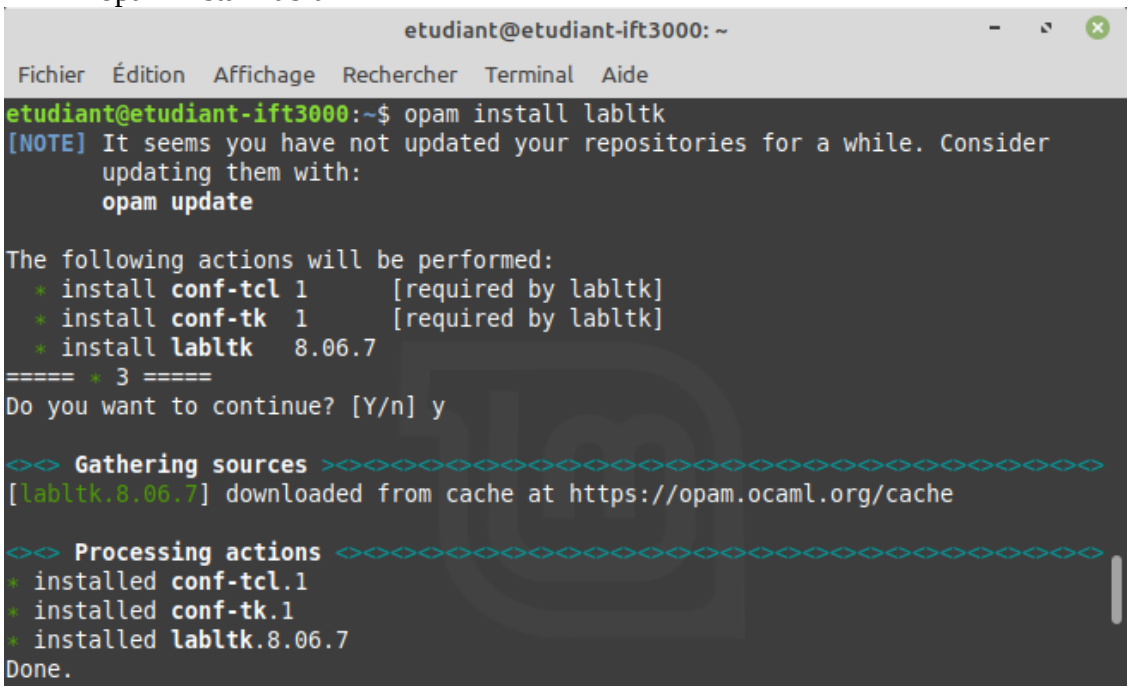
sudo apt-get install tcl-dev



sudo apt-get install tk-dev (cette commande risque de prendre du temps).

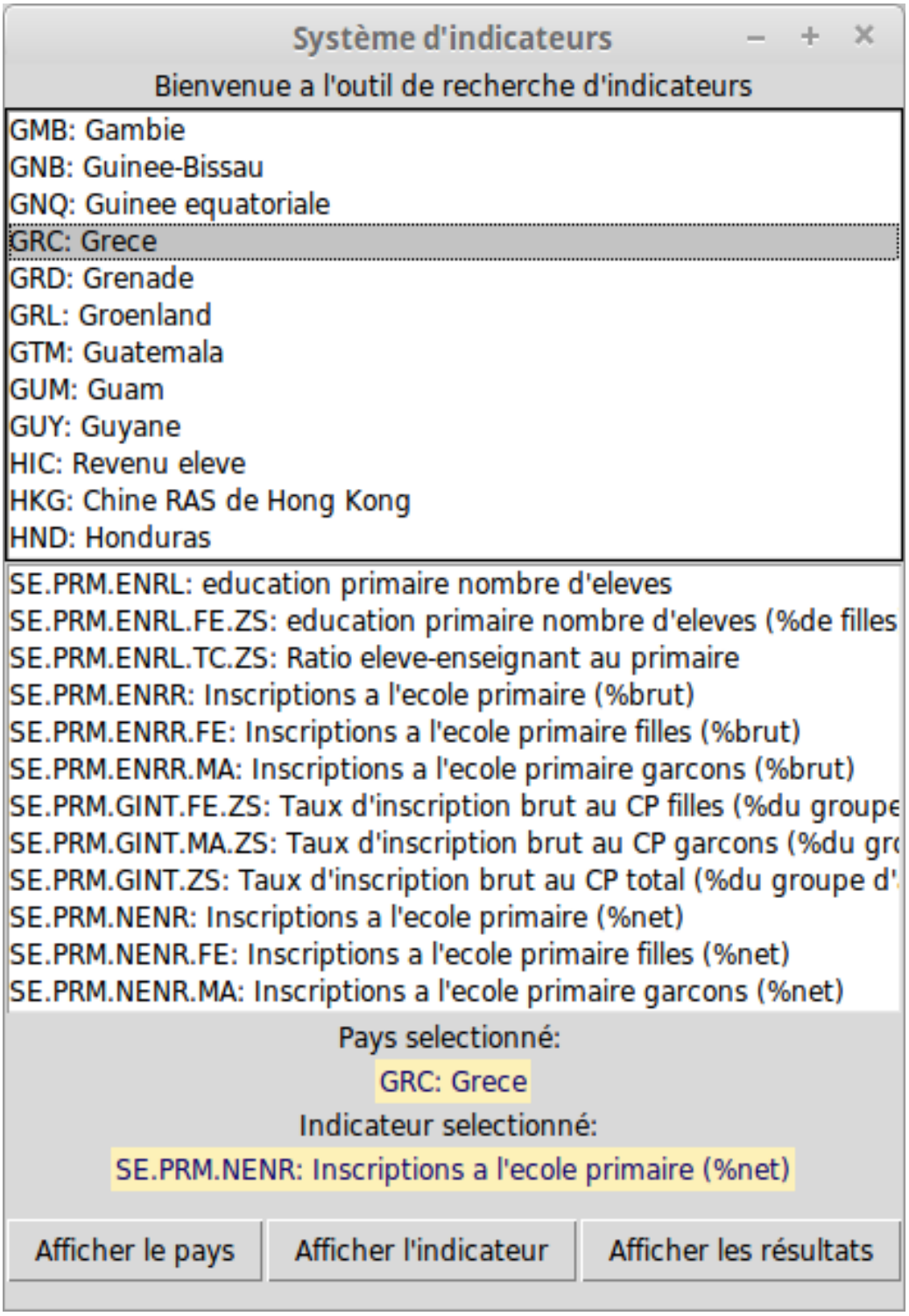


opam install lablтk

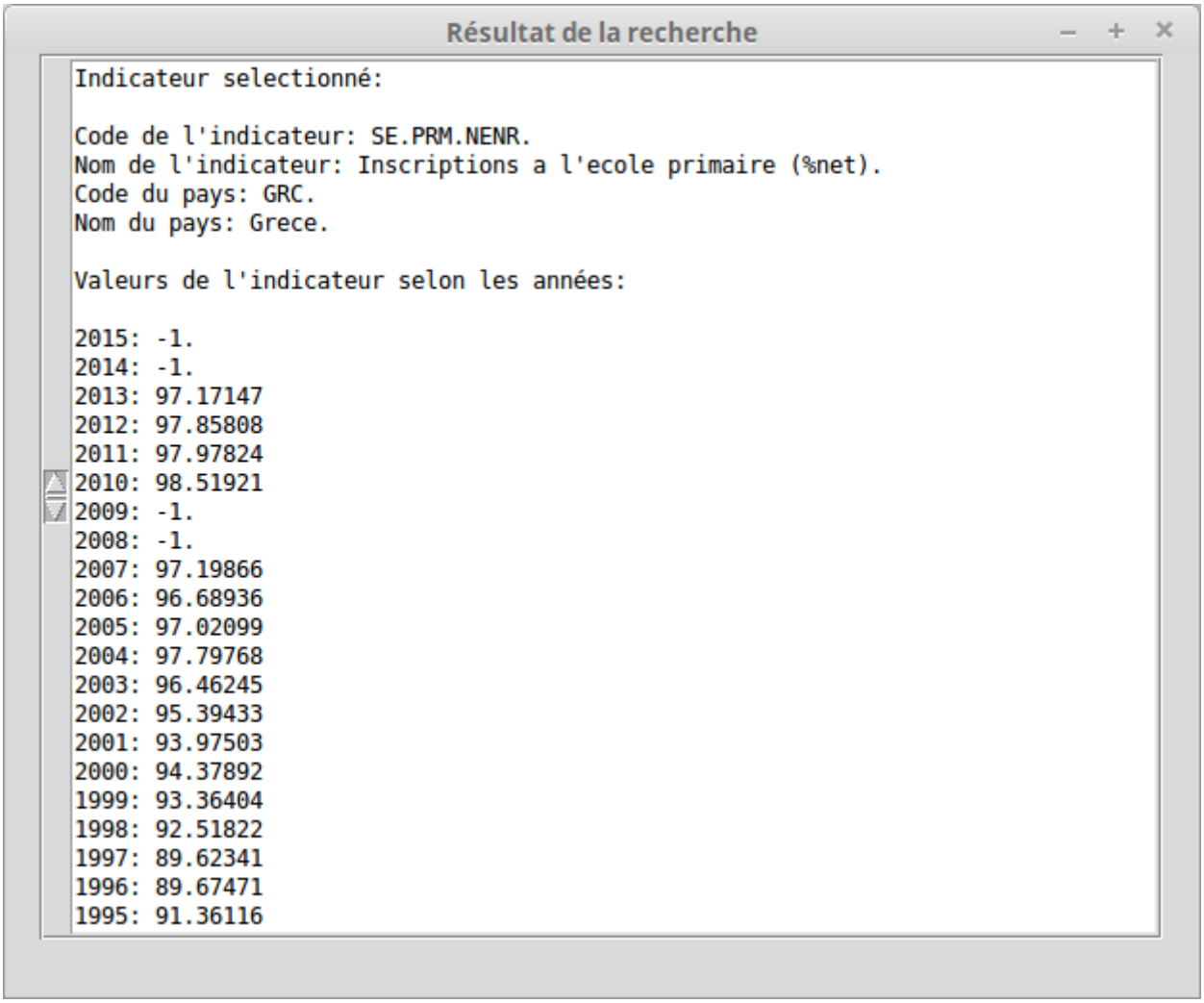


Afin d'accéder par la suite à la librairie dans OCaml, il faut utiliser les commandes suivantes :
#directory "/home/etudiant/.opam/4.08.0/lib/lablтk";;
#load "lablтk.cma";;
open Tk;;

Cependant, nous vous fournissons un fichier **utiles.ml** permettant de charger cette librairie, mais aussi contenant des fonctions et des modules utiles pour votre TP. Il est conseillé de consulter par exemple ce lien pour savoir comment utiliser la librairie lablтk afin d'ajouter et manipuler des widgets dans une fenêtre graphique :
https://who.rocq.inria.fr/Francois.Thomasset/Lablтk/Tutoriel_FT/
Voici le type de fenêtre (le minimum) que nous vous conseillons d'afficher, mais vous pouvez produire si vous le voulez une interface plus conviviale:



L'utilisateur doit dans ce cas sélectionner dans des Listboxes un pays ainsi qu'un indicateur. Il clique par la suite sur le bouton « afficher les résultats » pour avoir la fenêtre suivante :



Vous pouvez utiliser la méthode privée fournie retourner_chi() afin d'afficher les informations de l'indicateur sélectionné dans cette fenêtre. De plus et afin de vous aider, nous vous fournissons le début du code de la méthode lancer_interface_sindicateurs() qui affiche la première fenêtre et c'est à vous de la compléter.

3 Démarche à suivre

Le fichier "TP2-H2022.ml" a été conçu pour que vous puissiez interpréter le code même si les méthodes demandées ne sont pas encore implantées. En fait, nous faisons la simulation de la valeur de retour de chaque méthode. Vous pouvez donc faire ceci :

use "TP2-H2022.ml " ; ;

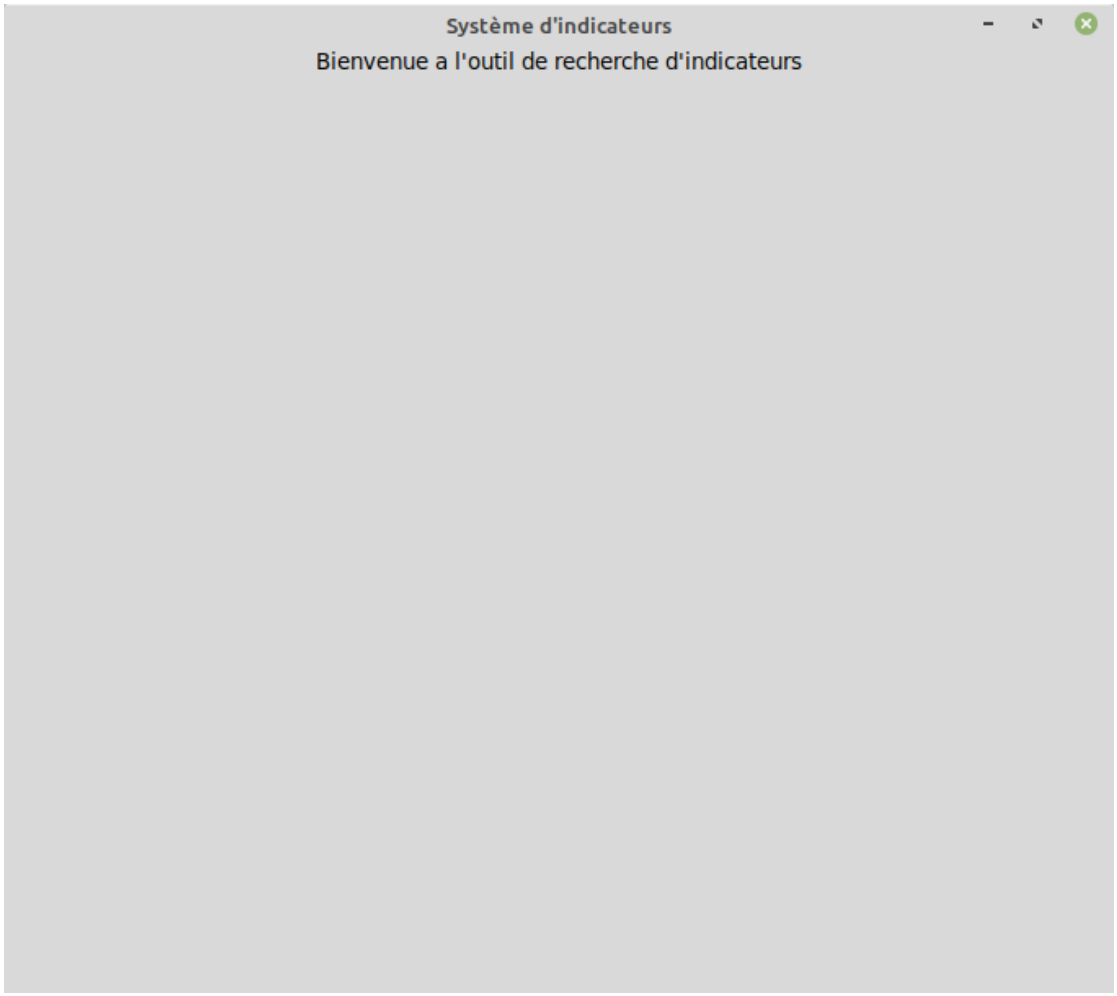
Vous allez avoir un module qui respecte la signature de TP :

…

module Tp2h22 : TP2H22

Vous pouvez donc ouvrir ce module :
open Tp2h22;;

Vous pouvez même lancer ce code pour afficher une fenêtre vide :
let asi = new app_sysindicateurs "data.csv" true;; (* cette méthode est lancée automatiquement lors de l'instanciation d'un objet de la classe *app_ sysindicateurs* si l'utilisateur fournit à new un booléen égal à true. *)



Par la suite, la démarche à suivre est la suivante :

- Utiliser un éditeur comme Emacs, Eclipse ou Sublime Text, pour compléter l'implantation des méthodes des différentes classes proposées.
- Une fois le module Tp2h22 complété, il vous sera alors possible de faire les tests en utilisant le fichier fourni "je.ml". Dans ce fichier, il y a les résultats attendus. Vous devez donc vérifier que vous obtiendrez exactement ces mêmes résultats. Pour la gestion des erreurs, vous pouvez utiliser la fonction **failwith** (string → 'a). D'ailleurs, à la fin du fichier "je.ml", on utilise **try … with** pour vérifier si l'exception **failure** a été levée.

– Il faut donc mettre tous les fichiers dans un même répertoire et de charger le fichier "je.ml" avec la commande : **#use "je.ml";; (**il ne faut pas oublier le # avec la commande use).

4 À remettre

Vous devez rendre un fichier .zip comportant **uniquement** les fichiers suivants :

- Le fichier « TP2-H2022.ml » complété.
- Les fichiers « TP2-SIG-H2022.mli » et « je.ml » non modifiés.
- Le fichier « utiles.ml » (vous pouvez le modifier si vous voulez ajouter d'autres fonctions).
- Le fichier « NoteTp2.xlsx » (un fichier Excel contenant le barème de correction. Il faut juste ajouter votre nom et matricule sur la première ligne).

Le nom du .zip doit respecter le format suivant : TP2-Matricule.zip. Nous vous rappelons qu'il est important de faire la remise par voie électronique uniquement en vous connectant à: http://monportail.ulaval.ca/ (aucun travail envoyé par courriel n'est accepté). Il est toujours possible de faire plusieurs remises pour le même travail. Vous pouvez donc remettre votre travail plus tôt afin d'être sûr qu'il a été remis en temps, et remettre par la suite une version corrigée avant l'heure limite de remise. De plus, il est de votre responsabilité de vérifier après la remise que vous nous avez envoyé les bons fichiers (**non vides et non corrompus**), sinon vous pouvez avoir un zéro.

Attention ! Il est **formellement interdit de partager ou publier** le code du TP ou votre solution sur le Web avant ou après la remise du TP, car c'est une source évidente de plagiat et vous risquez donc d'être pénalisé. De plus, **tout travail remis en retard se verra pénalisé de -25% de la note**. Le retard débute dès la limite de remise dépassée (dès la première minute). Un retard excédant une journée provoquera le rejet du travail pour la correction et la note de 0 pour ce travail.

Remarques : Il est à noter que **10 points** sont donnés pour le respect et la qualité des biens livrables ainsi que pour la structure générale du code (commentaires, indentation, compilation sans warnings, etc.). De plus, votre code doit absolument pouvoir être exécuté sans erreurs sur la machine virtuelle du cours. Si vous ne testez pas suffisamment votre travail, il risque de provoquer des erreurs à l'exécution lors de la correction. La moindre erreur d'exécution rend la correction extrêmement difficile et vous en serez donc fortement pénalisés.

Bon travail !