

Situation d'Apprentissage et D'Evaluation 1.01

Semaine 1 – 19 octobre 2021

Objectif pédagogique : mettre en place les outils nécessaires à la réussite du SAE

Modalité : travail en binôme

Nombre de séances de TP : 2*1h30

Nombre de dépôts sur Git : 4 minimum

- Le code de *GestionNotes*
- Le code des deux programmes (libres) du défi n°2
- Une notice d'utilisation de GNAT Studio pour mettre au point des programmes
- Le code de *verifierCeCode* et ses deux paquetages

Cette semaine vous avez trois défis à relever :

1. Organiser ses données
2. Apprendre à utiliser GPS
3. Apprendre à répartir le code dans des paquetages

La semaine prochaine vous déposerez l'ensemble de ces codes sur GIT.

Défi 1 : organiser ses données

Ce SAE durera 4 semaines. Chaque semaine il y aura une nouvelle mission à accomplir. Dans chaque mission il y aura plusieurs étapes, défis ou exercices à faire.



Défi 2 : apprendre à utiliser GPS

1. Découverte du fonctionnement de GNAT Studio

A partir de l'archive *GestionNotes* déposée sur Moodle, découvrez comment fonctionne GPS.

Indices :

- La porte d'entrée est *GestionNotes.gpr*
- Le code se trouve dans le répertoire *src*
- En anglais, exécuter un programme se dit *run*
- Le but est atteint lorsque le message « *process terminated successfully* » s'affiche dans deux fenêtres différentes

- Le répertoire *obj* peut être nettoyé avec la commande  *Clean all*
- Dans un QCM j'ai déjà vu un programme de tests pour ce programme
- Il y a plein d'icônes utiles pour mettre au point un programme 
- Quelque part, mon code peut être mis en forme automatiquement (Yes !!)
- Quand c'est rose, ce n'est pas bon signe. Mais GPS aime bien le rose et persiste parfois à tord.

Le résultat de cette mission :

- un superbe programme *GestionNotes.adb* qui n'a plus de bug et qui est convivial (il ne jette plus le malheureux utilisateur qui a saisi une note non comprise entre 0 et 20)
- une procédure de mise au point des programmes grâce aux différents boutons


2. A partir de RIEN

Indices :


- Créer un nouveau projet se dit *Create a new project* en anglais
- Une règle de base : faire toujours au plus SIMPLE
- C'est cool, il me donne un nom de projet par défaut
- Comment dit-on programme principal en anglais ?
- Comme petit programme, je peux prendre le *affichage.adb* qui est sur Moodle. J'apprendrai des choses utiles pour la séance de la semaine prochaine.
- Tant que je joue avec le programme principal, tout va bien ... Mais que se passe-t-il si je crée un nouveau programme ? Peut-être faudrait-il aller voir dans

+

Edit > Project Properties > Main et jouer avec le  ?

- Dans un QCM j'ai déjà vu un programme de tests pour ce programme
- Il y a plein d'icônes utiles pour mettre au point un programme 
- Quelque part, mon code peut être mis en forme automatiquement

Le résultat de cette mission :

- Mon nouveau projet est bien rangé dans le répertoire de l'exercice 2 de la semaine 1 du SAE1.01
- J'ai au moins deux programmes principaux différents que je peux exécuter au choix en cliquant sur la petite flèche de l'icone 

Défi 3 : apprendre à dissocier les sous-programmes du programme principal

A partir du code du programme *VerifierCeCode.adb* déposé sur Moodle, découvrez comment fonctionne un code PROPRE en dissociant les sous-programmes du programme principal.

Principe :

- Les sous-programmes se trouvent dans un paquetage. Un paquetage est composé de deux parties en Ada :
 - 1) Une partie spécification
 - Introduite par le mot clef *package NomDuPaquetage is*
 - Terminée par *end NomDuPaquetage ;*
 - Dont le nom de fichier est *NomDuPaquetage.ads*
 - Qui contient les entêtes (ou spécifications) des sous-programmes
 - Qui contient les structures de données
 - 2) Une partie implémentation
 - Introduite par le mot clef *package **body** NomDuPaquetage is*
 - Terminée par *end NomDuPaquetage ;*
 - Dont le nom de fichier est *NomDuPaquetage.adb*
 - Qui contient le corps des sous-programmes
 - Qui contient les opérations de manipulation des données
- Le programme principal utilise la commande : *use nomDuPaquetage ; with nomDuPaquetage* pour connaître les sous-programmes

Indices :

- Dans le TP dessins, *traceur* est un paquetage
- Ce code possède deux sous-programmes qui utilisent la même structure de données
- L'entête d'un sous-programme comprend tous les commentaires qui se trouvent AVANT le mot clef *procedure* ou *function*
- L'entête d'un sous-programme se termine AVANT le *is*
- L'entête d'un sous-programme se termine par ;
- Il y a une commande magique dans le menu *Code* qui génère automatiquement le *package body* pourvu que j'ai commencé par faire le paquetage de spécification avant
- Tous les codes doivent être dans le répertoire *src* sinon ça ne marche pas
- Le code 1234 est correct
- Le code 5732 n'est pas correct
- Quand le paquetage est terminé, dans le programme principal il n'y a plus que les appels aux sous-programmes.

Le résultat de cette mission :

- 3 fichiers :
 - *VerifierCeCode.adb* qui contient le programme principal qui permet de tester les sous-programmes
 - *codeUtils.ads* qui contient les spécifications des sous-programmes
 - *codeUtils.adb* qui contient le code des sous-programmes
- une exécution correcte de *VerifierCeCode*

Pour aller plus loin : proposer 2 autres versions de *vérifierCeCode* en utilisant des structures de données différentes.