Situation d'Apprentissage et D'Evaluation 1.01

Semaine 2 – 26 octobre 2021

Objectif pédagogique : Construire un paquetage à partir d'un enregistrement

Modalité : travail en binôme

Nombre de séances de TP : 2*1h30

Nombre de dépôts sur Git :

- Le travail de la semaine dernière
- Le paquetage *Type_Usager*
- Le programme principal contenant les tests
- Le *ReadMe* sur Git qui présente le thème de votre application

---

Au cours de ce SAE vous allez développer une application en Ada qui gère des usagers. Les informations de ces usagers seront stockées dans un tableau. L'application a pour objectif de permettre de saisir, stocker, afficher, rechercher, ajouter, supprimer voire trier ces usagers.

Vos missions de cette semaine :

1. Trouver une application adaptée à cette situation. Par exemple, cette application peut être la gestion des adhérents d'une mutuelle, d'un jeu vidéo, … et identifier les informations nécessaires au fonctionnement de cette application.
2. Programmer le paquetage Usager qui comprend
    a. La définition du type de donné *Type_Usager* constitué d'un nombre minimal de champs
    b. Un ensemble minimal de sous-programmes qui implémentent les opérations de manipulation du *Type_Usager*
3. Proposer (en parallèle avec l'implémentation des opérations) un programme principal qui assure le comportement attendu des sous-programmes

En ce qui concerne le dépôt de cette semaine : **déposez les sources de votre projet Ada dans un dépôt distant sur GitLab à 17h**.

Le *readme* du projet doit décrire l'applications que vous avez choisi.

Attention ! Il est important que votre dépôt ait les « tâches minimales » imposées pour cette semaine.

Lors des deux semaines suivantes vous compléterez votre application.

Ce qui est IMPOSE pour cette semaine :

1. Le type Usager

Cet enregistrement comprend à minima ces 3 champs :

a. Un numéro identifiant
b. Un nom d'usager
c. Une année de naissance

2. Les opérations du *type_Usager*

Les opérations suivantes doivent être implémentées dans le paquetage Usager. Vous pourrez néanmoins en implémenter d'autres si vous pensez qu'elles seront utiles pour votre application. Nous présentons chaque opération avec ses critères de validation.

Nom de l'opération	Fonctionnalité (Rôle du sous-programme)	Comportement attendu

getUsager	L'opération de lecture demande de saisir les valeurs des différents champs de l'enregistrement et stocke ces informations dans une variable u de *Type_usager* .	La figure ci-dessous montre le résultat attendu à l'écran par cette opération.
		`Quel est le numero d'adherent de l'usager ?`
		`123456`
		`Quel est le nom de l'usager ?`
		`Dark Vador`
		`Quel est l'annee de naissance de l'usager ?`
		`1968`
		
		ATTENTION : pour être validée, **cette opération doit absolument respecter cette interface lettre à lettre**. Pour les champs supplémentaires que vous ajoutez au Type-Usager, il n'y a aucune contrainte particulière.
		
		POUR ALLER PLUS LOIN :
		Vérifications de la validité des valeurs saisies. Par exemple la date de naissance ou la valeur de l'identifiant. Vous préciserez bien dans les préconditions la nature exacte des valeurs valides.

putUsager	L'opération d'affichage permet d'afficher toutes les valeurs des champs d'une variable u de type *Type_usager*.	La figure ci-dessous montre le résultat attendu à l'écran par cette opération :
		`Le numero d'adherent de l'usager est 123456`
		`Son nom est Dark Vador`
		`Son annee de naissance est 1968`
		
		ATTENTION : pour être validée, cette opération doit absolument respecter cette interface lettre à lettre. Pour les champs supplémentaires que vous ajoutez au *Type-Usager*, il n'y a aucune contrainte particulière

getAge	Cette opération permet de calculer l'âge actuel d'un usager *Type_usager*. L'âge est calculé à partir de son année de naissance	Pour être validée, l'âge en sortie doit être calculé comme : 2021 – l'année de naissance de l'usager

Nom de l'opération	Fonctionnalité (Rôle du sous-programme)	Comportement attendu
Affectation (opérateur :=)	Cette opération permet d'affecter à e variable de type *Type_Usager* les mêmes valeurs enregistrées dans une deuxieme variable de type *Type_Usager*. En pratique : Usager2 := usager1	Pour être validée, vous devez vérifier que après l'opération, TOUTES les informations stockées dans les deux usagers sont identiques.
Égalité (opérateur =)	Cette opération permet de comparer les informations de deux usagers de type *Type_Usagers*. L'opération retourne VRAI si les deux usagers ont les mêmes informations sans prendre en compte l'identifiant.	Pour valider cette opération, vous devez effectuer la vérification sur plusieurs variables pour envisager tous les scénarii possibles : - Deux utilisateurs avec même valeurs - Deux utilisateurs avec differentes valeurs - -...à reflechir ;;.
IdentityCompare	Cette opération permet de comparer si deux usagers de type *Type_Usagers* sont IDENTIQUES ou non, c'est-à-dire s'il ont les mêmes informations et le même identifiant. Si les informations et l'identifiant sont les mêmes, l'opération retourne VRAI	Pour valider cette opération, vous devez vous devez effectuer la vérification sur plusieurs variables pour envisager tous les scénarii possibles selon la même logique que pour l'opérateur d'égalité.
"<"	Cette opération permet de comparer si un usagers de type *Type_Usagers* a une valeur d'identifiant plus petite qu'un deuxième usagerde type *Type_Usager*. L'opération retourne VRAI si **l'identifiant** de l'usager 1 et inférieur à celui de l'usager 2.	Pour valider cette opération, vous devez effectuer la vérification sur plusieurs variables pour envisager tous les scénarii possibles selon la même logique que pour l'opérateur d'égalité : • Le cas égaux • Le cas inférieurs • Le cas supérieurs

3. Les jeux de tests

Pour chaque opération vous devez définir un ensemble de tests permettant de valider le sous-programme qui l'implémente. Vous pouvez définir des sous-programmes pour chaque test d'opération.

4. Pour aller plus loin

- Vous pouvez faire un paquetage de tests.
- Vous pouvez créer un fichier texte d'usagers via un nouveau paquetage qui contient l'opération *getUsager2File* qui inscrit les informations d'un usager dans un fichier

Ce qui sera évalué :

- le type *Type_Usager* qui comprend au moins les 3 champs imposés
- l'implémentation des opérations demandées
- le programme de test des opérations demandées

En points bonus

- la pertinence des champs supplémentaires ajoutés

- la pertinence des opérations supplémentaires proposées
- la pertinence du programme de tests des opérations supplémentaires que vous proposez