Matière	Programmation orientée objet
Enseignent	M. NAJIB
Date	03-01-2024
Durée	2h

Consignes : 1-Mémento de syntaxe autorisé. 2-Documents non-autorisés.

	***Course Outcomes* (Syllabus) avec le pourcentage**		
Course Outcomes	**Evaluer**	**Exercice**	**Pourcentage**
CO1: Master the basic concepts of OOP	X	1	50%
CO2: Master the visibility of attributes, methods, inheritance principles, polymorphism, abstract classes, and interfaces	X	2	33%
CO3: Develop ergonomic graphic interfaces and events programming using JAVA	X	3	50%

Exercice 1 : (3 pts)

Questions de cours :

1. Citez deux avantages de l'approche orientée objet. (**1pt**)
2. Expliquer quand est-ce qu'il faut utiliser une méthode de classe en JAVA. (**1pt**)
3. Expliquez le fonctionnement d'un BorderLayout en JAVA ? (**1pt**)

Exercice 2 : (9 pts)

Nous souhaitons développer une application basée JAVA pour la gestion des activités proposées aux enfants dans un camps d'hiver. Trois types d'activités sont proposées :

- **Activité** (***idActivité (auto-inc)***, intitulé)
- **ActivitéSport**(***idActivité***, intitulé, nombreSéance (privé), prixSéance)
- **ActivitéMusique** (***idActivité***, intitulé, duréeHeure (int), prixHeure)

Contraintes

- On suppose que la classe « **Activité »** est une classe abstraite qui contient une méthode abstraite **CalcMontant ()** pour le calcul du montant à payer pour une activité.
- Le montant à payer pour une activité est calculé sur la base de sa durée / le nombre de séance et le prix par heure ou le prix par séance.
- On suppose que la classe « **ActivitéMusique »** est donnée avec toutes les méthodes de base.

Questions

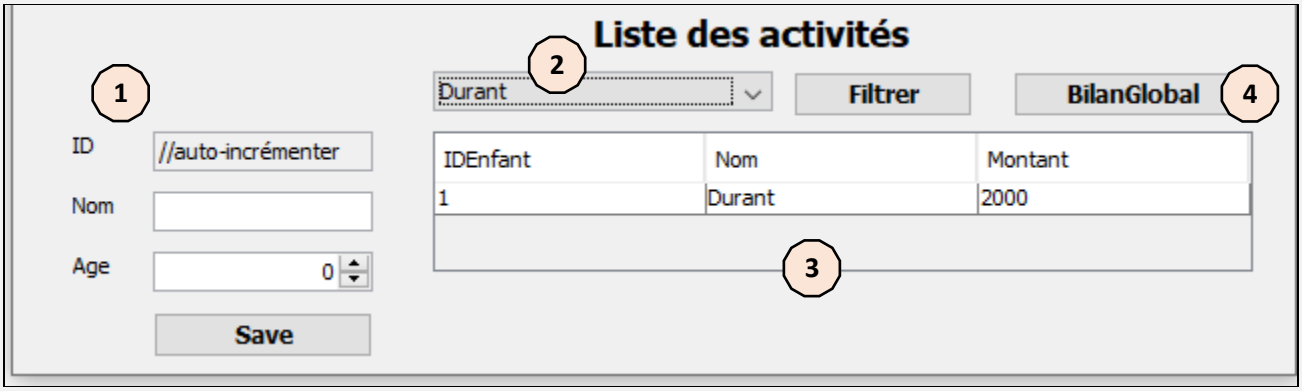
1. Donnez un schéma qui illustre la relation d'héritage entre les classes Activités. (Préciser les attributs) (**1pt**)
2. Donnez une implémentation de la classe « **Activité »** (constructeurs, méthode calcMontant, toString) (**2pt**)
3. Donnez une implémentation de la classe « **ActivitéSport »**. (**2pt**)
4. Créez une instance de l'activité de **Musique**, une instance de l'activité de **sport** et enregistrez ces deux objets dans une liste « **ListeActiv** » à déclarer dans la classe « **Main.java** ». (hard-coded)(**2pt**)

5. Donnez le code nécessaire pour trier la liste précédente en ordre croissant du montant à payer pour chaque activité. (**2pt**)

Exercice 3 : (8 pts)

Nous souhaitons gérer les inscriptions des enfants aux différentes activités proposées dans l'exercice 2. Chaque enfant est caractérisé par son idEnfant(**auto-inc**), age (int), nom, la liste des activités auxquelles il est inscrit.

- La classe « **Enfant.java** » est donnée avec toutes les méthodes de base (constructeurs, toString).
- La liste des enfants « **listeEnf** » (de type LinkedList) est déclarée comme un attribut de classe dans le « **Main.java** »
- Pour simplifier la gestion, on suppose que les noms des enfants ne sont pas dupliqués.



Questions

1. Donnez le code nécessaire (bouton save) pour enregistrer un nouvel enfant dans la liste « **listeEnf** » (avec une liste d'activité vide). Récupérer les informations à partir du formulaire. (**2pt**)
2. Donnez le code nécessaire pour initialiser la « **JComboBox** » avec la liste des étudiants au lancement de l'interface graphique. Précisez aussi le type d'événement à implémenter. (**2pt**)
3. Donnez le code nécessaire pour afficher le bilan suivant (voir **Jtable**) pour un enfant sélectionner à partir de la « **JcomboBox** ». Le montant affiché correspond au montant à payer pour toutes les activités de l'enfant sélectionné. (**2pt**)
4. Donnez le code de l'action du bouton « **BilanGlobal** » pour générer le bilan suivant dans une nouvelle fenêtre : (**2pt**)

Nombre global d'enfants	**12**
Nombre global d'activités	**25**
Chiffre d'affaire global	**21500 DH**

Bonne chance