UEF4.3. Programmation Orientée Objet
Contrôle Final
2h - Documents & Téléphone interdits

Nom ……..…… Prenom …..…… groupe ……..

Exercice 1 (5 pts)

Choisissez la ou les bonne(s) réponse(s):

Q1. Qu'affiche le programme suivant ? 0,75pt

```
static class Outer {
  public static void main(String[] args) {
     System.out.println("Outer");  }
  private class Inner {
    Inner() {System.out.println("Inner"); }
  }
}
```

A. Outer
B. Rien. Il y aura erreur à la compilation : la classe Inner doit être déclarée statique.
C. Inner.
D. Rien. Il y aura erreur à la compilation : la classe outer ne peut pas être déclarée statique.

Q2. A quoi sert le mot clé transient associé à un attribut ? 0,5

A. La valeur de cet attribut doit être sauvegardée dans un fichier
B. L'attribut doit être ignoré lors de la sérialisation
C. L'attribut est une instance d'une classe sérialisable.
D. Rien de tout cela

Q3. Que signifie la sérialisation d'un Objet ? 1pt=0,5*2

A. Enregistrer le code de la classe de l'objet
B. Sauvegarder l'état d'un objet dans un fichier.
C. Réinitialiser les valeurs des attributs de l'objet à null.
D. Rendre l'objet persistant.

Q4. Quel est l'effet de l'instruction? 0,5pt

File f = new File("source.txt");

A. Créer un fichier « source.txt » dans le système d'exploitation.
B. Créer un objet f et le lier au fichier « source.txt » s'il existe.
C. Ouvrir le fichier « source.txt » en lecture seule.
D. Si le nom du fichier « source.txt » n'existe pas, alors le fichier est créé. S'il existe, les données seront écrasées.

Q5. Quel est le nom de la superclasse des flux de sortie binaires? 0,5

a. InputStream
b. OutputStream
c. Reader
d. Writer

Q6. Qu'affiche le programme suivant ?0,5

```
public enum Sens {
 HAUT, BAS, DROITE, GAUCHE}
public enum Orientation extends Sens {
  public void afficher () {
     System.out.println (toString() ) ;
  }
}
```

```
public class Q6 {
 public static void main(String[] args) {
  for (Orientation s : Orientation.values() )
    s.afficher() ;
 }
```

A. HAUT, BAS, DROITE, GAUCHE

B. Rien. Il y aura erreur à la compilation : la classe Orientation ne peut pas hériter de Sens.

C. Rien. Il y aura erreur à la compilation : la classe Orientation ne peut pas déclarer la méthode afficher.

Q7. Quand est-ce qu'un programme JavaFX exécute le code des gestionnaires d'événement (event handlers)?0,5

A. Lors de la phase montante du parcours de la chaîne de traitement de la cible jusqu'à la racine.

B. Lors de la phase descendante du parcours de la chaîne de traitement de la racine jusqu'à la cible.

C. Lors de la création de la chaîne de traitement.

Q8. Qu'affiche le programme suivant sur la console lorsqu'on clique sur le bouton "OK" ?0.75pt

```
//tous les import nécessaires

public class Q8 extends Application {
public void start(Stage stage) {
BorderPane root = new BorderPane();
Scene scene=new Scene(root, 400, 300);
Button btn = new Button("OK");
btn.addEventFilter(
 MouseEvent.MOUSE_CLICKED,
new EventHandler<MouseEvent>() {
          public void handle(MouseEvent b) {
              System.out.print(" Événement traité
par le bouton ");
             }
      });
scene.addEventFilter(
   MouseEvent.MOUSE_CLICKED,
   new EventHandler<MouseEvent>() {
          public void handle(MouseEvent s) {
                System.out.print(" Événement
traité par la scène ");
              s.consume();
            }
       });
root.setCenter(btn);
stage.setScene(scene);
stage.show();
}
public static void main(String[] args) {
Application.launch(args);
}
}
```

A. Événement traité par la scène Événement traité par le bouton
B. Événement traité par le bouton
C. Événement traité par le bouton Événement traité par la scène
D. Événement traité par la scène

Exercice 2 (15 pts) (Le barème est sur 30)

A/ Un centre sportif offre à ses adhérents la possibilité de s'inscrire à une activité sportive parmi une liste de sports pratiqués dans le centre (natation, fitness, musculation et dance). A l'inscription d'un adhérent, un numéro lui est attribué comme identifiant. Il s'agit d'un numéro d'ordre qui s'incrémente de 1 à chaque nouvelle inscription. L'adhérent peut choisir de pratiquer ses séances individuellement ou s'inscrire dans un groupe pour être suivi par un coach. Pour suivre l'évolution des adhérents aux groupes, le coach met une appréciation à chacun d'eux sous forme d'un petit texte dans lequel il décrit la progression de l'adhérent.

Le centre sauvegarde pour chaque membre, qu'il soit coach ou adhérent, ses informations personnelles (nom, prénom, numéro de téléphone) ainsi que son groupe sanguin (O+, O-, A+, A-, B+, B-, AB+, AB-). Chaque coach est spécialisé dans une seule discipline sportive et peut entraîner plusieurs groupes. Un groupe est formé de plusieurs adhérents et est caractérisé par un code alphanumérique saisi par le gérant du centre au moment de sa création. Pour accéder au centre, l'adhérent doit procéder au paiement d'un certain nombre de séances qui est décrémenté après chaque séance. Le paiement est renouvelé une fois toutes les séances consommées.

Le club désire se doter d'un programme informatique permettant de:

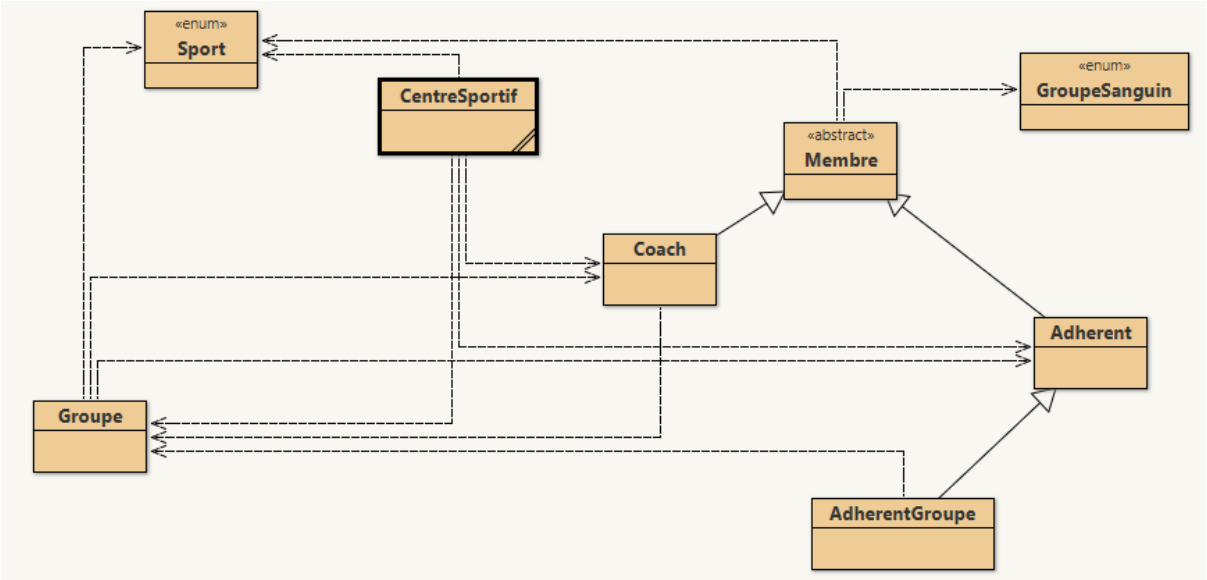
- Stocker les adhérents selon le sport qu'ils pratiquent.
- Stocker les coachs selon leur sport, ordonnés par ordre alphabétique de leurs noms.
- Stocker les groupes en garantissant que deux groupes n'ont pas le même nom.
- inscrire un nouvel adhérent ou nouveau coach
- Afficher les informations concernant un adhérent, un coach ou un groupe.

1. Proposer un programme orienté objet assurant les fonctionnalités demandées.
    a. Tracer le diagramme des classes en précisant la nature des classes et les relations entre elles.

Solution

Enumerations Sport et GroupeSanguin 0.25*2
Classes CentreSportif, Coach, Adherent, AdherentGroupe,Groupe 0.5*5
classe abstraite Membre 0.5+0.5
Relations d'héritage 0.5 *3
Relations d'utilisation 1 (de 0 à 1 selon les relations mentionnées)
—----------

6.5 points



b. Donner pour chaque classe un tableau présentant ses attributs et ses méthodes (hormis les constructeurs, les getters et les setters). Si des collections sont utilisées, préciser leurs types; et donner toutes les méthodes nécessaires pour leur bon fonctionnement en fournissant une brève description expliquant pourquoi ces méthodes sont nécessaires. En l'absence d'explication, la réponse n'est pas comptabilisée)

NB: le code source des méthodes n'est pas demandé

Type de la classe Nom de la classe	
Liste des attributs	
Modificateur-accès Type-attribut nom-attribut	Description
……	……..
Liste des méthodes	
Modificateur-accès Type-valeur-retour Signature de la méthode	Description
……	……..

Solution 10 pt

Enum Sport
natation, fitness, musculation, dance

Enum GroupeSanguin
O+, O-, A+, A-, B+, B-, AB+, AB-

0.25*2

abstract class Membre **1.25**	
Attribut	**description**
protected String nom, prenom; protected String numTel; protected GroupeSanguin grs; protected Sport sport; **0.25 * 4**	
Methode	**Description**
public abstract void afficher() **0.25**	Affiche l'état d'un objet Membre

class Adherent extends Membre **1.25**	
Attribut	**description**
protected int num; protected int nbSéances; **0.25 *2** static int nbAdherents; **0.5**	Static pour servir à générer le numéro d'Adhérent
Methode	**Description**
public void afficher() **0.25**	

class AdherentGroupe extends Adherent **0.5**	
Attribut	**description**
String appréciation; **0.25** Groupe groupe // facultatif	
Methode	**Description**
public void afficher() **0.25**	Redéfinition de afficher de la classe Adhérent

class Coach extends Membre implements Comparable 1.25	
Attribut	**description**
private ArrayList <Groupe> groupe; **0.5**	
Methode	**Description**
public void afficher () **0.25** public int compareTo() **0.25**	la redéfinition de la compareTo est nécessaire car la classe implémente l'interface Comparable dans le but de stocker les coachs ordonnés selon l'ordre alphabétique de leurs noms **0.25**

class Groupe 2.25	
Attribut	**description**
private String code; private Coach coach; private Sport sport; **0.25 * 3** private ArrayList <AdherentGroupe> adherents; **0.5**	
Methode	**Description**
public void afficherAdherents() **0.25** public boolean equals(Object o) **0.25** public int hashCode() **0.25**	affiche les adhérents appartenant au groupe redéfinition de equals(Object o) et hashCode() car les groupes sont stockés dans un HashSet **0.25**.

class CentreSportif 3	
Attribut	**description**
private HashMap<Sport, ArrayList<Adherent>> adhérents; **0.75** private HashMap<Sport, TreeSet<Coach>> **0.5** private HashSet<Groupe> groupes; **0.5**	
Methode	**Description**
public void InscrireAdherent(Adherent a) public inscrireCoach(Coach a) public affecterAdherentGroupe(aAdherent a, Groupe g) public void afficher(Membre m) public void afficher(Groupe g) **0.25 * 5**	

B/ Pour moderniser sa gestion, le club offre à chacun de ses adhérents une carte électronique qu'il présente à un lecteur de cartes situé au niveau de l'accès à la salle. Cette carte sert à sauvegarder le nombre de séances payées par l'adherent. A chaque séance ce nombre est dérémenté de 1 et une fois arrivé à zéro, un message invite l'adhérent à renouveler son paiement.

Voici ci dessous un code java incomplet présentant la classe Adherent

```
public class Adherent {
      ……..     // autres attributs de la classe Adherent
      private CarteAdherent carte;

     class CarteAdherent {
          private int nbSeances;
          public CarteAdherent (){nbSeances = 0; }
          public void paiementSeances(int nb){nbSeances = nb;}
       ……..  // méthode demandeAccès ()

     } // fin de la classe CarteAdherent

      ……..// autres méthodes de la classe Adherent
     }
```

1. Quelle est la nature de la classe CarteAdherent ?
2. Pourquoi a t-elle été conçue de cette manière à votre avis ?
3. Donner le code source de la méthode demandeAccès ()

Solution. **2.25**

4. Quelle est la nature de la classe CarteAdherent ? **0.5**
   C'est une classe interne
5. Pourquoi a t-elle été conçue de cette manière à votre avis ? **0.5**
   Car seule la classe Adherent l'utilise. Ceci facilite la maintenance et améliore la lisibilité du code et augmente l'encapsulation
6. Donner le code source de la méthode demandeAccès ()

```
public void demandeAcces ()throws PaiementException 0.25+0.25{
        if (nbSeances == 0) 0.25 throw new PaiementException(); 0.25
        else nbSeances --; 0.25
    }
// Remarque   PaiementException est une exception perosnnalisée dont voici le code
class  PaiementException extends Exception {}
```

C/ Le centre élargit son offre en accueillant des adhérents appartenant à des clubs qui ont un niveau avancé dans un sport donné. En plus des informations demandées aux adhérents, le centre garde trace des noms de leurs clubs. Pour apporter de la dynamique à la vie du centre, des compétitions sont occasionnellement organisées auxquelles seuls les

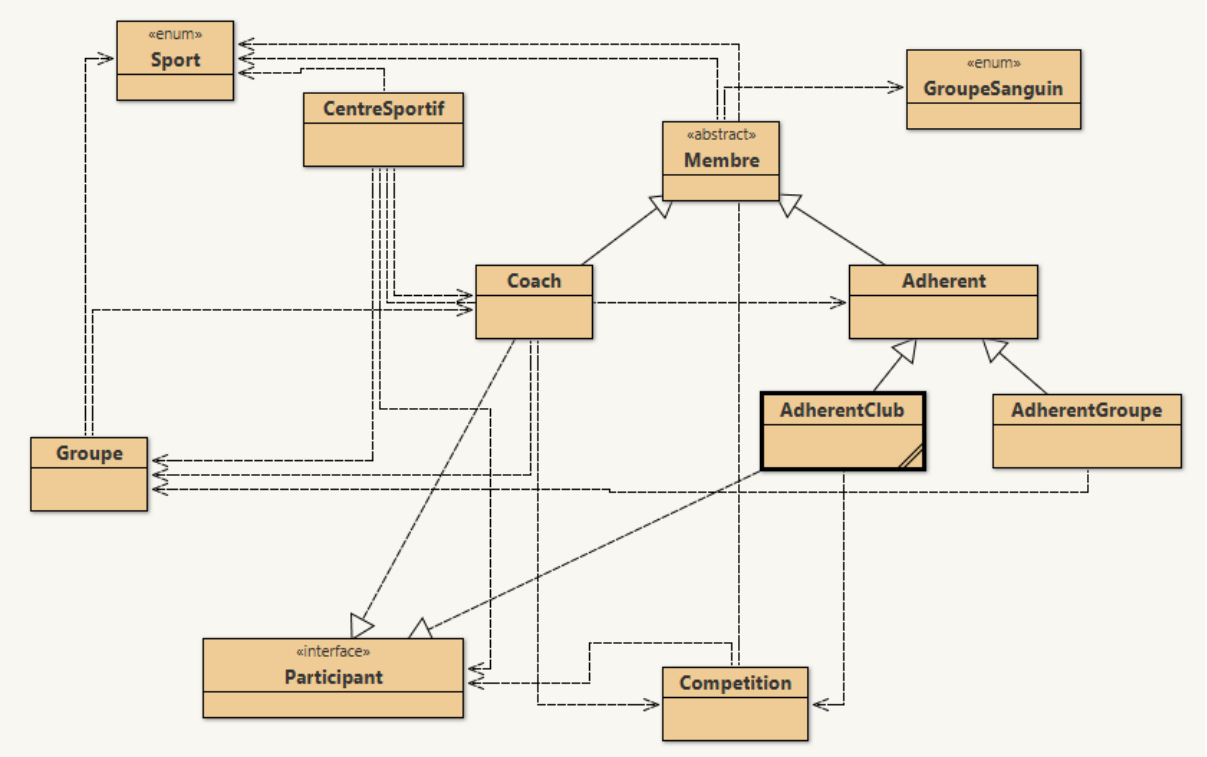
coachs et les adhérents issus de clubs peuvent participer. Le centre mémorise chaque compétition en sauvegardant sa date, le sport concerné et la liste des participants qui y prennent part.

Nous souhaitons adapter notre programme aux nouveaux besoins du centre sportif afin de lui permettre de:

- Stocker toutes les compétitions classées par discipline sportive.
- Stocker pour chaque coach ou adhérent issu d'un club les compétitions auxquelles il a participé et le meilleur classement qu'il a obtenu
- Stocker les participants selon la discipline sportive et ordonnés selon leur meilleur classement.
- Afficher les compétitions et les participants.

1. Quelles sont les modifications à apporter au programme proposé précédemment si l'on veut assurer ces nouvelles fonctionnalités.
    - Compléter le diagramme proposé dans la partie A par les nouvelles classes à ajouter en précisant la nature des classes et les relations entre elles
    - Donner les tableaux des nouvelles classes
    - Donner les tableaux des classes qui ont subi des modifications en soulignant ce qui a été modifié.

Solution
AdherentClub et competition 0.5 *2
interface Participant 1
Relations: héritage 0.5, utilisation 0.5; implementation 0.5*2
—----------
4 pt



Tableaux 27.25

Remarque: seules les méthodes essentielles que l'étudiant peut extraire à partir de l'énoncé sont données ici. D'autres méthodes nécessaires ne sont pas mentionnées.

interface Participant 0.5	
Attribut	**description**
Methode	**Description**
public void aficherCompetitions(); **0.5** **// Getters et setters de l'attribut meilleur classement**	

class Coach extends Membre implements Participant, Comparable 1.5	
Attribut	**description**
private ArrayList <Groupe> groupe; private ArrayList<Competition> competitions; **0.5** private int meilleurClassement; **0.25**	
Methode	**Description**
public void afficher() public void aficherCompetitions(); **0.25** public int compareto(Participant p) **0.5**	la méthode compareTo doit être redéfinie car la classe implémente Comparable. L'argument effectif est de type Participant car les coachs et les AdherentClubs sont stockés dans la même collection (voir ci-après la classe CentreSportif)

class AdherentClub extends Adherent implements Participant, Comparable 1.5	
Attribut	**description**
private String club; private ArrayList<Competition> competitions; **0.5** private int meilleurClassement; **0.25**	
Methode	**Description**

public void afficher() public void aficherCompetitions(); **0.25** public int compareto(Participant p) **0.5**	

class Competition 1.5	
Attribut	**description**
Sport sport; **0.25** LocaDate date; **0.25 // ou Date** ArrayList <Participant> participants; **0.75**	
Methode	**Description**
public void afficher ()**0.25**	

class CentreSportif 2.25	
Attribut	**description**
private HashMap<Sport, ArrayList<Adherent>> adhérents; private TreeSet <Coach> coachs; private HashSet<Groupe> groupes; private HashMap<Sport, ArrayList<Competition>> competitions; **0.75** private HashMap<Sport, TreeSet<Participant>> participants; **1 (dont 0.25 si le type du TreeSet est Particiant)**	
Methode	**Description**
public void aficherCompétitions() **0.25** public void afficherParticipants() **0.25**	