Projet noté

(encadré pendant 1 TD et 3 TP, travail personnel nécessaire)

Règles strictes :

- Projet à réaliser en **binôme** (1 binôme = 2 personnes) ;
- Rapport et programmes Java à rendre au plus tard le **20 décembre 2019** sous la forme d'un (et un seul) fichier compressé au format **zip** ou **tar.gz** ;
- Ce fichier doit être déposé sur madoc par **un** (et un seul) membre du binôme dans l'espace devoir correspondant à son groupe de TP.

Sujet

On s'intéresse à un réseau d'échanges de services dans lequel des membres réalisent des tâches en faveur d'autres membres et reçoivent des rétributions en échange. La monnaie d'échange au sein d'un réseau est le *jeton*.

- Un service a un nom et un coût horaire (un nombre entier de jetons), par exemple le service de *jardinage* valant 10 jetons par heure et par personne. Chaque membre d'un réseau propose une liste de services.
- Une tâche est définie par un service pour un membre bénéficiaire sur une durée (un nombre réel en heures) et devant être réalisé par un nombre de personnes donné. Le prix d'une tâche pour un service de coût horaire C sur une durée d effectué par p personnes est égal à la partie entière de $C \times d \times p$. Parmi toutes les tâches, les tâches bénévoles ne valent rien.
- Un membre possède un nom et une somme d'argent (un nombre entier de jetons). Il reçoit une certaine somme au départ. Cette somme évolue au gré des services rendus ou reçus.
- Le paiement d'une tâche dépend de la classe sociale du bénéficiaire. Si une tâche vaut p jetons, ce membre paye p s'il est de classe *normale*, $p/2$ s'il est de classe *demie* ou 0 s'il est de classe *zero*. Les membres ayant réalisé la tâche se partagent équitablement la somme payée par le bénéficiaire.
- Un réseau possède un administrateur. C'est un membre particulier de classe normale chargé de créer son réseau, d'ajouter de nouveaux membres, de retirer des membres et de valider la réalisation des tâches. Une tâche peut être réalisée si on trouve suffisamment de personnes au sein du réseau possédant la compétence recherchée et si le bénéficiaire a la somme d'argent nécessaire. Pour la création d'un administrateur et de son réseau, on peut envisager les instructions suivantes :

  ```
  Admin lea = new Admin("lea");
  lea.creerReseau("echange");
  ```

  Pour la création des autres membres, on peut envisager les instructions suivantes :

  ```
  Membre maxime = lea.creerMembre("maxime", new ClasseZero());
  Membre charly = lea.creerMembre("charly", new ClasseDemie());
  ```

Bien sûr, il faut en plus attacher des services à ces membres (y compris l'administrateur qui a aussi le rôle d'un membre ordinaire), ce qui n'est pas fait dans l'exemple ci-avant.

Le travail consiste donc à programmer toutes les classes nécessaires à la réalisation de ce cahier des charges.

1. Dans un premier temps, il faut identifier les classes, définir les relations entre ces classes et représenter les objets en mémoire, bien avant de commencer à programmer. Ce travail sera abordé dans la séance de TD encadrée avec chaque binôme (former les binômes avant la séance de TD et se mettre à côté pendant la séance).
2. Dans l'activité de programmation, vous pourrez commencer à développer les classes les plus élémentaires. Dans chaque classe, il est conseillé d'implémenter et de tester les méthodes une par une. Ne pas attendre de programmer toute une classe avant de tester les méthodes.
3. Vous constaterez que votre solution est extensible si l'ajout d'un nouveau type de tâche ou de classe sociale de membre du réseau se fait sans modifier le code existant. Il faudra avoir un œil critique sur ce point dans le rapport.

Questions subsidiaires

Si le temps le permet, vous pourrez traiter les aspects suivants :

— la protection d'un réseau pour empêcher les modifications (ajout ou retrait de membre) sans l'accord de son administrateur ;

— la possibilité d'avoir plusieurs administrateurs dans un réseau ;

— la gestion d'un historique des tâches réalisées ;

— la création des tâches à partir de données stockées dans un fichier au format *csv* ;

— l'archivage d'un historique dans un fichier ;

— une interface graphique pour gérer / animer un réseau.

Rapport et programmes Java à rendre

Les programmes Java seront dûment commentés. Le rapport (se voulant complet et succinct) comprendra les parties suivantes :

1. une page de titre : noms, prénoms, groupe de TP, formation, logo de l'université, année, titre ou sujet ;
2. une introduction rappelant brièvement le sujet ;
3. la modélisation présentée sous la forme d'un diagramme de classe en justifiant clairement les relations entre les classes (par exemple : il existe une hiérarchie de [*blablabla*] avec une super-classe [*blablabla*] et des sous-classes [*blablabla*] sachant que [*blablabla*] ; un objet B se comporte comme un objet A avec en plus [*blablabla*] donc B hérite de A ; un objet B se compose de plusieurs objets A car [*blablabla*]) ;
4. la spécification de chaque classe avec son rôle et un tableau à deux colonnes donnant pour chaque méthode publique sa signature et une description brève, par exemple :

   Classe : `UneClasse`

   Rôle : la classe `UneClasse` permet de créer des objets [*blablabla*] et de réaliser les actions [*blablabla*]

   Méthodes :

Signature	Description
`UneClasse()`	construit une instance telle que [*blablabla*]
`void action(int n)`	réalise `n` fois [*blablabla*]
`int f()`	calcule [*blablabla*] et retourne [*blablabla*]

5. des traces d'exécution sur des exemples choisis ;
6. une conclusion avec un regard critique sur votre projet en montrant que votre solution est extensible (ou pas), en identifiant clairement les réussites et les échecs (par exemple : l'algorithme permettant de [*blablabla*] est performant car [*blablabla*] ; l'implémentation de la classe [*blablabla*] n'a pas été faite car [*blablabla*]) et en indiquant si le principe d'encapsulation est respecté.

Enfin, il est fortement recommandé de ne pas écrire à la première personne (par exemple : j'ai fait [*blablabla*] ou nous avons créé [*blablabla*]) et de ne pas décrire la réalisation du projet de manière chronologique (par exemple : dans un premier temps nous avons fait [*blablabla*] puis comme [*blablabla*] nous avons fait autrement [*blablabla*] et enfin [*blablabla*]). Soyez clairs, précis et concis dans vos explications. Utilisez un correcteur orthographique. Faites relire votre rapport par des camarades ou des proches.

Compléments de programmation

Soit la déclaration suivante :

```
final int nb = 5;
```

Le mot-clé `final` indique que la valeur de `nb` ne pourra plus changer par la suite. C'est donc un moyen de définir une constante. Une classe peut être également `final`. Cela indique qu'on ne pourra pas en faire des sous-classes.

Dans une classe, il est possible de définir une variable qui n'est pas un attribut des objets ; on parle de variable de classe. Pour cela, il faut utiliser le mot-clé `static`, par exemple :

```
public class Param {
   public static int BaseArgent = 50;
   ...
}
```

La variable `Param.BaseArgent` est une variable déclarée dans la portée de la classe `Param`. De plus, il est possible de combiner les mot-clés `final` et `static`. Par exemple, dans la classe standard `Math` on trouve la valeur approchée de π :

```
public class Math {
   public static final double PI = 3,14159265358979323846;
   ...
}
```

Dans une classe, on peut définir également une fonction / procédure qui n'est pas une méthode des objets ; on parle de fonction de classe. Par exemple, dans la classe standard `Math` on trouve la fonction de calcul de la racine carrée d'un réel :

```
public class Math {
   public static double sqrt(double x) { ... }
   ...
}
```

Remarquons que nous avons déjà utilisé la fonction `Math.sqrt`.

Dans le projet, il peut être utile de décomposer la fonction `main` en plusieurs fonctions pour favoriser la lisibilité du code, par exemple :

```
public class Projet {
   public static void main(String[] args) {
      PremiereAction(...);
      DeuxiemeAction(...);
      ...
   }

   public static void PremiereAction(...) { ... }
   public static void DeuxiemeAction(...) { ... }
   ...
}
```

On comprend enfin ce que signifie le mot-clé `static` associé à la fonction `main`. Ce n'est pas une méthode mais une fonction (la fonction principale) dans la portée de la classe `Projet`. En `Java`, toutes les définitions se font au sein d'une classe.