1. Écrire une classe générique Triplet permettant de manipuler des triplets d'objets d'un même type. On la dotera :
   • d'un constructeur à trois arguments (les objets constituant le triplet),
   • de trois méthodes d'accès getPremier, getSecond et getTroisieme, permettant d'obtenir la référence de l'un des éléments du triplet,
   • d'une méthode affiche affichant la valeur des éléments du triplet.

   Écrire un petit programme utilisant cette classe générique pour instancier quelques objets et exploiter les méthodes existantes.

2. Écrire une classe générique TripletH semblable à celle de l'exercice précédent, mais permettant cette fois de manipuler des triplets d'objets pouvant être chacun d'un type différent. Écrire un petit programme utilisant cette classe générique pour instancier quelques objets et exploiter les méthodes existantes.

3. Soit la classe Point ainsi définie :

```
class Point {

        private double x ; // abscisse

        private double y ; // ordonnee

        public Point (int abs, int ord) {
                x = abs ;
                y = ord ;
        }
       public void affiche () {
              System.out.println ("Coordonnees " + x + " " + y) ;
      }
}
```

Lui ajouter une méthode maxNorme déterminant parmi deux points lequel est le plus éloigné de l'origine et le fournissant en valeur de retour. On donnera deux solutions :

- maxNorme est une méthode statique de Point,
- maxNorme est une méthode usuelle de Point.

Dans la suite, on va décrire différents composants et on va vous demander de proposer une implémentation. La description est (nous l'espérons) suffisamment précise pour avoir une implémentation, mais suffisamment vague pour vous forcer à choisir le type d'objet adéquat (une classe, une classe abstraite, une interface) et le type de relation entre ces objets (héritage, implémentation d'une interface).

On veut faire une application pour gérer le personnel qui s'occupe des différentes formations de l'université.

Une formation de l'université propose une liste de cours. Chaque cours appartient à un domaine (informatique, mathématique, gestion, etc). Pour modéliser ceci, on implémentera une classe `Domaine`, une classe `Cours` et une classe `Formation`. On aura également une classe `Etudiant` dans laquelle on aura pour le moment un nom et une formation. On donnera dans la suite des indications pour compléter ces classes.

On a deux types d'intervenants qui travaillent pour les formations : les maÃ®tres de conférences et les professeurs. La différence est qu'un professeur peut avoir des étudiants en thèse (ici, on fera l'hypothèse qu'un professeur possède un seul étudiant en thèse). Chaque intervenant possède un nom, un numéro de bureau et travaille dans un (et un seul) domaine (intervenant en informatique, intervenant en mathématique, intervenant en gestion). Chaque intervenant peut jouer différents rôle pour la formation : il peut être un responsable de formation, il peut être un enseignant, il peut aussi jouer ces deux rôles. Parmi les maîtres de conférences, on distinguera deux types : ceux qui sont juniors qui ne font qu'enseigner et les seniors qui enseignent et qui ont une responsabilité. De même pour les professeurs, les juniors sont ceux qui enseignent et ont une responsabilité, les seniors ceux qui n'ont qu'une responsabilité. On implémentera ces quatre classes `JuniorMdC`, `SeniorMdC`, `JuniorProf`, `SeniorProf`.

Un responsable de formation possède quatre responsabilités :

— il décide de l'admission ou non d'un étudiant dans la formation
— il décide si un étudiant obtient son diplôme
— il possède la liste des étudiants (on utilisera un tableau de taille `capaciteMax`)
— il possède la liste des cours de la formation (on utilisera un tableau de taille `nbCours`, ce nombre sera le même pour toutes les formations)

Un enseignant peut enseigner un cours de son domaine. Un enseignant doit effectuer ces deux fonctions :

— donner un cours
— évaluer un étudiant pour un cours

Dans la classe `Cours`, on trouvera un nom de cours, la description du cours et l'intervenant qui donne ce cours.

Sur une feuille de papier, donner l'architecture de l'application : dire quelles sont les classes, les interfaces si besoin, et indiquez les variable et méthodes (d'instance ou de classe).