Exercice 1 – Gestion de vols

On gère les vols de la compagnie aérienne *AirCountry*. On se limitera seulement au calcul de distance des vols.

Un vol relie deux aéroports. Les aéroports sont localisés par leurs coordonnées géographiques.

Pour la version actuelle de l'application, nous simplifions la localisation des aéroports sur un seul axe (axe des abscisses) mais qui peut évoluer vers des coordonnées plus complètes (coordonnées GPS par exemple) dans le futur. Par ailleurs, seuls les vols de passagers sont gérés mais d'autres types de vols (cargos, etc.) peuvent être considérés lors de prochaines évolutions de l'application. Les code java suivante est proposé pour cette application :

```
package airCountry;
public class AirCountry {
  public int distanceVol(VolPassager vol){
      Aeroport dept = vol.getDepart();
      Aeroport dest = vol.getDestination();
      return (dept.getCoordonnées() –
               dest.getCoordonnées() );
}}
```

```
package airCountry;
public class Aeroport {
      private int abscisse;
      public int getCoordonnées() {
            return this.abscisse;
      }
}
```

```
package airCountry;

public class VolPassager {
      private Aeroport depart;
      private Aeroport destination;

      public Aeroport getDepart(){
            return this.depart;
      }

      public Aeroport getDestination(){
            return this.destination;
      }
}
```

Travail demandé

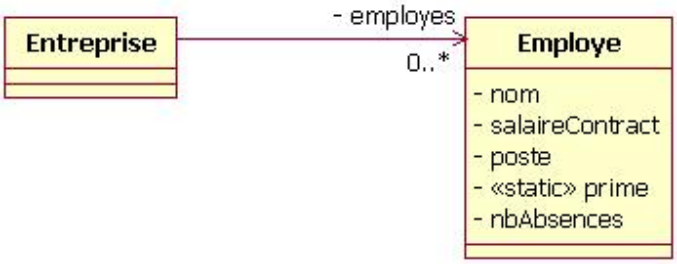
1. Récupérez les trois classes du *Commun* et les ajoutez au projet Java que vous créerez sous Eclipse.
2. Démontrez que cette application est fonctionnellement correcte en écrivant des tests fonctionnels automatiques JUnit.
3. Faites une étude de maintenabilité du code Java proposé. Est-il maintenable ?!
4. En appliquant les principes de délégation et d'inversion de dépendance, refaites la conception de cette application (vous réaliserez le « refactoring de code ») pour améliorer sa maintenabilité à l'égard des évolutions annoncées en réalisant :
   a. Le nouveau diagramme de classes de l'application.
   b. Le diagramme de séquence de calcul de distance d'un vol.
   c. Le code Java correspondant.
5. Rejouez les tests écrits en question 2 pour vous assurer de la non régression de l'application.
6. Vous allez maintenant réaliser la maintenance permettant de représenter la localisation des aéroports en coordonnées géographiques (latitude et longitude). La formule de calcul de distance entre deux points géographiques est donnée.
   a. Calculer le coût de réalisation de cette maintenance. L'OCP est-il respecté ?
   b. Modifiez le diagramme de classes de l'application et schématiser les dépendances.
   c. Coder en Java la maintenance.
   d. Faire des tests automatiques (en vous comparant au site http://www.sunearthtools.com/tools/distance.php).

Exercice 2 – Gestion de personnels

Pour mener à bien ses missions, une entreprise de réparation de matériel informatique emploie deux types de personnels : des techniciens et des ingénieurs en maintenance informatique. Il est précisé que l'entreprise pourrait embaucher des commerciaux si elle décidait de conquérir de nouveaux marchés dans le futur.

Cet exercice se limitera aux seules fonctions d'embauche d'employés et de calcul de salaire de fin de mois. Pour simplifier cette gestion, nous considérons que le nom de technicien est un identifiant de gestion.

Le salaire effectif de fin de mois des techniciens est calculé en appliquant une retenue de 1/60 par absence sur le salaire mensuel (une absence compte pour une demi-journée). Celui des ingénieurs est calculé à partir de leur salaire mensuel contractuel et de la prime mensuelle. Le montant de la prime est le même pour tous les ingénieurs. Contrairement aux techniciens, l'absence des ingénieurs est comptabilisée en journée entière et est déduite du salaire mensuel intégré (salaire contractuel et prime).



Travail demandé

1. Programmer cette application à partir de son diagramme de classes fourni ci-dessus en appliquant le principe de délégation.
2. Faire des tests de validation automatiques avec JUnit.
3. L'entreprise voudrait embaucher des commerciaux qui travailleront à la semaine (leur salaire contractuel est exprimé à la semaine, par contre le salaire touché sera mensuel). Une absence correspond à une semaine non travaillée. Quels impacts aura cette évolution sur l'application ? Le principe de l'OCP est-il mis en œuvre par l'application ? Expliquer.
4. En repartant des spécifications fonctionnelles données en tout début de l'exercice (et sans prise en compte des commerciaux), reconcevoir l'application en appliquant les principes « d'inversion de dépendance » et « d'OCP » (en prévention des évolutions envisagées).
5. Coder l'application Java et rejouer les tests de validation.
6. Quels impacts aura l'embauche de commerciaux à l'application ?
7. Il reste tout de même des dépendances nuisibles à la maintenabilité de l'application, lesquelles ? La solution fera l'objet d'un prochain cours portant sur les « Design Pattern ».