# Compagnie

Solution sur Github : <https://github.com/kilrasemifir/banque-java>

## Exercice 0 : Création du projet

Créer un projet Java.

Créer le package parent compagnie. Tous les packages, classes et interfaces seront dans un package enfants a compagnie.

Créer la classe principale du programme Application avec la méthode statique main et afficher "Hello world".

package company;

class Application {

    public static void main(String[] args){

        Systeme.out.println("Hello world");

    }

}

## Exercice 1 : Les employés

Créer une classe Employe avec les attributs nom, prénom et email. Ajouter les Getters, Setters et constructeurs nécessaires.

## Exercice 2 : La Compagnie

Créer une classe Compagnie dans le même package. Une Compagnie possède un nom et une liste d’employé. Ne pas oublier les getters/setters/constructeurs.

## Exercice 3 : Les développeurs

Créer un package compagnie.competences.

Dans ce package créer une interface Codeur qui définit la méthodes void coder() ;

Créer un package compagnie.postes.

Dans ce package, créer une classe Developpeur qui hérite d’Employe et qui implémente Codeur. La méthode coder affichera un message « [nom prenom] : code »

Dans la classe Compagnie :

* Ajouter une liste codeurs qui prend des Codeurs.
* Ajouter une méthode void ajouterCodeur(Codeur codeur) qui ajoute a un codeur a la liste des codeurs.
* Ajouter une méthode void lancerProjet() qui appel la méthode coder de tous les codeurs.

## Exercice 4 : Les Managers

Créer une interface Manager dans le package compagnie.competences qui définit la méthode void manager() .

Créer une classe ProjetManager dans le package compagnie.postes. Elle hérite de la classe Employe et implémente l’interface Manager. La méthode manager affiche « [nom prénom] : manage »

Ajouter a la classe Compagnie une liste de manager, une méthode pour ajouter un manager (ajouterManager()) et une méthode pour appeler toutes les méthode manager des managers.

## Exercice 5 : L’embauche

Dans la classe Compagnie, créer une méthode void embaucher(Employe employe) qui :

* Si employé est Coder et qu’il y a moins de 10 codeurs, alors il est ajouté à la liste des employés et des codeurs.
* Si employé est Manager et qu’il y a moins de 3 managers, alors il est ajouté aux employés de la compagnie ainsi qu’aux managers.

## Exercice 6 : Lead-Dev

Créer une classe LeadDeveloppeur qui hérite de développeur mais qui implémente aussi Manager.

La méthode manager affiche « [nom prenom] : manager technique ».

En l’embauchant, il doit être considéré comme un Codeur et un Manager mais doit n’être qu’un seul employé (attention aux doublons dans la liste des employés).

## Exercice 7 : Administration

Créer une interface Administratif dans le package compagnie.competences qui définie la méthode administrer().

Créer une classe AgentAdministratif qui hérite de la classe Employe et qui implémente les interfaces Administratif et Manager. La méthode administrer() affiche « [nom prénom] : administre.

Dans la classe Compagnie, ajouter une liste d’Administratifs. Il ne peut y avoir que 3 administratifs. Ajouter une méthode ajouterAdministratif(Administratif administratif) qui ajout un administratif a la liste des administratif . Ajouter une méthode administrer() qui appel toutes les méthodes administrer() des Administratifs.

## Exercice 8 : Statistique de la compagnie

Réécrire la méthode toString() qui retourne le format suivent :

Compagnie <Nom de la compagnie>

Nombre de codeurs : <codeurs>

Nombre de managers : <managers>

Nombre d’administratif : <administratifs>

Effectif Total : <nombre d’employé total>