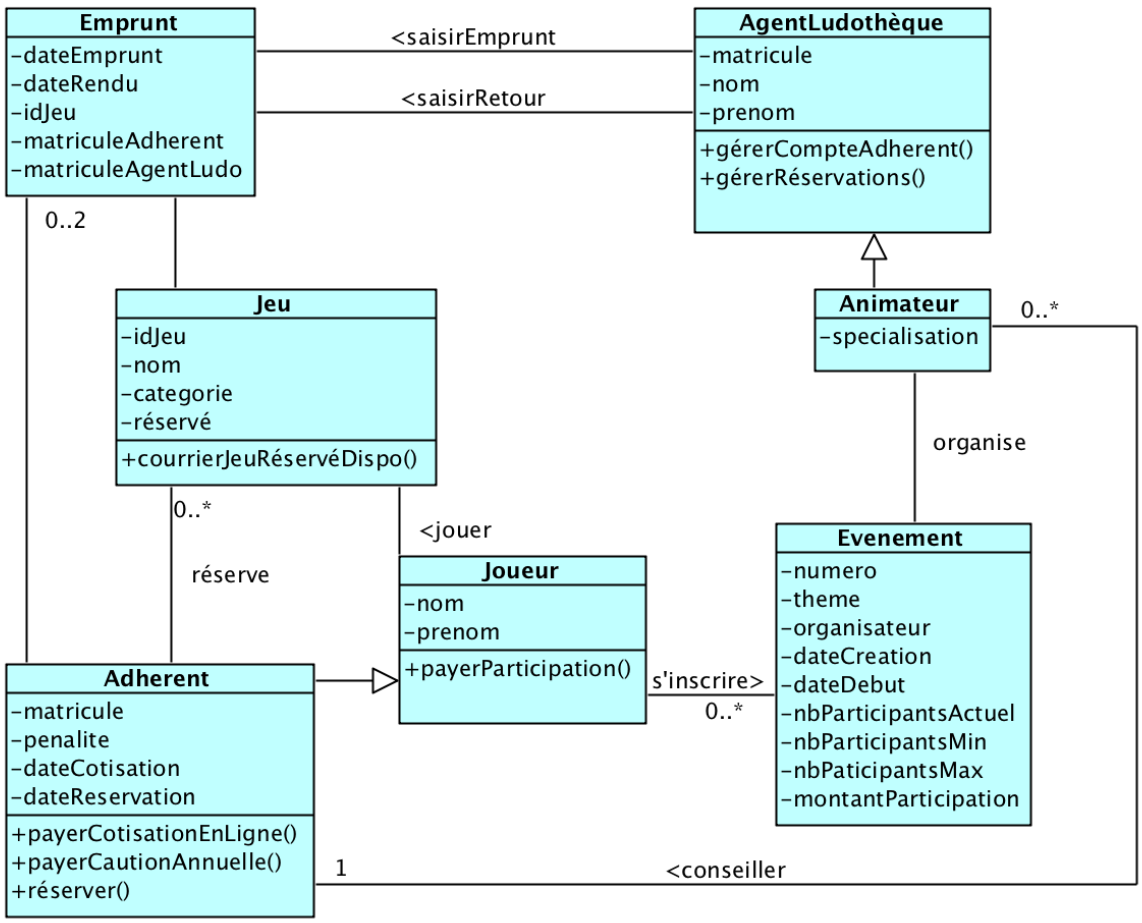
TD1 – Premiers diagramme de Classes

Exercice 1 – LaPoste

L'application concerne les courriers *recommandés*. Un facteur distribue des courriers en recommandé aux habitants de la zone géographique qui lui est affectée. Les habitants sont aussi associés à une zone géographique. Les recommandés sont de deux sortes : lettres ou colis. Comme plusieurs facteurs peuvent intervenir sur la même zone, on souhaite garder une trace du facteur qui a distribué le recommandé avec la date et l'heure, en plus du destinataire.

Donner les classes et leurs relations, sans attribut ni méthode.

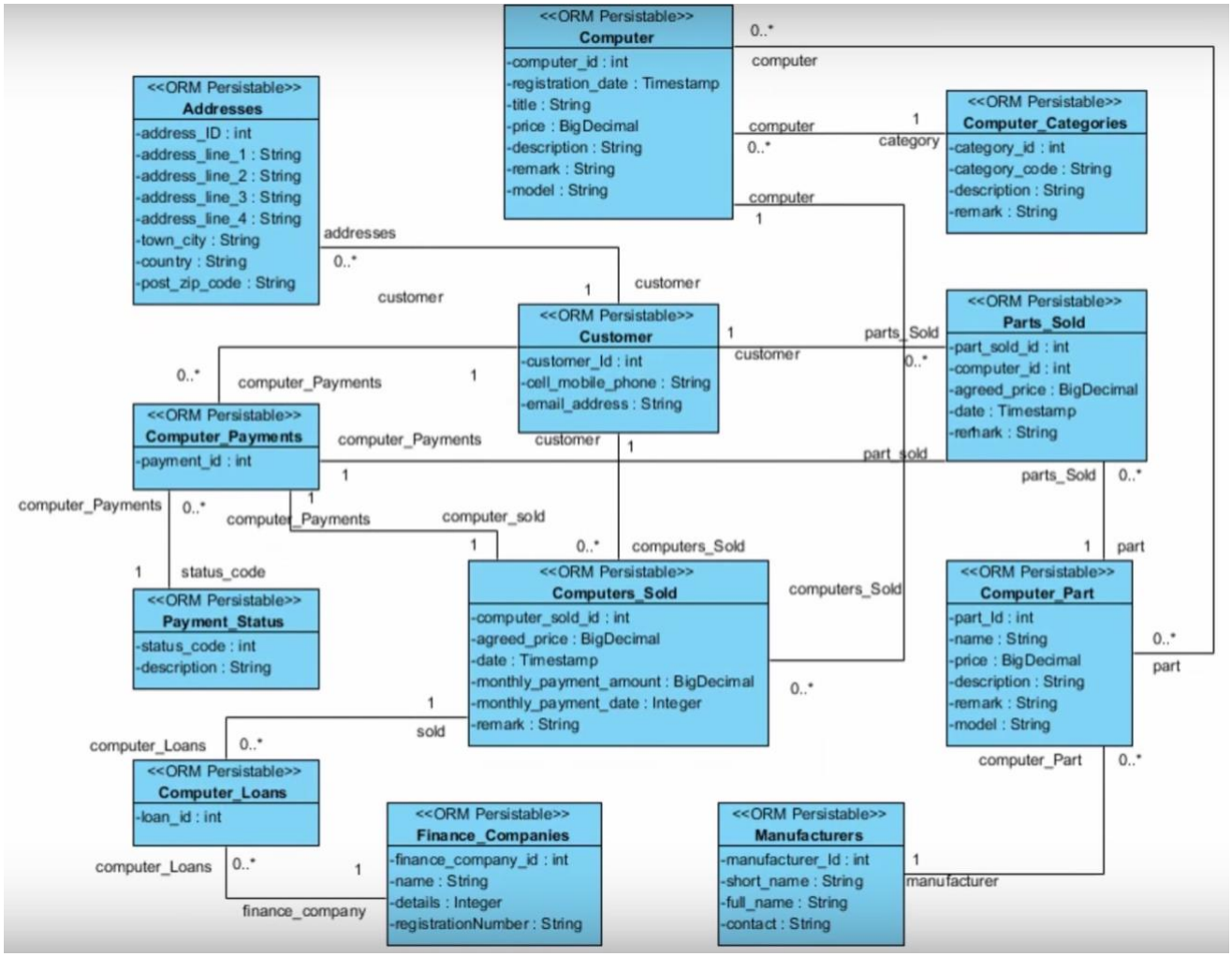
Exercice 2 – DCL mystère avec erreurs



De quoi parle de diagramme ? quelles erreurs y voyez-vous ?

Exercice 3 – Analyse d'un DCL

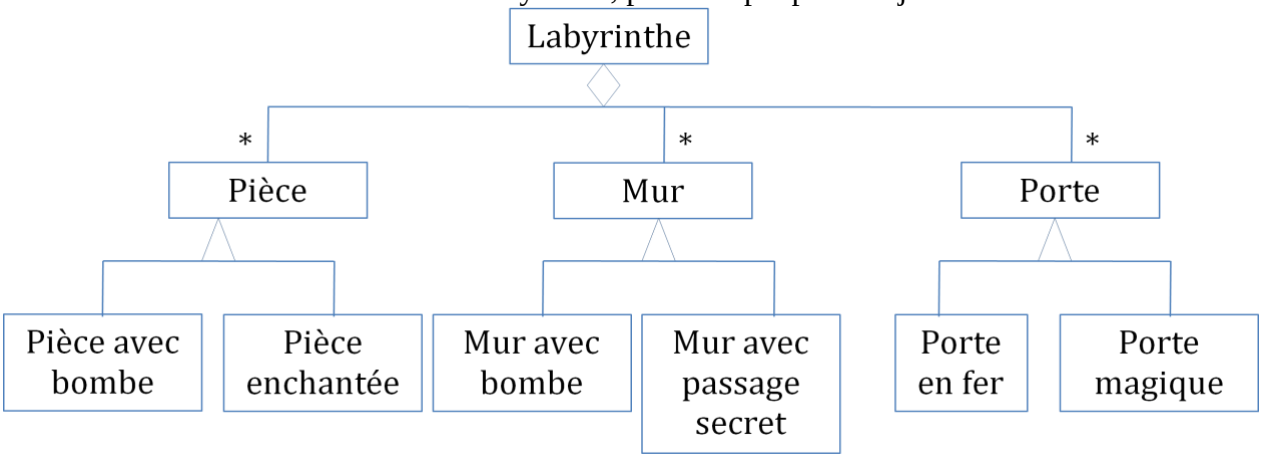
Voici le DCL proposé pour une application de ventes d'ordinateur :



3.1- Donner en langue naturelle, toutes les **explications** fournies par ce modèle.
3.2- Le **nommage** des différents éléments est-il correct (règles UML d'écriture) ?
3.3- Que signifie **« ORM Persistable »** ?
3.4- A quoi correspond la classe **Computer_Payments** ? Pourquoi n'a-t-elle qu'un ID et rien d'autre ?
3.5- Dans ce modèle d'analyse, il n'y a **aucune méthode** : est-ce normal, docteur ?

Exercice 4 – Labyrinthe

On considère le modèle suivant d'un labyrinthe, par exemple pour un jeu :



Votre collègue définit le code Java qui crée un labyrinthe de ce type (la classe Labyrinthe étant définie par ailleurs) :

```
class JeuLabyrinthe {
     public Labyrinthe creerLabyrinthe(){
          Labyrinthe unLabyrinthe = new Labyrinthe();
          Piece unePiece = new Piece(1);              // 1 = identifiant
          Piece uneAutrePiece = new Piece(2);
          Porte unePorte = new Porte( unePiece , uneAutrePiece );

          unLabyrinthe.addPiece( unePiece );
          unLabyrinthe.addPiece( uneAutrePiece );

          unePiece.setSide( North, new Mur() );
          unePiece.setSide( East, unePorte );
          unePiece.setSide( South, new Mur() );
          unePiece.setSide( West, new Mur() );

          uneAutrePiece.setSide( North, new Mur() );
          uneAutrePiece.setSide( East, new Mur() );
          uneAutrePiece.setSide( South, new Mur() );
          uneAutrePiece.setSide( West, unePorte );

          return unLabyrinthe;
    }
…
}
```

4.1. Dans le code Java, il y a plus d'informations que dans le digramme de classes : quels sont les oublis dans la modélisation ? Donner le **diagramme de classes** avec les relations manquantes.

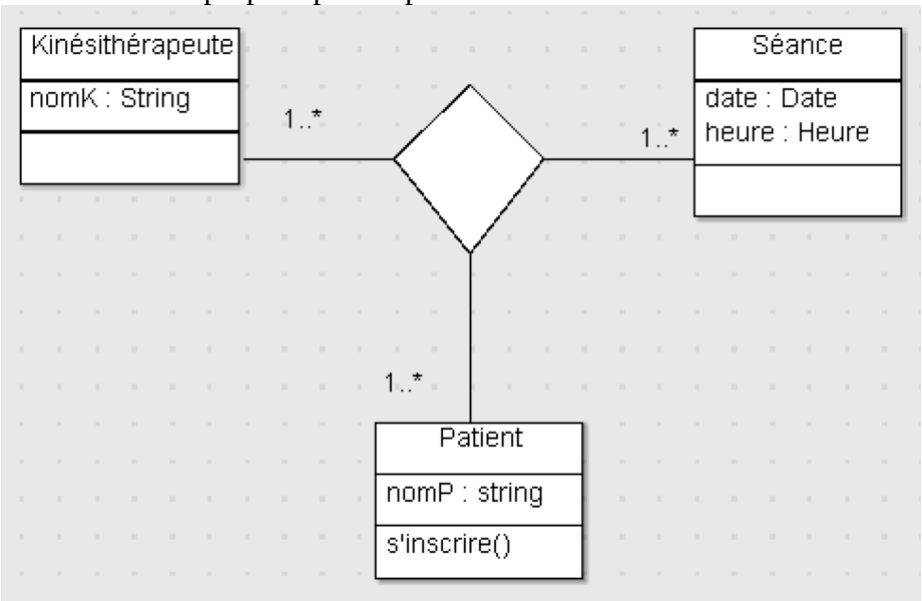
4.2. Ajouter les **attributs** qui figurent dans le code Java et qui manquent dans le DCL.

4.3. Quel problème de **cohérence** pose ce code ? Comment le résoudre ?

Exercice 5 – Kiné

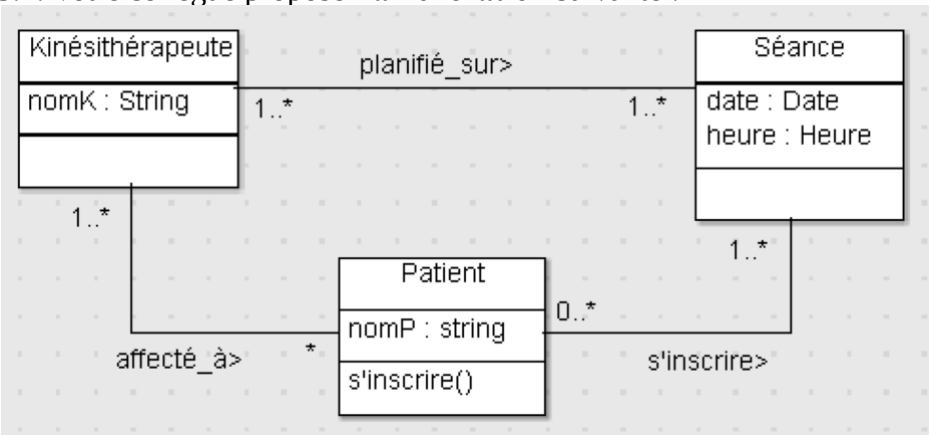
Dans une clinique, un patient s'inscrit à des séances de rééducation avec le kinésithérapeute qui a été affecté pour leur traitement. En septembre, les kinésithérapeutes se positionnent sur des séances définies sur toute l'année. Il peut y avoir plusieurs kinésithérapeutes par séance, un kiné peut s'occuper de plusieurs patients sur une même séance (pour par ex., des séries d'exercices effectués en autonomie).

Voici le modèle proposé pour répondre à ce besoin :



5.1. Ce modèle n'est **pas satisfaisant**, pourquoi ?

5.2. Votre collègue propose **l'amélioration** suivante :



Qu'en pensez-vous ? Si c'est incorrect, que proposeriez-vous ?