BCOO Sujet TD1

PreReq	1. Je sais programmer, j'ai conscience qu'il faut réfléchir avant de se lancer dans le codage. J'ai des notions de structure de données. 2. Je connais les concepts principaux du Diagramme de Classes : classe, attribut, attribut dérivé, identifiant, association, composition, multiplicité, classe-association. 3. Je connais les conventions d'écriture du Diagramme de Classes
ObjTD	Comprendre ce qu'est un **modèle**.
Durée	**1 ou 2 séance(s)** de 1,5h (selon le niveau initial en UML)

1. Rappel du cours

Dans le diagramme de classe suivant, repérez les différents éléments :

1. classe
2. nom de classe
3. attribut
4. association
5. cardinalité


dc1

Quel type élément n'avez-vous identifiéssur le diagramme ? Saurez-vous trouver son nom ?

2. Diagrammes de classes simples

Réalisez les diagrammes de classes suivants :

- Examens
  - Les étudiants possèdent un numéro d'étudiant, un nom, un prénom, une date de naissance. Ils suivent des cours (titre, code du module),
  - Les examens concernent un cours donné. Chaque examen a lieu à une certaine date et possède un coefficient,
  - Une note est attribuée à un étudiant, par examen,
  - Les cours sont enseignés par un enseignant (nom, prénom).
- Ordinateurs portables

Un portable possède un clavier,

- Un clavier peut-être de type "azerty" ou "querty",
- Un clavier possède des touches,
- Un portable a éventuellement un propriétaire qui a lui même un nom et un prénom,
- Un portable a un prix d'achat et une valeur actuelle (souvent différente).

3. Tour-opérateur

3.1. Enoncé

Des tour-operateurs (Fram, Jet-Tours, Nouvelles Frontières…) proposent des voyages. Chaque TO est caractérisé par un identifiant, un nom et une adresse.

Chaque voyage est rattaché à un seul TO et est défini par un identifiant, un libellé, un lieu de départ et de destination, une date de départ et d'arrivée ainsi qu'un prix de base.

Différentes options (chambre climatisée, pension complète, voyages en 1ère classe…) peuvent être choisies pour les différents voyages avec un montant différent selon celui-ci (par exemple l'option « Pension complète » coûtera plus cher pour un voyage à Paris que pour un voyage au Vietnam).

Des clients (identifiant, nom, prenom, adresse) passent des commandes pour des voyages, chaque commande ne concernant qu'un voyage. Une commande est définie par un numéro, une date et un état. À chaque commande est rattachée un certain nombre de voyageurs caractérisés par un identifiant, un nom, un prénom, un sexe et une date de naissance.

3.2. Exercices

1. Réalisez le Diagramme de Classes de ce cas d'étude
2. Ajoutez aux classes les méthodes qui permettront de réaliser les fonctionnalités attendues suivantes :
   - Calculer le prix d'un Voyage
   - Calculer le prix moyen d'un voyage pour un Tour-opérateur
   - Afficher la liste des options d'un Voyage
   - Afficher la liste des commandes en cours (à venir) d'un Client

4. Reverse Java → UML

4.1. Enoncé

Soient les codes Java suivants :

Les codes complets ont été générés automatiquement, ils sont donc très "verbeux"…

Accident.java (version complète ici)

```
package accidents;
import java.sql.Date;
import java.sql.Time;

public class Accident
{
  private String id;
  private String description;
  private Date date;
  private Time time;
  private String other_details;

  private Employee employee;
  private AccidentType accidentType;
  private SeriousnessLevel seriousnessLevel;

  public Accident(String aId, String aDescription, Date aDate, Time aTime, String
aOther_details, Employee aEmployee, AccidentType aAccidentType, SeriousnessLevel
aSeriousnessLevel)
  { ... }
  // gettes and setters
  ...
  public Employee getEmployee(){ ... }
  public AccidentType getAccidentType(){ ... }
  public SeriousnessLevel getSeriousnessLevel(){ ... }
  public boolean setEmployee(Employee aEmployee){ ... }
  public boolean setAccidentType(AccidentType aNewAccidentType){ ... }
  public boolean setSeriousnessLevel(SeriousnessLevel aNewSeriousnessLevel){ ... }
}
```

AccidentType.java (version complète ici)

```
package accidents;
public class AccidentType
{
  private String code;
  private String description;

  public AccidentType(String aCode, String aDescription){ ... }
  ...
}
```

Employee.java (version complète ici)

```
package accidents;
public class Employee
{
  private String id;
  private String department;
  private String name;
  private String supervisor;
  private String other_employee_details;

  private List<Accident> accidents;

  public Employee(String aId, String aDepartment, String aName, ...){ ... }
  public Accident getAccident(int index){ ... }
  public List<Accident> getAccidents(){ ... }
  public int numberOfAccidents(){ ... }
  public boolean hasAccidents(){ ... }
  public int indexOfAccident(Accident aAccident){ ... }
  public static int minimumNumberOfAccidents(){return 0;}
  public Accident addAccident(String aId, String aDescription, Date aDate, ...){ ... }
  public boolean addAccident(Accident aAccident){ ... }
  public boolean removeAccident(Accident aAccident){ ... }
}
```

SeriousnessLevel.java (version complète ici)

```
package accidents;

public class SeriousnessLevel
{
  private String code;
  private String description;

  public SeriousnessLevel(String aCode, String aDescription){ ... }
}
```

4.2. Exercice

Réalisez le Diagramme de Classes correspondant (avec associations et méthodes)