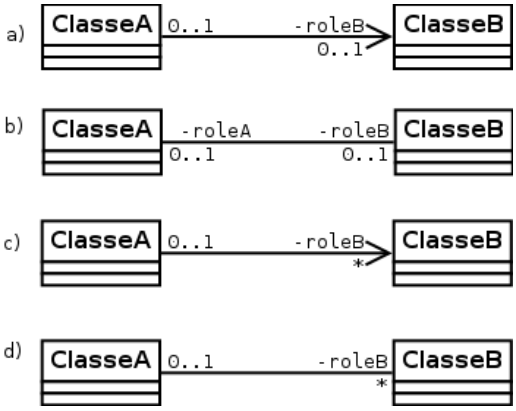
TD révisions UML - JAVA

I) Partie UML

1) Les instances décrivent-elles des méthodes ? Où sont décrites les méthodes ?
2) Quelle est la différence entre une opération et une méthode ? Dans quelle situation cette différence est-elle notable ?
3) Citez un diagramme par type de diagrammes.
4) Proposez une implémentation java pour chacune des situations suivantes :



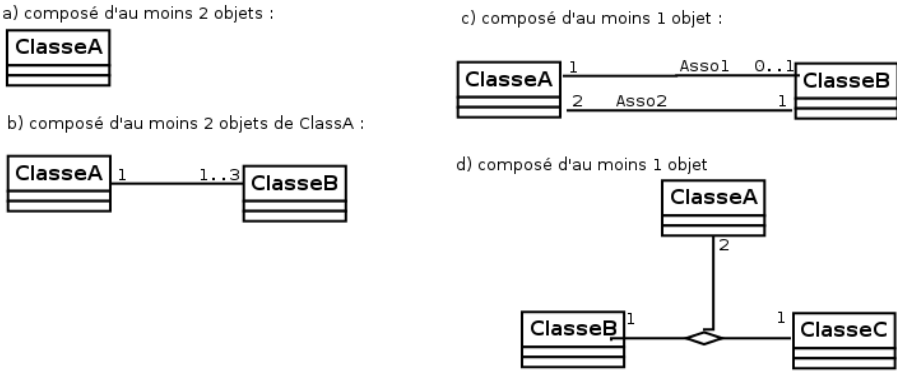
5) Faites de la rétro-ingénierie : proposez le diagramme de classes de ce bout de code :

```
public class Panier {private Proprietaire sonProprio ;
                     private ArrayList<Article> sesArticles;
                     private float montantTotal;
                     public Panier(Proprietaire p){...}
                     public void ajouterArticle(){...}

                     public void payer(){...}

                   ... }
```

6) Proposez les plus petits diagrammes d'objets respectant les diagrammes de classe suivant :



7) On cherche à modéliser la structure de documents hiérarchisés. De tels documents contiennent deux types d'éléments : les paragraphes et les sections. Les sections peuvent contenir des paragraphes ainsi que des sous-sections, elles-même structurées de la même manière. Par exemple, dans l'exemple ci-dessus, la section de plus haut niveau a pour titre "TD", et elle contient le paragraphe "Les exercices ne sont pas indépendants" ainsi que deux sous-sections, chacune correspondant à un exercice. La section correspondant à l'exercice 1 a pour titre "Exercice 1" et contient deux sous sections, une par question. On doit pouvoir afficher tous les types d'éléments, et en ajouter aux sections et sous sections.

II) Partie JAVA

1) Héritage et constructeur

L'exercice consiste seulement à deviner la sortie du programme ci-dessous (la méthode `main` de la classe `EssaiConstructeurs`)

```
class A {
    A() {
        System.out.println("bonjour de A");
    }
}

class B extends A {
    boolean verite;
    int valeur;

    B() {
        System.out.println("constructeur B()");
    }

    B(int valeur) {
        this();
        this.valeur = valeur;
        System.out.println("constructeur B(int)");
    }

   B(boolean verite) {
        this.verite = verite;
        System.out.println("constructeur B(boolean)");
    }

    B(boolean verite, int valeur) {
        this(valeur);
        this.verite = verite;
        System.out.println("constructeur B(boolean, int)");
    }

    public String toString() {
        return "B : (" + verite + ", " + valeur +")\n";
    }
}

class EssaiConstructeurs {
    public static void main(String[] argv) {
        B b = new B(true);
        System.out.println(b);
        b = new B(false, 5);
        System.out.println(b);
    }
}
```

2) Classe abstraite

Ecrivez les quatre classes suivantes :

- La classe `Animal` est abstraite et déclare uniquement une méthode abstraite nommée action, sans paramètre et qui ne retourne rien.
- La classe `Chien` hérite de `Animal` et définit la méthode `action` qui écrit à l'écran "J'aboie".
- La classe `Chat` hérite de `Animal` et définit la méthode `action` qui écrit à l'écran "Je miaule".
- La classe `EssaiChat` contient trois champs statiques :
    - un champ statique pour un attribut de type `java.util.Random` qui est initialisé dès sa définition
    - une méthode statique nommée `tirage` sans paramètre qui retourne un `Animal` qui a une chance sur deux d'être un `Chat` et une chance sur deux d'être un `Chien`.
    - une méthode `main` qui utilise la méthode `tirage` et invoque la méthode `action` sur l'animal obtenu par cette méthode.

3) Interface

Soit l'interface Dessinable et les classes représentant un rectangle, un cercle et un triangle. Ecrivez une classe TestDessin qui remplit un tableau contenant un triangle, deux rectangles et un cercle puis dessine chacun de ces objets.

```
interface Dessinable{
   public void dessiner();
}


class Rectangle implements Dessinable{
   public void dessiner(){
      System.out.println("je dessine un rectangle");
       }
}

class Cercle implements Dessinable{
   public void dessiner(){
      System.out.println("je dessine un cercle");
       }
}

class Triangle implements Dessinable{
   public void dessiner(){
      System.out.println("je dessine un triangle");
       }

}
```