Examen UML Java

Il vous est demandé d'apporter un soin particulier à la présentation de votre copie.
Le code généré devra être bien indenté et commenté.
*Par ailleurs, si le texte du sujet, de ses questions ou de ses annexes, vous conduit à formuler une ou plusieurs hypothèses, il vous est demandé de la (ou les) mentionner **explicitement** dans votre copie.*

I. Question de cours

Expliquer à quoi servent les concepts Java de *classe abstraite* et d'*interface* et en quoi ils sont différents.

II. Étude de code

Examiner le code java en Annexe I.

Il comporte

- une erreur de syntaxe
- une erreur de conception

Corriger ces erreurs.

III. Etude de cas

On souhaite réaliser une application pour gérer une liste de collectivités locales (communes, communautés de communes ou d'agglomération, département).

III.a Analyser les besoins de la solution

Pour être en mesure de réaliser les diagrammes UML demandés, vous devez vous baser sur les informations ci-dessous, mais également des informations de la partie III.b.

Plusieurs intervenants devraient pouvoir utiliser l'application :

- Pour commencer, le personnel administratif de la région devrait pouvoir créer une collectivité locale, que celle-ci soit une commune ou un département. Il doit être possible, si souhaité, de créer les communes pendant la création d'un département. Il doit également être possible d'ajouter ultérieurement des communes à un département.
- Par ailleurs, toute personne devrait pouvoir consulter les informations d'une commune (que celle-ci soit une commune, une communauté de communes ou une communauté d'agglomération).

Réalisez les diagrammes UML suivants :

1. Le diagramme de cas d'utilisation
2. Le diagramme de classes (basé sur les informations de la partie III.b).

III.b Écriture de code

Une collectivité locale est décrite intrinsèquement par son nom.
Deux autres données sont intéressantes : sa population et sa superficie. Ces données peuvent être soient intrinsèques dans le cas d'une commune, soient calculées dans le cas d'une communauté de communes ou d'un département, comme la somme des communes qui les composent.

1. Écrire une classe générique *CollectiviteLocale*, qu'on ne pourra pas instancier, mais qui décrira les méthodes permettant de connaitre la collectivité : *getNom()*, *getPopulation(), getSuperficie()*
   Écrire une méthode *getDensite()* qui calculera la densité de population de cette collectivité locale.
2. Écrire une classe *Commune* qui est une collectivité locale et qui sera construite à partir de son nom, sa population et sa superficie.
3. Une communauté de communes est une collectivité locale qui est composée de communes. Écrire une classe *CommunautesCommunes* qui sera construite à partir de son nom et d'un tableau de communes. Son constructeur devra donc avoir la signature suivante :
   CommunautesCommunes (String nom, Commune[] collectivite)
   Sa population et sa superficie sont la somme respective des populations et des superficies des communes qui la composent.
4. Un département est une collectivité locale qui est composée de communauté de communes. Écrire une classe *Departement* sur le même principe qu'une communauté de communes.
5. Une communauté d'agglomération est une communauté de communes qui a au moins 50000 habitants. Écrire une classe *CommunautesAgglomeration* dont le constructeur vérifie cette condition ou génère une exception.
6. Écrire un programme principal permettant de tester votre code en instanciant :
   - Plusieurs communes,
   - Au moins une communauté de communes et une communauté d'agglomération
   - Un département dont vous afficherez la population et la densité.

Annexe I : fichier Bouteille.java

```
abstract class Bouteille {

      public void deboucher() {
            System.out.println(this);

      public String toString() {
            return "clink";
      }

      public static void main(String args[]) {
            Bouteille[] casier = new Bouteille[5];
            casier[0] = new BouteilleChampagne();
            casier[1] = new BouteilleSoda();
            casier[2] = new BouteilleWhisky();
            casier[3] = new BouteilleEau();
            BouteilleSansAlcool b = (BouteilleSansAlcool) casier[3];
            b.deboucher();
            for (int i=0; i<casier.length; i++)
                  casier[i].deboucher();
      }
}

class BouteilleSansAlcool extends Bouteille { }

class BouteilleSoda extends BouteilleSansAlcool {
      public String toString() { return "psshit"; }
}

class BouteilleEau extends BouteilleSansAlcool {
      public String toString() { return "eau"; }
}

class BouteilleAlcool extends Bouteille {
      public String toString() { return "blup"; }
}

class BouteilleVin extends BouteilleAlcool {
      public String toString() { return "pop"; }
}

class BouteilleChampagne extends BouteilleVin {
      public String toString() { return "pan"; }
}

class BouteilleWhisky extends BouteilleAlcool {}
```