

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.

2 points

La classe abstraite est une classe à part entière qui peut être héritée.

Le code non écrit doit être codé entièrement pour que la classe héritée puisse être instanciée.

Une interface n'est pas une classe à part entière.

L'interface garantit le respect de sa définition et non l'obligation d'en coder toutes les fonctions non écrites.

Une interface ne contient que des champs static et final.

Une classe peut hériter de plusieurs interfaces alors que l'héritage multiple est impossible, donc aussi pour les classes abstraites.

II. Étude de code

Examiner le code java en Annexe I.

Il comporte

- une erreur de syntaxe
- une erreur de conception

Corriger ces erreurs.

1 points : Voir Annexe I

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 soit intrinsèques dans le cas d'une commune, soit 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 connaître 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 *CommunauteCommunes* qui sera construite à partir de son nom et d'un tableau de communes. Son constructeur devra donc avoir la signature suivante :
CommunauteCommunes (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 { // La classe n'a pas à être abstract
                        et doit être public
    public void deboucher() {
        System.out.println(this);
    } // MANQUE ACCOLADE
    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();
        // la dernière case ne contient rien car non instanciée
        Donc cela provoquera une erreur
        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"; }
}

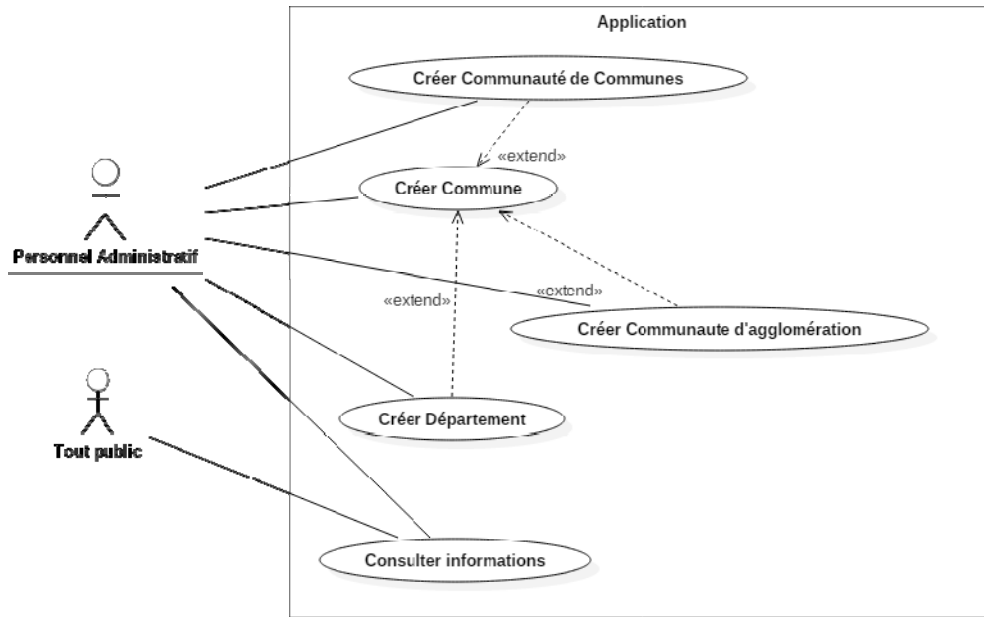
class BouteilleVin extends BouteilleAlcool {
    public String toString() { return "pop"; }
}

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

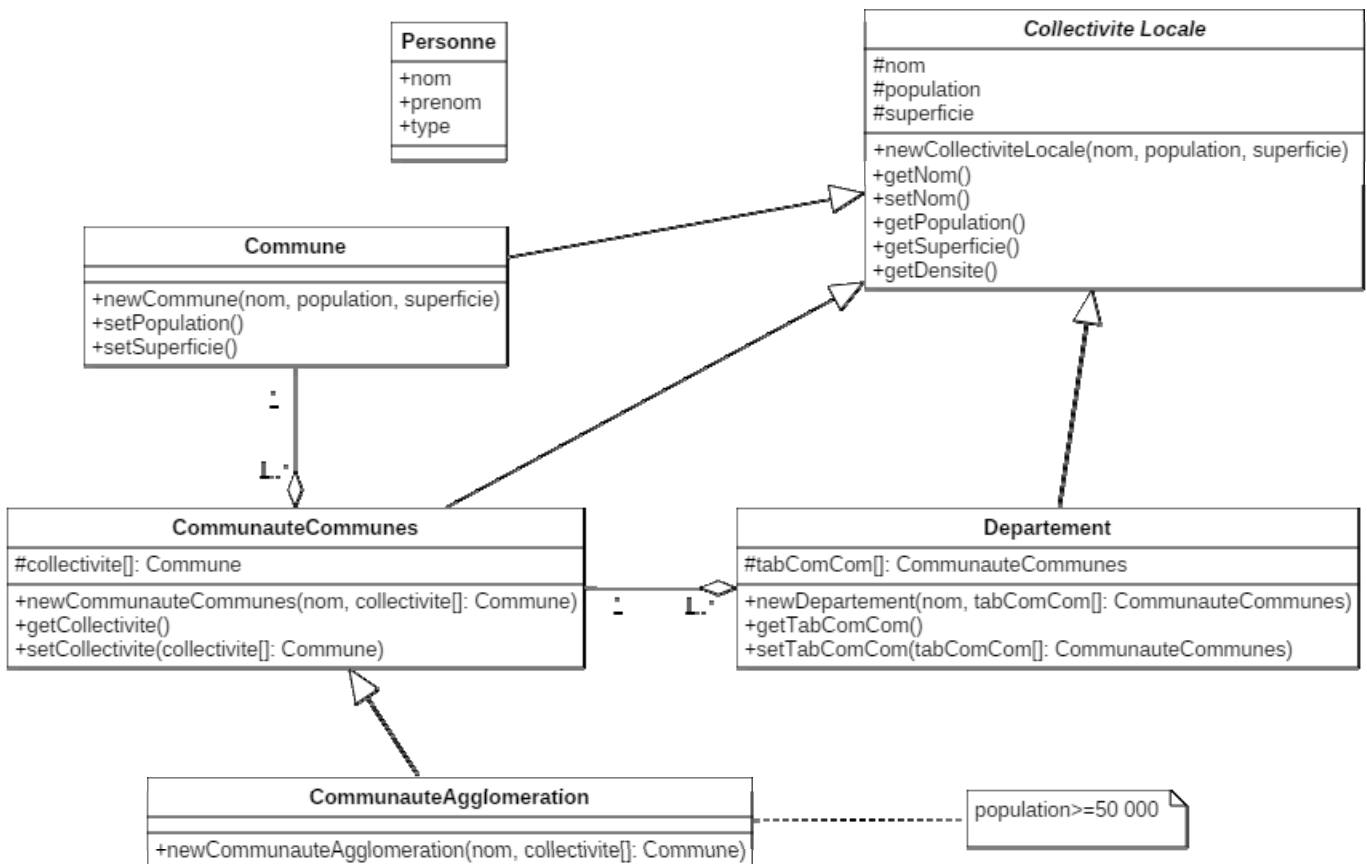
class BouteilleWhisky extends BouteilleAlcool {}

```

III.a.1 3 pts



III.b.1 4 pts



2 points

```
public abstract class CollectiviteLocale {
    protected String nom;
    protected int superficie;
    protected int population;

    public CollectiviteLocale(String nom, int superficie, int population) {
        super();
        this.nom = nom;
        this.superficie = superficie;
        this.population = population;
    }

    public String getNom() {
        return nom;
    }

    public int getPopulation() {
        return population;
    }

    public int getSuperficie() {
        return superficie;
    }

    // Calcul de la densite de population
    public double getDensite() {
        return (double) (population / superficie);
    }

    public void setNom(String nom) {
        this.nom = nom;
    }
}
```

1 points

```
public class Commune extends CollectiviteLocale {

    public Commune(String nom, int population, int superficie) {
        super(nom, population, superficie);
    }

    public void setSuperficie(int superficie) {
        this.superficie = superficie;
    }

    public void setPopulation(int population) {
        this.population = population;
    }

    public String toString() {
        return "Commune [nom=" + nom + ", superficie=" + superficie
            + ", population=" + population + "];"
    }
}
```

1.5 points

```
import java.util.Arrays;

public class CommunautesCommunes extends CollectiviteLocale {

    protected Commune[] collectivite;

    public CommunautesCommunes(String nom, Commune[] collectivite) {
        super(nom, 0, 0);
        this.collectivite = collectivite;
        population = 0;
        superficie = 0;
        for (Commune com : collectivite) {
            population += com.getPopulation();
            superficie += com.getSuperficie();
        }
    }

    @Override
    public String toString() {
        return "CommunautesCommunes [collectivite="
            + Arrays.toString(collectivite) + ", nom=" + nom
            + ", superficie=" + superficie + ", population=" + population
            + "]";
    }
}
```

2 points

```
import java.util.Arrays;

public class Departement extends CollectiviteLocale {

    protected CommunautesCommunes[] tabComCom;

    public Departement(String nom, CommunautesCommunes[] tabComCom) {
        super(nom, 0, 0);
        this.tabComCom = tabComCom;
        for (CommunautesCommunes comCom : tabComCom) {
            population += comCom.getPopulation();
            superficie += comCom.getSuperficie();
        }
    }

    @Override
    public String toString() {
        return "Departement [tabComCom=" + Arrays.toString(tabComCom)
            + ", nom=" + nom + ", superficie=" + superficie
            + ", population=" + population + "]";
    }
}
```

2 points

```
import java.util.Arrays;
```

```
public class CommunautesAgglomeration extends CommunautesCommunes {

    public CommunautesAgglomeration(String nom, Commune[] collectivite) {
        super(nom, collectivite);
        try {
            if (population < 50000)
                throw new MonException("Population insuffisante");
        } catch (MonException e) {
            e.printStackTrace();
        }
    }

    public String toString() {
        return "CommunautesAgglomeration [collectivite="
            + Arrays.toString(collectivite) + ", nom=" + nom
            + ", superficie=" + superficie + ", population=" + population + "];"
    }
}
```

```
public class MonException extends Exception {

    public MonException () {
        super();
    }

    public MonException (String s) {
        super(s);
    }
}
```

2 points

```
public class Main {

    public static void main(String[] args) {
        Commune paris = new Commune("Paris", 5000000, 105);
        Commune cergy = new Commune("Cergy", 200000, 50);
        Commune levallois = new Commune("Levallois", 100000, 20);
        Commune clichy = new Commune("Clichy", 30000, 15);
        System.out.println(paris + " Densite : " + paris.getDensite());
        System.out.println(cergy + " Densite : " + cergy.getDensite());
        System.out.println(levallois + " Densite : " + levallois.getDensite());
        System.out.println(clichy + " Densite : " + clichy.getDensite());

        Commune [] tabCom = {paris, cergy, levallois, clichy};
        CommunautesCommunes cc = new CommunautesCommunes("Grand Paris", tabCom);
        System.out.println(cc);
        CommunautesAgglomeration ca1 = new CommunautesAgglomeration("NewParis", tabCom);
        System.out.println(ca1);

        CommunautesCommunes [] tabComCom = {cc};
        Departement depParis = new Departement("ParisDep", tabComCom);
        System.out.println(depParis);
        Commune [] tabCom2 = {clichy};
        CommunautesAgglomeration ca2 = new CommunautesAgglomeration("NewParis", tabCom2);
        System.out.println(ca2);
    }
}
```