

**Nom:**

**Prénom:**

**Matricule:**

---

**EXAMEN**

**INFORMATIQUE POUR LA GESTION II  
(INFO-D-201)**

**2ème Année du Bachelier  
Ingénieur de Gestion**

13 Janvier 2011

Nom:

Prénom:

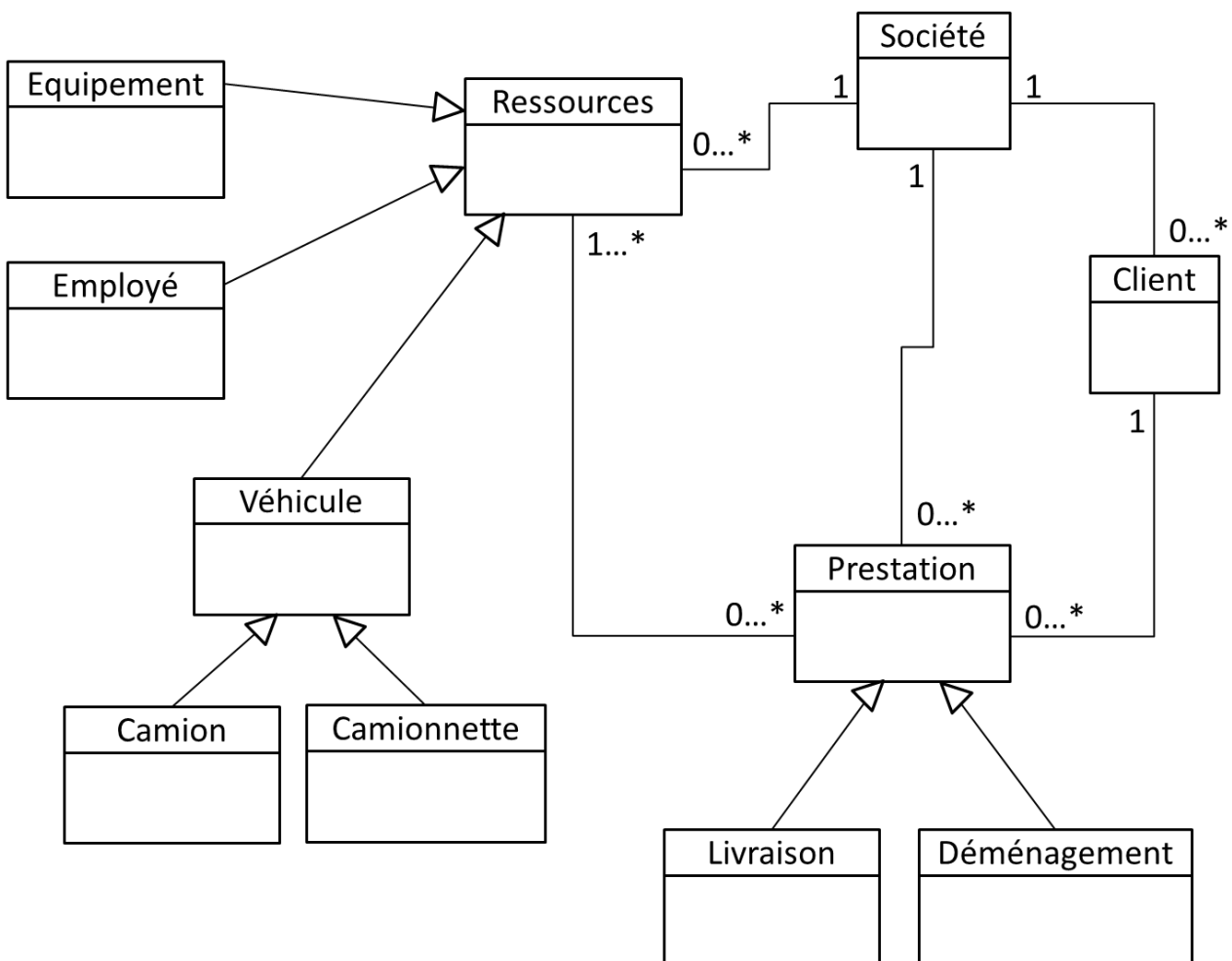
Matricule:

**Question n°1: UML (/3)**

Vous devez développer un programme pour aider une entreprise de transport à gérer ses ressources et leur affectation aux livraisons et déménagements qu'elle doit réaliser pour ses clients à des dates bien spécifiées. Pour le transport, l'entreprise possède des camions et des camionnettes, qui outre leur capacité, diffèrent par le type de permis nécessaire pour les conduire, le mode de chargement, ou encore la procédure à suivre pour en assurer la maintenance. Les équipes qui effectuent les livraisons et les déménagements peuvent varier en nombre de personnes suivant les cas (les équipes sont généralement plus importantes pour les déménagements que pour les livraisons qui sont parfois réalisées par une seule personne). La plupart des déménagements et certaines livraisons nécessitent en outre divers équipements tels que des élévateurs. Pour assurer la planification de ses livraisons et déménagements, l'entreprise doit ainsi pouvoir disposer d'une vue sur la disponibilité de chaque ressource à tout moment.

Veillez tracer le diagramme UML de base de cette application en y représentant les classes et leurs relations précises (il n'est pas nécessaire d'énumérer les attributs et méthodes de chaque classe).

**REPONSE :**



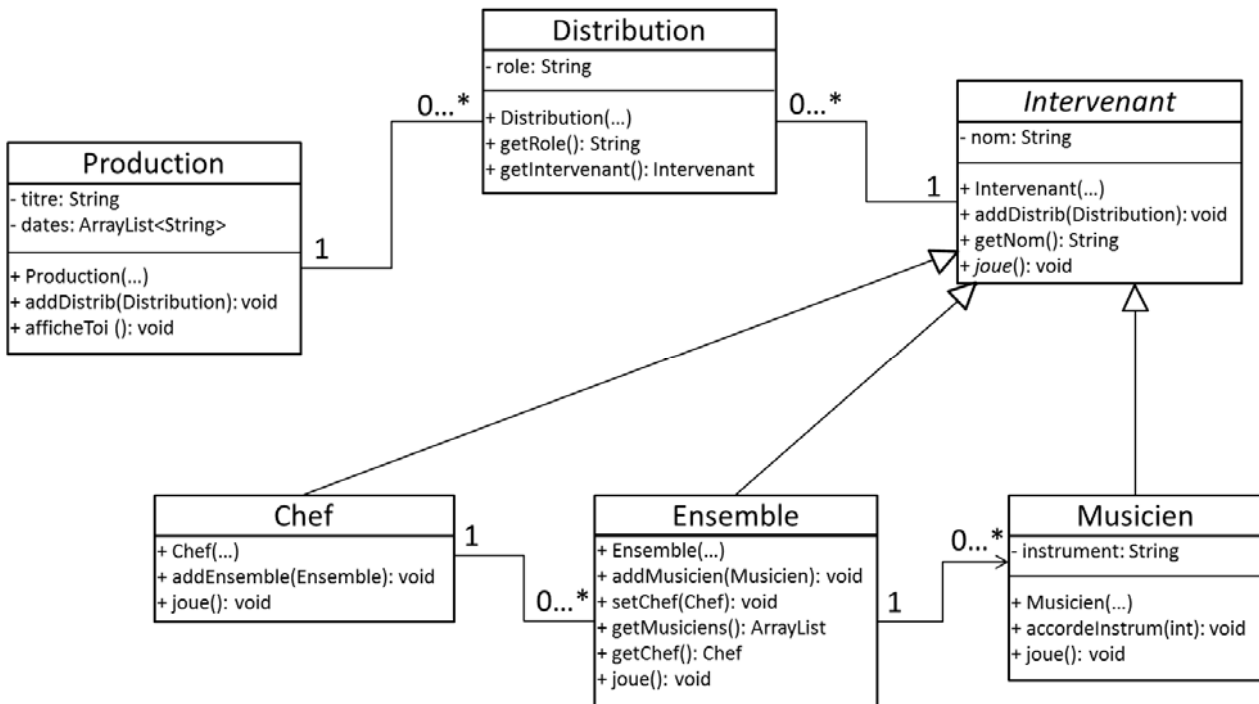
Nom:

Prénom:

Matricule:

**Question n°2: UML → Code Java (/2)**

Veillez produire le code Java minimum correspondant au diagramme de classes ci-dessous, avec toutes les déclarations de classes, attributs et méthodes représentés dans le diagramme. Vous devez développer le corps des constructeurs ainsi que des méthodes 'get', 'add' et 'set', mais pas celui des autres méthodes (joue(), afficheToi(), et accordeInstrum(int)). Votre code ne devrait pas « réaliser » quelque chose (puisque certaines méthodes ne seront pas implémentées et qu'il n'y aura pas de 'main'), mais il devrait au minimum pouvoir être compilé sans erreur.

**REPONSE :**

```

import java.util.ArrayList;
public class Production {
    private String titre;
    private ArrayList<String> dates;
    private ArrayList<Distribution> distributions;
    public Production(String titre, ArrayList<String> dates){
        this.titre = titre;
        this.dates = dates;
        this.distributions = new ArrayList<Distribution>();
    }
    public void addDistrib(Distribution d){
        this.distributions.add(d);
    }
    public void afficheToi(){ }
}

```

**Nom:****Prénom:****Matricule:**

---

```
public class Distribution {
    private String role;
    private Intervenant intervenant;
    private Production production;
    public Distribution(Production p, String r, Intervenant i){
        this.production = p;
        this.role = r;
        this.intervenant = i;
        p.addDistrib(this);
        i.addDistrib(this);
    }
    public String getRole(){
        return role;
    }
    public Intervenant getIntervenant(){
        return intervenant;
    }
}
```

```
import java.util.ArrayList;
public abstract class Intervenant {
    private String nom;
    public ArrayList<Distribution> distributions;
    public Intervenant(String n){
        this.nom = n;
        this.distributions = new ArrayList<Distribution>();
    }
    public String getNom(){
        return nom;
    }
    public void addDistrib(Distribution d){
        this.distributions.add(d);
    }
    public abstract void joue();
}
```

```
public class Musicien extends Intervenant {
    private String instrument;
    public Musicien(String n, String i){
        super(n);
        this.instrument = i;
    }
    public void accordeInstrum(int frequence){}
    public void joue(){}
}
```

**Nom:**

**Prénom:**

**Matricule:**

---

```
import java.util.ArrayList;
public class Chef extends Intervenant {
    private ArrayList<Ensemble> ensembles;
    public Chef(String n){
        super(n);
        this.ensembles = new ArrayList<Ensemble>();
    }
    public void addEnsemble(Ensemble e){
        this.ensembles.add(e);
    }
    public void joue(){}
}
```

```
import java.util.ArrayList;
public class Ensemble extends Intervenant {
    private ArrayList<Musicien> musiciens;
    private Chef chef;
    public Ensemble(String n, Chef c){
        super(n);
        this.chef = c;
        this.musiciens = new ArrayList<Musicien>();
    }
    public void addMusicien(Musicien m){
        this.musiciens.add(m);
    }
    public void setChef(Chef c){
        this.chef = c;
        c.addEnsemble(this);
    }
    public ArrayList<Musicien> getMusiciens(){
        return this.musiciens;
    }
    public Chef getChef(){
        return chef;
    }
    public void joue(){}
}
```

---

**Nom:****Prénom:****Matricule:**

---

**Question n°3: Compréhension Code Java (/3)**

Lisez attentivement le code Java ci-dessous (sans erreur). Que se passe-t-il à l'exécution (i.e. que va-t-il apparaître à l'écran) ?

```
import java.util.ArrayList;
public class Pathologie {
    private String nomSavant;
    private String symptome;
    private ArrayList<Therapie> therapies;
    private int gravite;
    public Pathologie(String n, String s, int g){
        nomSavant = n;
        symptome = s;
        gravite = g;
        therapies = new ArrayList<Therapie>();
    }
    public int getGravite(){
        return gravite;
    }
    public void addTherapie(Therapie nouvelle){
        therapies.add(nouvelle);
    }
    public ArrayList<Therapie> getTherapies(){
        return therapies;
    }
    public String ouCaFaitMal(){
        return symptome;
    }
    public String getNomSavant(){
        return nomSavant;
    }
}

public class Therapie {
    private String nom;
    private int force;
    public Therapie(String n, int f){
        nom = n;
        force = f;
    }
    public String getNom(){
        return nom;
    }
    public int getForce(){
        return force;
    }
}
```

**Nom:****Prénom:****Matricule:**

---

```
public class Patient {
    private Medecin doc;
    private String nom;
    private ArrayList<Pathologie> mesPetitsSoucis;
    private ArrayList<Therapie> mesTraitements;
    public Patient(String n, Medecin d){
        nom = n;
        doc = d;
        mesPetitsSoucis = new ArrayList<Pathologie>();
        mesTraitements = new ArrayList<Therapie>();
    }
    public void vaChezLeMedecin(){
        doc.soigne(this);
    }
    public void addUnPetitSouci(Pathologie p){
        mesPetitsSoucis.add(p);
    }
    public void addTraitement(Therapie t){
        mesTraitements.add(t);
    }
    public void ouAvezVousMal(){
        for(int i=0; i<mesPetitsSoucis.size(); i++){
            System.out.println(mesPetitsSoucis.get(i).ouCaFaitMal());
        }
    }
    public ArrayList<Pathologie> getPathologies(){
        return mesPetitsSoucis;
    }
    public void avezVousCompris(boolean docEstBon){
        if(docEstBon){
            String s = "Je souffre de ";
            for(int i=0;i<mesPetitsSoucis.size();i++){
                s = s+mesPetitsSoucis.get(i).getNomSavant()+" ";
            }
            s = s+"et vous me conseillez de prendre ";
            for(int i=0;i<mesTraitements.size();i++){
                s = s+mesTraitements.get(i).getNom()+" ";
            }
            System.out.println(s+"et ça ira beaucoup mieux.");
        } else {
            String s = "Merci docteur, je vais donc prendre ";
            for(int i=0;i<mesTraitements.size();i++){
                s = s+mesTraitements.get(i).getNom();
                s = s+" pour soigner mon problème de ";
                s = s+mesPetitsSoucis.get(i).getNomSavant()+" ";
            }
            System.out.println(s+"et tout ira bien.");
        }
    }
}
```

**Nom:****Prénom:****Matricule:**

```

public abstract class Medecin {
    private Therapie traitementParDefaut;
    public Medecin(Therapie t){
        traitementParDefaut = t;
    }
    public Therapie getTraitementParDefaut(){
        return traitementParDefaut;
    }
    public void soigne(Patient p){
        p.ouAvezVousMal();
    }
    public abstract Therapie choisitTherapie(Pathologie p);
}

public class BonMedecin extends Medecin {
    public BonMedecin() {
        super(new Therapie("du repos",1));
    }
    public void soigne(Patient p) {
        filtrePathologies(p);
        ArrayList<Pathologie> pt = p.getPathologies();
        for(int i=0;i<pt.size();i++){
            p.addTraitement(choisitTherapie(pt.get(i)));
        }
        p.addTraitement(new Therapie("je vous tiens au courant",0));
        p.avezVousCompris(true);
    }
    private void filtrePathologies(Patient p){
        for(int i=p.getPathologies().size()-1; i>=0; i--){
            if(p.getPathologies().get(i).getGravite(<5){
                p.getPathologies().remove(i);
            }
        }
    }
    public Therapie choisitTherapie(Pathologie p){
        Therapie therapieChoisie = getTraitementParDefaut();
        int forceMinimum = 10;
        for(int i=0;i<p.getTherapies().size();i++){
            if(p.getTherapies().get(i).getForce(<forceMinimum){
                forceMinimum = p.getTherapies().get(i).getForce();
                therapieChoisie = p.getTherapies().get(i);
            }
        }
        return therapieChoisie;
    }
}

public class MauvaisMedecin extends Medecin {
    public MauvaisMedecin() {
        super(new Therapie("des antibiotiques",8));
    }
    public void soigne(Patient p) {
        ArrayList<Pathologie> pt = p.getPathologies();

```

**Nom:**

**Prénom:**

**Matricule:**

```

        for(int i=0;i<pt.size();i++){
            p.addTraitement(choisitTherapie(pt.get(i)));
        }
        p.avezVousCompris(false);
    }
    public Therapie choisitTherapie(Pathologie p){
        Therapie therapieChoisie = getTraitementParDefaut();
        int forceMaximum = 0;
        for(int i=0;i<p.getTherapies().size();i++){
            if(p.getTherapies().get(i).getForce()>forceMaximum){
                forceMaximum = p.getTherapies().get(i).getForce();
                therapieChoisie = p.getTherapies().get(i);
            }
        }
        return therapieChoisie;
    }
}

public class CentreMedical {
    public static void main(String[] args) {
        Medecin docteurG = new BonMedecin();
        Medecin attila = new MauvaisMedecin();
        ArrayList<Pathologie> pt = new ArrayList<Pathologie>();
        pt.add(new Pathologie("lombalgie", "j'ai mal au dos quand je me réveille",3));
        pt.add(new Pathologie("gastro-entérite", "tout ce que je mange ressort aussitôt",7));
        pt.add(new Pathologie("rhume", "j'ai le nez qui coule, je tousse, et je suis fatigué",5));
        pt.get(0).addTherapie(new Therapie("du sport",3));
        pt.get(0).addTherapie(new Therapie("de la chirurgie",10));
        pt.get(1).addTherapie(new Therapie("du motilium",6));
        pt.get(1).addTherapie(new Therapie("un régime",2));
        Patient nicolas = new Patient("Van Zee",docteurG);
        Patient hugues = new Patient("Bersini",attila);
        nicolas.addUnPetitSouci(pt.get(0));
        nicolas.addUnPetitSouci(pt.get(1));
        nicolas.addUnPetitSouci(pt.get(2));
        hugues.addUnPetitSouci(pt.get(2));
        hugues.addUnPetitSouci(pt.get(0));
        nicolas.vaChezLeMedecin();
        hugues.vaChezLeMedecin();
    }
}

```

**REPONSE :**

Je souffre de gastro-entérite, rhume, et vous me conseillez de prendre un régime, du repos, je vous tiens au courant, et ça ira beaucoup mieux.

Merci docteur, je vais donc prendre des antibiotiques pour soigner mon problème de rhume, de la chirurgie pour soigner mon problème de lombalgie, et tout ira bien.

**Nom:****Prénom:****Matricule:****Question n°4: Correction Code Java (/2)**

Le code Java ci-dessous contient quelques erreurs de logique qui seraient détectées par le compilateur. Veuillez identifier, corriger et expliquer brièvement ces erreurs dans le code directement. La justification importe autant que l'identification et la correction des erreurs. Pointer des erreurs à tort et à travers pourrait être pénalisé !

```

public abstract class Presentation {
    private String titre, auteur;
    public Presentation(String t, String a){
        titre = t;
        auteur = a;
    }
    public String getTitre() {
        return titre;
    }
    public String getAuteur() {
        return auteur;
    }
    public abstract String afficheToi();
}

public class Normale extends Presentation { // Méthode abstraite afficheToi:String non définie
    private boolean accepte;
    public Normale(String t, String a) { // Manque l'appel au super constructeur : super(t,a)
        accepte = false;
    }
    public boolean isAccepte(){
        return accepte;
    }
    public void accepte(){
        accepte = true;
    }
}

public class Invitee extends Presentation {
    public boolean confirme;
    public Invitee(String t, String a) {
        super(t,a);
        confirme = false;
    }
    public void orateurConfirme(){
        confirme = true;
    }
    public String afficheToi() {
        if(!confirme){
            String conf = " [à confirmer]";
        }
        String s = auteur+": "+titre; // auteur et titre sont private > Utiliser les getters
        return s+conf; // conf n'existe pas, → déclarer 'String conf' avant le if
    }
}

```

**Nom:****Prénom:****Matricule:**

```
public class ComiteScientifique {
    private double selectivite;
    public ComiteScientifique(double nbPresentations, double nbSlots){
        selectivite = nbSlots/nbPresentations;
    }
    public void decide(Presentation p){
        if(Math.random()<selectivite){
            p.accepte(); // la méthode accepte() n'existe pas dans la classe Presentation
                       // il aurait fallu déclarer 'Normale p' dans la signature de la méthode
        }
    }
}

import java.util.ArrayList;
public class Session {
    private int capacite;
    private String date_heure, titre;
    private ArrayList<Presentation> presentations;
    public Session(String dh, String t, int c){
        date_heure = dh;
        titre = t;
        capacite = c;
        presentations = new ArrayList<Presentation>();
    }
    public boolean ajoutePresentation(Presentation p){
        if(presentations.size()>=capacite){
            return false;
        } else{
            presentations.add(p); // Il manque return true dans le else
        }
    }
    public int capaciteRestante(){
        return capacite-presentations.size();
    }
    public void afficheToi(){
        System.out.println("Session "+titre+" - "+date_heure);
        for(int i=0;i<presentations.size();i++){
            System.out.println("\t"+(i+1)+". "+presentations.get(i).afficheToi());
        }
    }
}
```

**Nom:****Prénom:****Matricule:**

```

public class Conference {
    private String titre, dates, lieu;
    private ArrayList<Session> sessions;
    private ArrayList<Normale> presentations;
    public Conference(String titre, String dates, String lieu, String org, String so_dh) {
        this.titre = titre;
        this.dates = dates;
        this.lieu = lieu;
        this.sessions = new ArrayList<Session>();
        Session ouverture = new Session(so_dh,1); // Il manque un argument dans le constructeur
        ouverture.ajoutePresentation(new Presentation(titre,org)); // Presentation est abstraite
        this.sessions.add(ouverture);
        this.presentations = new ArrayList<Normale>();
    }
    public void ajouteSession(Session s){
        sessions.add(s);
    }
    public void ajoutePresentation(Normale p){
        this.presentations.add(p);
    }
    public int capaciteRestante(){
        int c = 0;
        for(int i=0;i<sessions.size();i++){
            c += sessions.get(i).capaciteRestante();
        }
        return c;
    }
    private void afficheToi(){
        System.out.println(this.titre+" - "+lieu+", "+dates);
        for(int i=0;i<sessions.size();i++){
            sessions.get(i).afficheToi();
        }
    }
    public void traitePropositions(){ // ci-dessous : comite n'a pas été déclarée
        comite = new ComiteScientifique(presentations.size(), this.capaciteRestante());
        for(int i=0;i<presentations.size();i++){
            comite.decide(presentations.get(i));
        }
    }
    public void allouePropositions(){
        int session = 0;
        for(int i=0;i<presentations.size();i++){
            if(presentations.get(i).isAccepte() && session<sessions.size()){
                if(!sessions.get(session).ajoutePresentation()){ // Manque 'p' dans les ()
                    session++;
                    i--;
                }
            }
        }
    }
}

```

**Nom:****Prénom:****Matricule:**

---

```
public class Main {
    public static void main(String args[]){
        String c_t = "L'informatique à Solvay";
        String c_d = "20-23/12/2010";
        String c_l = "Honolulu";
        String c_o = "Bersini & VanZee";
        String c_ouv = "23/12 - 17h30-18h";
        Conference c = new Conference(c_t,c_d,c_l,c_o,c_ouv);
        Session s1 = new Session("20/12 - 18h-20h","Microsoft or not?",1);
        Session s2 = new Session("21/12 - 14h-18h", "L'avenir de Java",1);
        s1.ajoutePresentation(new Invitee("Pourquoi enseigner Microsoft?","Gates"));
        s1.ajoutePresentation(new Invitee("Parlons plutôt d'Apple","Jobs"));
        s2.ajoutePresentation(new Invitee("Pourquoi j'ai racheté Java","Ellison"));
        c.ajouteSession(s1);
        c.ajouteSession(s2);
        c.ajouteSession(new Session("22/12 - 10h-18h","Semaine info ou TP",2));
        c.ajouteSession(new Session("23/12 - 10h-18h","Rendre la semaine info encore + fun",2));
        c.ajoutePresentation(new Normale("L'apprentissage à la dure","Stakhanov"));
        c.ajoutePresentation(new Normale("La gestion du stress","Hitchcock"));
        c.ajoutePresentation(new Normale("En forgeant, on devient forgeron","Bersini"));
        c.ajoutePresentation(new Normale("Pourquoi on aime la semaine Java","Quentin et Gaël"));
        c.ajoutePresentation(new Normale("Du Java à la Java","Gaël et Benjamin"));
        c.ajoutePresentation(new Normale("Matière à rire","Devos"));
        c.ajoutePresentation(new Normale("Les jeux, c'était plus chouette","Van Zee"));
        c.traiterPropositions();
        allouerPropositions(); // Sur quoi s'applique cette méthode ? → c.allouerPropositions()
        c.afficheToi(); // afficheToi() est private dans la classe Conference
    }
}
```