

Présentation du contexte

L'organisation cliente : le centre de formation d'apprentis SupMaster

SupMaster est un centre de formation d'apprentis (CFA) préparant des jeunes âgés de 18 à 25 ans au diplôme d'ingénieur. Établissement de formation professionnelle par alternance placé sous la tutelle du ministère de l'Education nationale, SupMaster est investi d'une mission de service public dont la création, le fonctionnement et les attributions sont très précisément décrits dans le livre I du code du travail. C'est au titre de cette mission de service public que le CFA perçoit de l'argent public qui couvre, en partie, ses frais de fonctionnement pédagogique.

SupMaster forme des chefs de projets (bac+5) alliant compétences techniques et managériales. Le centre couvre deux domaines : les systèmes d'information (cursus IM pour Informaticien Manager) et le multimédia (cursus MM pour Media Manager). L'aspect transversal des formations joue un rôle d'accélérateur d'employabilité pour des apprentis recrutés à bac+2. Les formations du CFA sont reconnues au niveau 1 par le RNCP (répertoire national des certifications professionnelles). SupMaster forme ses apprentis dans le cadre de contrats d'alternance pluriannuels entre l'étudiant, le CFA et une entreprise. Le principe des études consiste à effectuer une période d'un mois dans un établissement de formation SupMaster et une période d'un mois dans l'organisation.

Depuis sa création à Marseille en 2002, le centre de formation a choisi une stratégie de diversification géographique en s'implantant dans de grandes villes de France (Lyon, Lille, Paris, Strasbourg, Rennes et Bordeaux). Il forme aujourd'hui environ 300 étudiants, possède 25 salariés à temps plein, auxquels il faut ajouter 70 intervenants extérieurs qui réalisent différents enseignements dans les cursus proposés.

L'entreprise prestataire de services

SIForm est une entreprise de services du numérique (ESN) marseillaise implantée à proximité du centre de formation. Elle réalise des développements auprès de clients qu'elle démarché, ou pour ses besoins propres.

Elle compte un effectif de 6 personnes à temps plein qui interviennent dans tous les domaines de l'informatique d'aujourd'hui, et plus particulièrement les développements Web et mobiles.

Problématique du projet

Le choix exclusif de l'alternance pour tous les étudiants impose un travail important sur la reconnaissance de SupMaster par les organisations (entreprises, administrations, ou encore associations.) susceptibles de fournir des contrats aux étudiants. Pour maintenir sa place dans le tissu concurrentiel des écoles du domaine des technologies de l'information et de la communication, le directeur général, Léopold C. a décidé de développer une application destinée à améliorer la notoriété du centre. Compte tenu de la stratégie de diversification géographique, les organisations situées dans les régions couvertes par chaque établissement du centre constituent la cible principale de l'action.

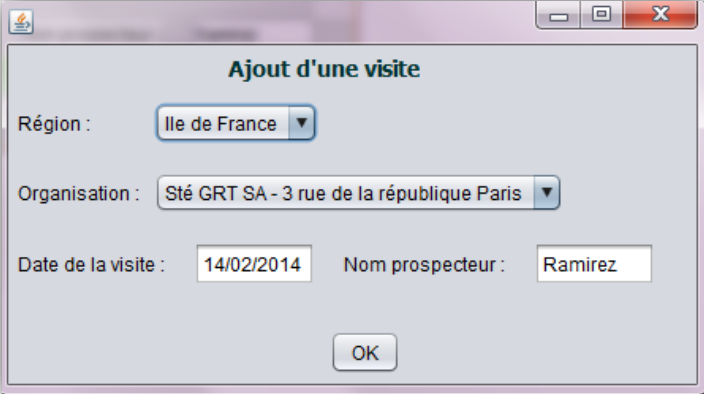
Ce projet est confié à SIForm. L'ESN aura pour rôle d'adapter l'existant aux exigences nouvelles de la direction générale. La proximité géographique des deux structures facilite la communication et le contrôle des travaux. Fraichement embauché-e par SIForm, vous intervenez dans plusieurs missions mises en place par l'ESN dans le cadre de ce projet.

Mission 3 : Suivi des visites de prospection

Document(s) à utiliser : 4, 5, 6, 7, 8

IMPORTANT : la candidate ou le candidat peut choisir de présenter les éléments de code à l'aide du langage de programmation de son choix ou de pseudo-code algorithmique.

Pour prospecter les organisations, SupMaster emploie une personne par établissement. Pour faciliter le travail de ce prospecteur, une application java *GestVisite* est en cours d'écriture. Elle permettra notamment la saisie d'une visite par le prospecteur d'un centre SupMaster. L'exemple ci-contre, consiste à enregistrer la première visite réalisée auprès de l'organisation *GRT SA* située dans la région *Ile de France*, par Monsieur Ramirez, prospecteur de l'établissement *SupMaster-Paris*.



Trois points sont à traiter :

- À la 1^{ère} exécution du programme, le message d'erreur suivant est apparu :

`java.lang.NullPointerException`
at CodeSource.Organisation.ajoutVisite(Organisation.java:1)
at CodeSource.GestVisite (GestVisite.java:10)

- Le responsable du projet a constaté une erreur de conception dans le programme *GestVisite* et l'a formalisée dans la *demande de modification PO_14*. Un de vos collègues a commencé à réfléchir à ce problème. Il a complété la *demande de modification* en précisant les étapes de l'implémentation technique.
- Pour améliorer le suivi des visites, le responsable des prospections souhaite que chaque prospecteur dispose de statistiques concernant son établissement. Il serait entre autres intéressant de connaître, pour chaque région couverte par l'établissement, l'indicateur de couverture des visites qui est calculé ainsi :

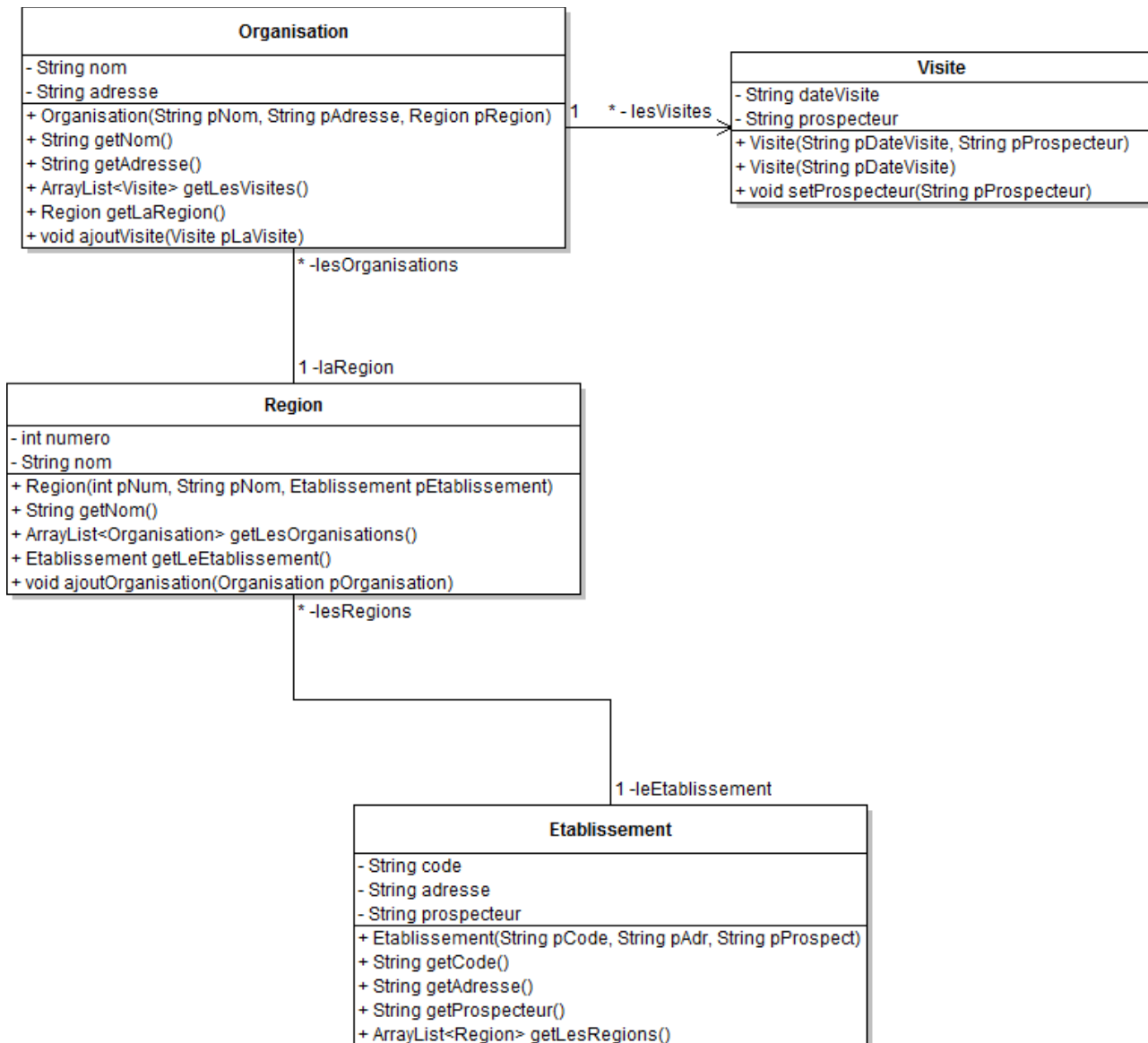
nombre d'organisations de la région déjà visitées au moins une fois divisé par nombre total d'organisations de la région

Votre mission

1. Expliquer la cause du message d'erreur apparu à la première exécution du programme *GestVisite* et donner la solution permettant d'y remédier.
2. Coder les différentes étapes permettant de réaliser la modification demandée dans la fiche *PO_14*.
3. Coder une solution permettant de calculer l'indicateur de couverture des visites d'une région.

4 Extrait du diagramme des classes métier

La description des classes est fournie ici à un niveau d'abstraction intermédiaire, suffisant pour la compréhension des informations principales à gérer.



5 Description détaillée des classes métier (extrait)

```
public class Region {
    private int numero;
    private String nom;
    private Etablissement leEtablissement; // Etablissement de Supmaster couvrant la région
    private ArrayList <Organisation> lesOrganisations;
    // Collection des organisations prospectées ou susceptibles de l'être dans la région.

    public Region(int pNum, String pNom, Etablissement pEtablissement) {...}
    // Constructeur qui initialise les 3 premiers attributs avec les paramètres.
    public String getNom() {...} // retourne l'attribut nom
    public ArrayList<Organisation> getLesOrganisations() {...} // retourne l'attribut lesOrganisations
    public Etablissement getLeEtablissement() {...} // retourne l'attribut leEtablissement
    public void ajoutOrganisation(Organisation pOrganisation) {...}
}

public class Etablissement {
    private String code;
    private String adresse;
    private String prospecteur;
    private ArrayList<Region> lesRegions; // zone géographique couverte par l'établissement
    public Etablissement(String pCode, String pAdr, String pProspect) {...}
    // Constructeur qui initialise les 3 premiers attributs avec les paramètres.
    public String getCode() {...} // retourne l'attribut code
    public String getAdresse() {...} // retourne l'attribut adresse
    public String getProspecteur() {...} // retourne l'attribut prospecteur
    public ArrayList<Region> getLesRegions() {...} // retourne l'attribut lesRegions
}

public class Organisation {
    private String nom;
    private String adresse;
    private ArrayList<Visite> lesVisites;
    private Region laRegion;

    public Organisation(String pNom, String pAdresse, Region pRegion) {
        this.nom = pNom ;
        this.adresse = pAdresse ;
        this.laRegion = pRegion ;
    }

    public String getNom() {...} // retourne l'attribut nom
    public String getAdresse() {...} // retourne l'attribut adresse
    public ArrayList<Visite> getLesVisites() {...} // retourne l'attribut lesVisites
    public Region getLaRegion() {...} // retourne l'attribut laRegion
    public void ajoutVisite(Visite pLaVisite) {
        lesVisites.add(pLaVisite) ;
    }
}

public class Visite {
    private String dateVisite;
    private String prospecteur;
    public Visite(String pDateVisite, String pProspecteur) {...}
    // Constructeur qui initialise les deux attributs avec les deux paramètres
    public Visite(String pDateVisite) {...}
    //constructeur qui initialise uniquement la date
    public void setProspecteur(String pProspecteur) {...} // renseigne l'attribut prospecteur
}
```

6 Utilisation d'une collection

L'exemple ci-dessous permet de manipuler une collection de chaînes de caractères. Le principe est le même quel que soit le type des éléments.

```
ArrayList<String> mesChaines;           // déclaration d'une collection de chaînes de caractères
mesChaines = new ArrayList<String>();   // instantiation de la collection
mesChaines.add("un");                   // ajout d'une chaîne à la collection
mesChaines.add("deux");
mesChaines.add("trois");
System.out.println(mesChaines.size()); // affichage du nombre d'éléments de la collection
for (String uneChaine : mesChaines) {  // parcours de la collection
    System.out.println(uneChaine);      // affichage de l'élément courant
}
mesChaines.remove(1);                   // suppression du 2ème élément (indice 1)
System.out.println(mesChaines.get(0));  // affichage du 1er élément (indice 0)
```

7 Extrait du programme *GestVisite*

```
1.  public class GestVisite extends JFrame {
2.      [...]
3.          // clic sur le bouton OK
4.          private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
5.              [...]
6.              // À ce stade l'objet uneOrga contient l'organisation sélectionnée par
7.              //l'utilisateur et est correctement instancié
8.              Visite uneVisite = new Visite(      txtSaisieDateVisite.getText(),
9.                                                  txtSaisieNomProspecteur.getText());
10.             uneOrga.ajoutVisite(uneVisite);
11.             [...]
12.         }
13.     }
```

8 Demande de modification PO_14

Projet : Prospection des Organisations (PO)		Émise le : 26/02/2014	
Demande de modification n° : PO_14			
Description de la modification demandée			
Nom :Gandri Jacques		Fonction : Responsable du projet	
Niveau d'importance :		Faible <input type="checkbox"/>	
		Majeur <input checked="" type="checkbox"/>	
		Bloquant <input type="checkbox"/>	
<i>Fonctionnalité concernée</i> : saisie des visites réalisées par les prospecteurs.			
<i>Description</i> : La saisie du nom du prospecteur dans la fenêtre du programme <i>GestVisite</i> pourrait induire des incohérences avec l'attribut <i>prospecteur</i> de la classe <i>Etablissement</i> . La solution consiste à ne demander que la saisie de la date dans l'interface de saisie.			
Implémentation technique :			
1. Modification du formulaire de saisie d'une visite : saisie de la date seule (la visite sera instanciée avec le constructeur paramétré qui instancie la visite uniquement avec la date)			
2. Réécriture de la méthode <i>AjoutVisite()</i> de la classe <i>Organisation</i> avec la récupération du nom du prospecteur grâce à l'attribut <i>prospecteur</i> de la classe <i>Etablissement</i> , mise à jour de l'attribut <i>prospecteur</i> de l'objet <i>Visite</i> puis ajout de la visite à la collection.			