

Module Programmation Objet

Chapitre 10 - Éléments de Conception à objets

Pascal André, Gilles Ardourel

Université de Nantes

2010

Projet DVD Miage



Introduction

Un exemple illustré

Règles de codage : Exemples en Java et C++

Résumé

Introduction

Synthèse appliquée du module

Dans ce chapitre, nous ne proposons pas un cours de modélisation, ni même de conception, nous faisons simplement une petite synthèse d'éléments de conception détaillée, illustrée à travers quelques exemples.

L'approche n'est pas

- ▶ exhaustive : l'objectif est de modéliser de manière intéressante et d'implanter les modèles,
- ▶ théorique : le discours est vu à travers les exemples.

Un autre module, intitulé **Outils de modélisation, UML** est consacré à la modélisation.

Objectifs du chapitre

Appréhender la conception à objets :

- ▶ Concevoir des applications réutilisables et extensibles à travers le modèle à objets : (concepts, combinaisons, alternatives).
- ▶ Etudier les problèmes de faisabilité du codage avec le modèle à classes : (type, héritage, instanciation).
- ▶ Pointer des points-clé du passage du modèle conceptuel UML à celui d'un programme objet.
- ▶ Pratiquer la modélisation et le prototypage (programmation Java et Smalltalk).

Le chapitre est structuré en un document illustrant le problème sur un exemple et une série d'exercices. Certains exercices sont corrigés.

Outils supports du chapitre

Pour illustrer le discours, nous utiliserons trois standards, largement adoptés par les communautés de développeurs autour de l'objet et du Web.

- ▶ une notation graphique : UML
Unified Modeling Language est le langage adopté dans les années 1990 pour représenter des systèmes à objets. Parmi les nombreux diagrammes à disposition, nous utiliserons les diagrammes
 - ▶ d'objets, séquence, collaboration
 - ▶ de classes

Sur ce même support DVD, il existe un cours de modélisation à UML auquel nous renvoyons le lecteur pour plus de détails.

Outils supports du chapitre

Pour illustrer le discours, nous utiliserons trois standards, largement adoptés par les communautés de développeurs autour de l'objet et du Web.

- ▶ une notation graphique : UML
- ▶ un langage à objets : Java

Ce langage hérité et fait une bonne synthèse de ses prédécesseurs C++, Smalltalk et Eiffel. Nous serons amenés à illustrer parfois notre discours par ces autres langages pour tel ou tel concept.

Outils supports du chapitre

Pour illustrer le discours, nous utiliserons trois standards, largement adoptés par les communautés de développeurs autour de l'objet et du Web.

- ▶ une notation graphique : UML
- ▶ un langage à objets : Java
- ▶ un environnement de développement : Eclipse
Cet IDE riche s'est imposé par sa facilité d'extension et de personnalisation.

Introduction

Un exemple illustré

Règles de codage : Exemples en Java et C++

Résumé

Un exemple illustré

Nim

Le jeu de Nim se joue entre deux joueurs et avec un tas d'allumettes. Les joueurs enlèvent alternativement 1, 2 ou 3 allumettes. Le perdant est celui qui épuise le tas.

Démarche

1. Conception à objets avec UML
 - 1.1 Conception abstraite à objets
 - 1.2 Conception détaillée
 2. Programmation, codage
 - 2.1 Java
 - 2.2 Smalltalk
- ▶ Exemple du jeu de Nim : [fiche1_nim.pdf](#)
 - ▶ Tennis : [fiche10_tennis.pdf](#)

Introduction

Un exemple illustré

Règles de codage : Exemples en Java et C++

Résumé

Commentaires

```
//surcharge de l'opérateur +
Nlong& Nlong::operator+ (Nlong & e) {
    if (x+e.x>max) {
        throw ExceptionBorne("valeur de résultat non représentable");
    }
    else{
        x+=e.x; //on modifie la valeur de l'objet courant
        return *this; //on retourne une référence sur un Nlong ici l'objet courant
    }
}

//surcharge de l'opérateur -
Slong &Nlong::operator- (Nlong & e) {
    Slong nb; //on créer un objet Slong pour mettre le résultat
    nb.x=this->x-e.x; //on met le résultat de la soustraction dans nb.x
    return *nb; //on retourne une référence sur un Slong
}
```

Méthodes longues

```
public FicheEntreprise fusion( FicheEntreprise ficheEntr ) {  
    // renvoie une fiche correspondant a la fusion de la fiche courante et  
    // de la fiche en parametre  
    // ex: s'il y a 2 adresses dans la fiche1 et 1 autre dans la fiche2 la  
    // fiche fusionnee en aura 3  
    // creation de la fiche de retour et copie des elements de la fiche  
    // courante  
    FicheEntreprise ficheFusion = new FicheEntreprise(  
        ( Entreprise ) this.personne);  
    for ( int i = 0; i < this.adresse.size (); i++)  
        ficheFusion .ajouterAdresse( (AdEntreprise) this.adresse.get(i));  
    for ( int i = 0; i < this.email.size (); i++)  
        ficheFusion .ajouterEmail( (Email) this.email.get(i));  
    for ( int i = 0; i < this.numtel.size (); i++)  
        ficheFusion .ajouterNumtel( (Numtel) this.numtel.get(i));  
}
```

*// pour les adresses, les emails, les numtels, on teste si la fiche
// courante les possede deja, sinon on les ajoute*

```
String testLabel = "", testMail = "", testTel = "";  
LinkedList listeAdresse = ficheEntr.getAdresse();  
LinkedList listeEmail = ficheEntr.getEmail();  
LinkedList listeNumtel = ficheEntr.getNumtel();  
boolean estPresent = false;
```

```
// adresses
for (int i = 0; i < listeAdresse . size (); i++) {
    testLabel = ((AdEntreprise) listeAdresse . get(i)).getLabel();
    for (int j = 0; j < this.adresse . size (); j++) {
        if (((AdEntreprise) adresse . get(j)).getLabel())
            .indexOf(testLabel) == 0) {
            estPresent = true;
        }
    }
    if (!estPresent)
        ficheFusion . ajouterAdresse((AdEntreprise) listeAdresse . get(i));
    estPresent = false;
}
```

```
// emails
for (int i = 0; i < listeEmail . size (); i++) {
    testMail = ((Email) listeEmail . get(i)).getMail();

    for (int j = 0; j < this.email . size (); j++) {
        if (((Email) this . email . get(j)).getMail (). indexOf(testMail) == 0) {
            estPresent = true;
        }
    }
    if (!estPresent)
        ficheFusion . ajouterEmail ((Email) listeEmail . get(i));

    estPresent = false;
}
```

```
// numtels
for (int i = 0; i < listeNumtel.size (); i++) {
    testTel = ((Numtel) listeNumtel.get(i)).getNumtel();

    for (int j = 0; j < this.numtel.size (); j++) {
        if (((Numtel) this.numtel.get(j)).getNumtel())
            .indexOf(testTel) == 0) {
            estPresent = true;
        }
    }
    if (!estPresent)
        ficheFusion .ajouterNumtel(((Numtel) listeNumtel.get(i)));

    estPresent = false;
}
return ficheFusion ;
}
```

Paramètres

```
public void ajouterAdresse(String label, String adresse, String codePostal,
    String ville ) {
    boolean labelExist = true;
    for (int i = 0; i < this.adresse.size (); i++) {
        if (((Adresse) this.adresse.get(i)).getLabel ().indexOf(label) == 0)
            labelExist = false;
    }
    if ( labelExist )
        this.adresse.add(new Adresse(label, adresse, codePostal, ville ));
    else
        System.out
            .println ("Vous ne pouvez créer une adresse avec un label
                déjà existant\n");
}
```

Désir des propriétés d'autrui

```
public boolean egal(String nom, String prenom) {  
    return ((( Particulier ) this.personne).getNom()).indexOf(nom) == 0)  
        && ((( Particulier ) this.personne).getPrenom()).  
            indexOf(prenom) == 0);  
}
```

Paramètres attachés (1/3)

```
public void ajouterFiche(String nom, String prenom) {  
    // ajout d'un particulier a partir de son nom et de son prenom  
    // si celui ci n'a pas deja une fiche  
    if (this.searchFiche(nom, prenom) == null) {  
        FicheParticulier fiche = new FicheParticulier(nom, prenom);  
        this.repertoire.add(fiche);  
    } else  
        System.out.println("Ce particulier a deja une fiche");  
}  
  


---

  
public void supprimerFiche(String nom, String prenom) {  
    // supprime une fiche a partir d'un nom et prenom  
    for (int i = 0; i < this.repertoire.size(); i++) {  
        if (((FicheParticulier) this.repertoire.get(i)).egal(nom, prenom))  
            this.repertoire.remove(i);  
    }  
}
```

Paramètres attachés (2/3)

```
public FicheParticulier searchFiche(String nom, String prenom) {  
    // cherche un particulier (nom et prenom) et renvoie sa fiche  
    FicheParticulier fiche = null;  
    int i = 0;  
    while ((i < this.repertoire.size()) && (fiche == null)) {  
        if (((FicheParticulier) this.repertoire.get(i)).egal(nom, prenom))  
            fiche = (FicheParticulier) this.repertoire.get(i);  
        i++;  
    }  
    return fiche;  
}
```

Paramètres attachés (3/3)

```
public String chercheFiche(String nom, String prenom) {  
    // renvoie l'affichage du resultat d'une recherche de particulier  
    FicheParticulier fiche = searchFiche(nom, prenom);  
    if (fiche == null)  
        return "Cette personne n'est pas repertoriee.\n";  
    else  
        return fiche.toString();  
}
```

Test de type

```
while(e.hasMoreElements()){  
    courant = (CaseMemoire)e.nextElement();  
    if ((courant instanceof CaseMemoireVide)&&  
        (precedent instanceof CaseMemoireVide)) {  
  
        indice = _memoire.indexOf(precedent);  
        adDeb = precedent.getAdresseDeb();  
        taille = (precedent. getTaille ())+ (courant. getTaille ());  
  
        vide = new CaseMemoireVide(adDeb,taille);  
        _memoire.removeElementAt(indice);  
        _memoire.removeElementAt(indice);  
        _memoire.insertElementAt(vide , indice );  
        e = _memoire.elements();  
        precedent = null;  
    } else { precedent = courant; }  
}
```

Programmation copier et coller (1)

FicheParticulier

```
public FicheParticulier fusion( FicheParticulier fichePart ) {  
    // renvoie une fiche correspondant a la fusion de la fiche courante et  
    // de la fiche en parametre  
    // ex: s'il y a 2 adresses dans la fiche1 et 1 autre dans la fiche2 la  
    // fiche fusionnee en aura 3  
    // creation de la fiche de retour et recopie des elements de la fiche  
    // courante  
    FicheParticulier ficheFusion = new FicheParticulier(  
        ( Particulier ) this.personne);  
    for ( int i = 0; i < this.adresse.size (); i++)  
        ficheFusion .ajouterAdresse ((Adresse) this.adresse.get(i));  
    for ( int i = 0; i < this.email.size (); i++)  
        ficheFusion .ajouterEmail ((Email) this.email.get(i));  
    for ( int i = 0; i < this.numtel.size (); i++)  
        ficheFusion .ajouterNumtel ((Numtel) this.numtel.get(i));  
}
```

Programmation copier et coller (2)

FicheEntreprise

```
public FicheEntreprise fusion( FicheEntreprise ficheEntr ) {  
    // renvoie une fiche correspondant a la fusion de la fiche courante et  
    // de la fiche en parametre  
    // ex: s'il y a 2 adresses dans la fiche1 et 1 autre dans la fiche2 la  
    // fiche fusionnee en aura 3  
    // creation de la fiche de retour et recopie des elements de la fiche  
    // courante  
    FicheEntreprise ficheFusion = new FicheEntreprise(  
        ( Entreprise ) this.personne);  
    for ( int i = 0; i < this.adresse.size (); i++)  
        ficheFusion .ajouterAdresse ((AdEntreprise) this.adresse.get(i));  
    for ( int i = 0; i < this.email.size (); i++)  
        ficheFusion .ajouterEmail ((Email) this.email.get(i));  
    for ( int i = 0; i < this.numtel.size (); i++)  
        ficheFusion .ajouterNumtel ((Numtel) this.numtel.get(i));  
}
```

Programmation copier et coller (3)

FicheParticulier

```
// pour les adresses, les emails, les numtels, on teste si la fiche  
// courante les possede deja, sinon on les ajoute  
String testLabel = "", testMail = "", testTel = "";  
LinkedList listeAdresse = fichePart.getAdresse();  
LinkedList listeEmail = fichePart.getEmail();  
LinkedList listeNumtel = fichePart.getNumtel();  
boolean estPresent = false;
```

Programmation copier et coller (4)

FicheEntreprise

```
// pour les adresses, les emails, les numtels, on teste si la fiche  
// courante les possede deja, sinon on les ajoute  
String testLabel = "", testMail = "", testTel = "";  
LinkedList listeAdresse = ficheEntr.getAdresse();  
LinkedList listeEmail = ficheEntr.getEmail();  
LinkedList listeNumtel = ficheEntr.getNumtel();  
boolean estPresent = false;
```

Programmation copier et coller (5)

FicheParticulier

```
// adresses
for (int i = 0; i < listeAdresse . size (); i++) {
    testLabel = ((Adresse) listeAdresse . get(i)).getLabel();

    for (int j = 0; j < this.adresse . size (); j++) {
        if (((((Adresse) adresse . get(j)).getLabel())
            .indexOf(testLabel) == 0) {
            estPresent = true;
        }
    }
    if (!estPresent)
        ficheFusion . ajouterAdresse((Adresse) listeAdresse . get(i));

    estPresent = false;
}
```

Programmation copier et coller (6)

FicheEntreprise

```
// adresses
for (int i = 0; i < listeAdresse . size (); i++) {
    testLabel = ((AdEntreprise) listeAdresse . get(i)).getLabel();

    for (int j = 0; j < this.adresse . size (); j++) {
        if (((AdEntreprise) adresse . get(j)).getLabel()
            .indexOf(testLabel) == 0) {
            estPresent = true;
        }
    }
    if (!estPresent)
        ficheFusion . ajouterAdresse((AdEntreprise) listeAdresse . get(i));

    estPresent = false;
}
```

Programmation copier et coller (7)

FicheParticulier

```
// emails
for (int i = 0; i < listeEmail . size (); i++) {
    testMail = ((Email) listeEmail . get(i)).getMail ();

    for (int j = 0; j < this.email . size (); j++) {
        if (((Email) this . email . get(j)).getMail ().indexOf(testMail) == 0) {
            estPresent = true;
        }
    }
    if (!estPresent)
        ficheFusion . ajouterEmail ((Email) listeEmail . get(i));

    estPresent = false;
}
```

Programmation copier et coller (8)

FicheEntreprise

```
// emails
for (int i = 0; i < listeEmail . size (); i++) {
    testMail = ((Email) listeEmail . get(i)).getMail ();

    for (int j = 0; j < this.email . size (); j++) {
        if (((Email) this . email . get(j)).getMail ().indexOf(testMail) == 0) {
            estPresent = true;
        }
    }
    if (!estPresent)
        ficheFusion . ajouterEmail ((Email) listeEmail . get(i));

    estPresent = false;
}
```

Programmation copier et coller (9)

FicheParticulier

```
// numtels
for (int i = 0; i < listeNumtel.size (); i++) {
    testTel = ((Numtel) listeNumtel.get(i)).getNumtel();
    for (int j = 0; j < this.numtel.size (); j++) {
        if (((Numtel) this.numtel.get(j)).getNumtel())
            .indexOf( testTel ) == 0) {
            estPresent = true;
        }
    }
    if (!estPresent)
        ficheFusion .ajouterNumtel((Numtel) listeNumtel.get(i));
    estPresent = false;
}
return ficheFusion ;
}
```

Programmation copier et coller (10)

FicheEntreprise

```
// numtels
for (int i = 0; i < listeNumtel.size (); i++) {
    testTel = ((Numtel) listeNumtel.get(i)).getNumtel();
    for (int j = 0; j < this.numtel.size (); j++) {
        if (((Numtel) this.numtel.get(j)).getNumtel())
            .indexOf( testTel ) == 0) {
            estPresent = true;
        }
    }
    if (! estPresent)
        ficheFusion .ajouterNumtel((Numtel) listeNumtel.get(i));
    estPresent = false;
}
return ficheFusion ;
}
```

Introduction

Un exemple illustré

Règles de codage : Exemples en Java et C++

Résumé

Synthèse du chapitre

Ce chapitre met en pratique des éléments vu au cours des chapitres sur des exemples complets. Nous illustrons certains mécanisme du codage d'une conception à objets dans les concepts des langages de la programmation à objets.

Vous êtes maintenant prêts à passer à plus grande échelle :

- ▶ programmation objet avancée et architectures techniques
- ▶ persistance avec les bases de données
- ▶ programmation Web
- ▶ programmation générative
- ▶ transformation de modèles
- ▶ ...