Ressourcesinformatiques

Développement informatique

Apprenez à concevoir avant de programmer

Michel GINESTE

Fichiers complémentaires à télécharger

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence ENI de l'ouvrage **RIDEVINF** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

1. Présentation 29
2. Techniques informatiques 30
   2.1 Méthodes, conception 30
   2.2 Langages 30
3. Environnements techniques 30
4. Les fichiers de données 31
5. Remerciements 31

Partie 1 : Langage C - Algorithmique

Chapitre 1-1
Initiation

1. Objectifs du chapitre 33
2. Premier programme 33
   2.1 Programme complet 33
   2.2 Explications détaillées du programme 34
3. Les types de base 38
   3.1 Types entiers 38
   3.2 Types flottants 39
   3.3 Taille des types de base 39
   3.4 Utilisation du type char 40
4. Exemples de formats numériques pour la fonction printf() 41
5. Principaux codes de format de la fonction scanf() 42
6. Quelques opérateurs 43
   6.1 Opérateurs arithmétiques 43
   6.2 Opérateurs de comparaison 43

6.3 Opérateurs d'affectation 44
6.4 Opérateurs logiques 44
6.5 Opérateurs unaires 45
6.6 Opérateurs de conversion (cast) 46
7. Structures logiques 47
7.1 La structure alternative 48
7.2 Les structures répétitives 54
8. Travail pratique : structures logiques 59
8.1 Sujet 59
8.2 Structures logiques : proposition de correction 60

Chapitre 1-2
Les tableaux

1. Objectifs du chapitre 67
2. Présentation des tableaux 67
3. Exemples d'utilisations de tableaux 69
3.1 Tableau à une dimension 69
3.2 Tableau à deux dimensions 70
4. Adresse d'un tableau et de ses postes (opérateurs * et &) 72
4.1 Tableau à une dimension 72
4.2 Tableau à deux dimensions 74
5. Travail pratique : tableaux 75
5.1 Sujet 75
5.2 Tableaux : proposition de correction 75
6. Travail pratique : tri de tableaux 78
6.1 Sujet 78
6.2 Tri de tableaux : proposition de correction 79
7. Travail pratique : triangle de Pascal 83
7.1 Objectif 83
7.2 Sujet 83
7.3 Triangle de Pascal : proposition de correction 84

Chapitre 1-3
Les pointeurs

1. Objectifs du chapitre 87
2. Définition 87
3. Exemples 87
   3.1 Pointeur sur un entier 87
   3.2 Pointeur sur un tableau 89
4. Pointeurs : une erreur classique 90
5. Allocation dynamique d'un tableau 91
   5.1 Tableau à une dimension 91
   5.2 Tableau à deux dimensions 92
6. Travail pratique : triangle de Pascal 94
   6.1 Objectif 94
   6.2 Sujet 94
   6.3 Triangle de Pascal : proposition de correction 94

Chapitre 1-4
Les chaînes de caractères

1. Objectifs du chapitre 97
2. Présentation 97
   2.1 Définition 97
   2.2 Affichage des lettres d'une chaîne et de leurs adresses 98
3. Affichage d'une chaîne de caractères 99
4. Stockage d'une chaîne de caractères 100
   4.1 Tableau de caractères 100
   4.2 Erreurs courantes 101
5. Bibliothèque standard de manipulation des chaînes : string.h 102
   5.1 Liste des fonctions 102
   5.2 Utilisation de quelques fonctions 104
6. Utilisation des chaînes pour les saisies numériques 105
   6.1 Problème du scanf() 105
   6.2 Saisies numériques correctes 106

7. Travail pratique : chaînes de caractères 107
7.1 Sujet 107
7.2 Chaînes de caractères : proposition de correction 108

Chapitre 1-5
Les fonctions

1. Présentation 117
2. Premières fonctions 118
2.1 Syntaxe 118
2.2 Calcul du carré et du cube d'un entier 118
3. Le passage de paramètres à une fonction 121
3.1 Échange de deux valeurs entières (première version) 121
3.2 Échange de deux valeurs entières (deuxième version) 124
4. Tableaux et fonctions : moyenne des valeurs d'un tableau 126
5. Tableaux et chaînes : codes ASCII des caractères d'une chaîne 127
6. Paramètres facultatifs de la fonction main : utilisation de argc, argv 128
7. Travail pratique : fonctions et fractions 129
7.1 Sujet 129
7.2 Fonctions et fractions : proposition de correction 130
8. Travail pratique : fonctions et chaînes 132
8.1 Objectif 132
8.2 Sujet 133
8.3 Fonctions et chaînes : proposition de correction 133

Chapitre 1-6
Les structures de données : struct, typedef struct

1. Objectifs du chapitre 141
2. Présentation 141
3. Structure de données 141
3.1 Mot-clé struct 141
3.2 typedef struct 143
4. Pointeurs sur des structures, opérateur -> 143

5. Structures contenant des structures 145
6. Travail pratique : structures de données 146
6.1 Objectif 146
6.2 Sujet 146
6.3 Structures de données : proposition de correction 146

Chapitre 1-7
Bibliothèques de fonctions

1. Objectifs du chapitre 149
2. Présentation 149
3. Création d'une bibliothèque de fonctions 150
3.1 Découpage de l'application "Structure de données" 150
3.2 Création d'un nouveau projet 151
3.3 Programmes du projet biblFractions 152
4. Utilisation d'une bibliothèque de fonctions 155
4.1 Création de dossiers pour la bibliothèque de gestion des fractions . . 155
4.2 Utilisation de la bibliothèque de gestion des fractions. 156
5. Travail pratique : bibliothèque de fonctions - chaînes 158
5.1 Objectif 158
5.2 Sujet 158
5.3 Bibliothèque de fonctions - chaînes : proposition de correction 158
5.4 Programmes utilitaires : la bibliothèque chaines.h 160

Chapitre 1-8
Les fichiers

1. Objectifs du chapitre 161
2. Les fichiers en C 161
2.1 Présentation 161
2.2 Lecture et écriture 162
2.3 Options d'ouverture d'un fichier 162
2.4 Opérations nécessaires pour utiliser un fichier 163

3. Création d'un nouveau fichier binaire : écriture séquentielle 163
3.1 Structure des enregistrements 163
3.2 Programme de création du fichier 164
4. Lecture séquentielle d'un fichier binaire et écriture dans un fichier texte . 167
4.1 Structure des enregistrements 167
4.2 Programme de lecture du fichier 167
5. Ajout d'un tableau d'enregistrements au fichier binaire 169
6. Lecture directe et mise à jour d'un fichier binaire 171
6.1 Programme de modification du fichier 171
6.2 Anomalie de fonctionnement des fonctions de gestion de fichiers . . 173
7. Différences de codage entre un fichier binaire et un fichier texte 174
8. Travail pratique : fichier des employés 175
8.1 Sujet 175
8.2 Fichier des employés : proposition de correction 178

Chapitre 1-9

Algorithmique - Présentation de la méthode

1. Objectifs du chapitre 183
2. Présentation de la méthode 183
2.1 Pourquoi une méthode de programmation ? 183
2.2 Quelle méthode, pour quels types d'applications ? 184
2.3 La démarche hiérarchique 185
3. Exemple : édition de factures 185
3.1 État de sortie à obtenir 185
3.2 Décomposition de l'état de sortie 186
3.3 Le fichier des factures 188
3.4 Décomposition hiérarchique du fichier d'entrée 189
3.5 Ensemble de la démarche 190
3.6 Validation 191
3.7 Structure du programme 192
3.8 Le fichier des factures : organigramme 193
3.9 Les instructions 194
3.10 Le fichier des factures : instructions 194
3.11 Le fichier des factures : programme complet 202

4. Travail pratique : statistique des ventes 203
4.1 Commentaire sur l'énoncé 203
4.2 Sujet 204
4.3 Proposition de correction 208

Chapitre 1-10
Algorithmique - Groupes alternatifs de données

1. Préambule 215
2. Mise à jour du fichier "stock" 215
2.1 Sujet 215
2.2 Décompositions hiérarchiques 218
2.3 Validation 219
2.4 Organigramme 220
2.5 Instructions 220
2.6 Programme 222
2.7 Simplification des tests : l'astuce de la valeur maximale 224

Chapitre 1-11
Algorithmique - Données de phase, sauts de page

1. Objectifs du chapitre 227
2. État des chiffres d'affaires par client et secteur 227
2.1 Sujet 227
2.2 Décompositions 229
2.3 Organigramme 232
2.4 Instructions 233
2.5 Programme sans saut de page 234
2.6 Sauts de page 235
2.7 Programme avec saut de page 236
3. Travail pratique : édition de factures 238
3.1 Sujet 238
3.2 Édition de factures : proposition de correction 247

Chapitre 1-12
Algorithmique - Les tableaux

1. Préambule 265
2. Statistiques RATP - RER Ligne A 265
   2.1 Sujet 265
   2.2 Décompositions 267
   2.3 Validation 268
   2.4 Organigramme (première version) 270
   2.5 Instructions 270
   2.6 Programme 273
3. Travail pratique : statistiques sur les chiffres d'affaires 277
   3.1 Sujet 277
   3.2 Statistiques sur les chiffres d'affaires : proposition de correction . . . 281

Chapitre 1-13
Algorithmique - Alternatives complexes

1. Préambule 289
2. Mise à jour du fichier "stock" avec incidents 289
   2.1 Sujet 289
   2.2 Décompositions hiérarchiques 292
   2.3 Validation 294
   2.4 Alternatives complexes et organigramme 295
   2.5 Organigramme 297
   2.6 Instructions 297
   2.7 Programme 300
3. Mise à jour du fichier des employés 302
   3.1 Sujet 302
   3.2 Décompositions hiérarchiques 304
   3.3 Validation 305
   3.4 Organigramme 305
   3.5 Programme 308
4. Travail pratique : mise à jour de l'historique des ventes 310
   4.1 Sujet 310
   4.2 Mise à jour de l'historique des ventes : proposition de correction . . . 313

Partie 2 : Programmation objet - Java

Chapitre 2-1
Présentation

1. Le langage Java 319
   1.1 Historique 319
   1.2 Les caractéristiques principales 319
   1.3 Les outils 321
2. Les objectifs du cours Java - Java EE 324
   2.1 La programmation objet - Java 324
   2.2 Java EE 324
3. Les architectures développées 325
   3.1 2-tiers client lourd 325
   3.2 3-tiers client lourd 325
   3.3 3-tiers client lourd (XML) 325
   3.4 3-tiers client lourd (objets distants) 326
   3.5 3-tiers client léger 326
   3.6 3-tiers client lourd 326
   3.7 4-tiers client léger 326
4. Organisation des dossiers, pour les applications Java et Java EE 327
5. Les programmes utilitaires : packages UtilitairesMG et UtilitairesDivers. 328
   5.1 UtilitairesMG : liste des classes utilitaires 328
   5.2 UtilitairesDivers : liste des classes utilitaires 330

Chapitre 2-1
Éléments syntaxiques

1. Objectifs du chapitre 331
2. Premier programme 331
   2.1 Programme 331
   2.2 La classe 332
   2.3 La méthode main() 333
   2.4 Affichage dans la console 333
   2.5 Saisie clavier dans la console java : classe Clavier 334
   2.6 Instructions import 335

3. Les commentaires. 336
4. Les noms des identificateurs 336
4.1 Variables, propriétés, méthodes 336
4.2 Classes 337
5. Les types de base 338
5.1 Entiers 338
5.2 Booléens 338
5.3 Flottants 338
6. Quelques opérateurs 339
6.1 Opérateurs arithmétiques. 339
6.2 Opérateurs de comparaison 339
6.3 Opérateurs d'affectation. 339
6.4 Opérateurs logiques 340
6.5 Opérateurs ++ et -- 340
6.6 Opérateurs de conversion (cast). 340
7. Structures logiques du langage Java 341

Chapitre 2-3
Programmation objet, notions de base

1. Objectifs du chapitre 343
2. Classe, propriétés, référence, instanciation 343
2.1 Classe Point 343
2.2 Représentation UML de la classe Point 344
2.3 Utilisation de la classe Point. 344
3. Encapsulation. 347
3.1 Le mot-clé private 347
3.2 Méthodes d'instance 348
3.3 Conception : Getters, Setters 350
4. Travail pratique : classe Employe 352
4.1 Objectif 352
4.2 Sujet 352
4.3 Travail 353
4.4 Annexe : la classe String 353
4.5 Classe Employe : proposition de correction 354

5. Constructeur 354
5.1 Classe Point 354
5.2 Classe principale 355
6. Surcharge de méthodes - Surdéfinition 356
6.1 Classe Point 356
6.2 Classe principale 357
7. Première approche de l'héritage - Surcharge - Redéfinition 358
7.1 L'héritage 358
7.2 La redéfinition 360
7.3 Mécanisme de l'héritage 360
8. Propriétés d'instance, propriétés de classe 361
8.1 Déclaration 361
8.2 Conception 361
9. Méthodes d'instance, méthodes de classe 361
9.1 Exemple : classe Point 362
9.2 Classe principale 363
10. Constructeurs et méthodes 'private' 364
10.1 Classe Lapin 364
10.2 Classe TestLapin 365
11. Ramasse-miettes (garbage collector) 366
12. Travail pratique : classe Employe (deuxième version) 367
12.1 Objectif 367
12.2 Sujet 367
12.3 Travail 368
12.4 Classe Employe (deuxième version), proposition de correction 368
13. Travail pratique : voitures 370
13.1 Objectif 370
13.2 Sujet 370
13.3 Travail 372
13.4 Voitures : proposition de correction 372

Chapitre 2-4
Chaînes de caractères

1. Objectifs du chapitre 379
2. Classes de traitement des chaînes de caractères 379
   2.1 String et StringBuffer 379
   2.2 Programme de test 380
   2.3 Les performances de String et StringBuffer 381
3. Travail pratique : String et StringBuffer 381
   3.1 Objectifs 381
   3.2 Sujet 381
   3.3 String et StringBuffer, proposition de correction 382

Chapitre 2-5
Les tableaux

1. Objectifs du chapitre 393
2. Définition 393
3. Exemples d'utilisation de tableaux 393
   3.1 Tableau de variables de type primitif (int, double, float...) 393
   3.2 Tableau de références (String) 395
   3.3 Tableau à deux dimensions 396
   3.4 Tableau irrégulier 398
4. Travail pratique : Voitures (avec tableaux) 399
   4.1 Objectif 399
   4.2 Sujet 399
   4.3 Travail 401
   4.4 Voitures (avec tableaux) : proposition de correction 401
   4.5 Classe Vendeur 403

Chapitre 2-6
L'héritage

1. Objectifs du chapitre 405
2. Héritage, polymorphisme 405
   2.1 Classe Cercle 405
   2.2 Classe CercleCouleur 407
   2.3 Programme de test 410
   2.4 Méthodes surchargeables dans un constructeur 413
3. Travail pratique : Animal 415
   3.1 Objectif 415
   3.2 Sujet 415
   3.3 Animal : proposition de correction 417
4. Classes abstraites 421
   4.1 Définition 421
   4.2 Exemple du TP Animal 422
5. Interfaces 425
   5.1 Présentation 425
   5.2 Exemple d'interfaces 425
   5.3 Implémentation d'interfaces 425
   5.4 Exemple d'utilisation d'interface 427
6. Travail pratique : surfaces comparables 429
   6.1 Objectif 429
   6.2 Sujet 429
   6.3 Surfaces comparables : proposition de correction 430
7. La visibilité 433
   7.1 Visibilité des classes 434
   7.2 Visibilité des membres 434

Chapitre 2-7
Les collections

1. Objectifs du chapitre 435
2. Introduction 435
   2.1 Définition 435
   2.2 Hiérarchie partielle des collections. 436
   2.3 Vector ou ArrayList ? 436
3. Exemples d'utilisation de collections 437
   3.1 ArrayList de String 437
   3.2 Vector de String 439
4. Travail pratique : Voitures (avec collections) 439
   4.1 Objectif 439
   4.2 Sujet 439
   4.3 Voitures (avec collections de type ArrayList) : proposition de correction 440

Chapitre 2-8
Les exceptions

1. Objectifs du chapitre 443
2. Mécanisme des exceptions 443
   2.1 Compréhension des messages 443
   2.2 Exceptions surveillées, exceptions non surveillées 444
   2.3 Traitement des exceptions : try... catch... finally 447
   2.4 En résumé 450
3. Exemples d'utilisation des exceptions 450
   3.1 Classe Fraction 450
   3.2 Saisie du numérateur et du dénominateur dans un bloc try 451
   3.3 try... catch imbriqués 452
   3.4 finally 453
4. Création et lancement d'une exception 455
5. Création d'une nouvelle classe d'exception 455

6. Travail pratique : saisies d'entiers 457
6.1 Sujet 457
6.2 Saisies d'entiers : proposition de correction 458

Chapitre 2-9
Les flux

1. Objectifs du chapitre 463
2. Définition 463
3. Les flux de la classe System 463
3.1 System.out 463
3.2 System.err 465
3.3 System.in 465
4. Codage et flux de texte 466
4.1 Utilisation de la méthode read() de la classe InputStream 466
4.2 Classes Reader, InputStreamReader 467
4.3 Classe BufferedReader 470
5. Les fichiers 472
5.1 Classe File 472
5.2 Classes FileInputStream, FileOutputStream 473
6. Les fichiers texte 476
6.1 Recopie d'un fichier texte avec changement de codage 476
7. Travail pratique : total des notes 479
7.1 Objectif 479
7.2 Sujet 479
7.3 Ressource 480
7.4 Total des notes : proposition de correction 480
8. Travail pratique : fichier texte des contacts 482
8.1 Objectif 482
8.2 Sujet 483
8.3 Exemple d'exécution 483
8.4 Fichier texte des contacts : proposition de correction 484
9. Les fichiers binaires 488
9.1 Classes DataInputStream, DataOutputStream 488
9.2 RandomAccessFile 492

10. Travail pratique : fichier binaire des contacts 495
10.1 Objectif 495
10.2 Le fichier binaire contacts.dat 495
10.3 Données numériques en C et en Java 496
10.4 La classe Fichier (package utilitairesMG.divers) 496
10.5 Réflexion 497
10.6 Travail à effectuer 497
10.7 Fichier binaire des contacts : proposition de correction 498

Chapitre 2-10
Fenêtres - Évènements

1. Objectifs du chapitre 503
2. AWT - SWING 503
3. Définitions 504
3.1 Component 504
3.2 Container 504
3.3 JComponent 504
4. Hiérarchie des composants (extrait) 505
5. Fenêtre : création et affichage 506
5.1 Fonctionnement par défaut 506
5.2 Modification de quelques paramètres (position, dimensions, fermeture) 507
6. Évènements 508
6.1 Introduction 508
6.2 Objets évènements 508
6.3 Écouteurs d'évènements 509
6.4 Étapes d'écriture d'une application graphique avec écoute d'évènements 510
7. Conception de programmes avec écoute d'évènements 511
7.1 Fenêtre avec écoute des évènements WindowEvent : WindowListener 511
7.2 Fenêtre avec écoute des évènements WindowEvent : WindowAdapter 514
7.3 Fenêtre écouteur des évènements WindowEvent 515

8. EDT (Event Dispatching Thread) 517
8.1 Thread 517
8.2 Event Dispatching Thread (EDT) 517
9. Travail pratique : Souris 521
9.1 Objectif 521
9.2 Sujet 521
9.3 Recherche de documentation 522
9.4 Conseils 522
9.5 Souris : proposition de correction 522

Chapitre 2-11
Contrôles - Layout

1. Objectifs du chapitre 527
2. Composants dans une fenêtre : JPanel, JButton 527
2.1 Fonctionnement général 527
2.2 Modification des tailles préférées des boutons 531
3. Les layouts 532
3.1 Présentation 532
3.2 BorderLayout 532
3.3 Conception graphique 534
4. Travail pratique : Cases à cocher 537
4.1 Objectif 537
4.2 Sujet 537
4.3 Renseignements utiles 538
4.4 Cases à cocher : proposition de correction (BorderLayout, FlowLayout) 538
4.5 Cases à cocher : proposition de correction (GridLayout, FlowLayout) 540
5. Barres de défilement 541
5.1 JScrollPane 541
5.2 JScrollPane - FlowLayoutMG 543

6. Dynamique des composants 543
6.1 Définition 543
6.2 Ajout et suppression de boutons dans un panneau 544
6.3 Dynamique des composants et scrolling 545
7. Travail pratique : Dynamique des composants 547
7.1 Objectifs 547
7.2 Sujet 547
7.3 Dynamique des composants : proposition de correction 548

Chapitre 2-12
Dessins

1. Objectifs du chapitre 553
2. Fonctionnement général 553
2.1 La méthode paint(Graphics g) 553
2.2 Le contexte graphique 554
2.3 Dessin dans un JPanel 554
2.4 Premier dessin : un simple trait 554
2.5 Dessins dynamiques - Méthode repaint() 556
2.6 Dessins dynamiques - Tracé de clics de souris - Version 1 558
2.7 Dessins dynamiques - Tracé de clics de souris - Version 2 559
3. Travail pratique : Tracé de clics de souris permanents 561
3.1 Objectifs 561
3.2 Sujet 561
3.3 Tracé de clics de souris permanents : proposition de correction 562
4. Travail pratique : Les petits carrés 563
4.1 Objectif 563
4.2 Sujet 564
4.3 Les petits carrés : proposition de correction 566

Chapitre 2-13
JTable - DAO - MVC

1. Objectifs du chapitre 577
2. Fonctionnement par défaut d'une JTable 577
   - 2.1 Fenêtre affichée 577
   - 2.2 Classe Fenetre : utilisation d'une JTable 578
   - 2.3 Classe Controleur : classe principale de l'application 579
   - 2.4 Conception 581
3. Modèle de table 582
   - 3.1 Définition 583
   - 3.2 Classes et interfaces pour créer un modèle de table 583
   - 3.3 Modèle de table - Classe Colonne 584
   - 3.4 Modèle de table spécialisé - Classe Contact 590
4. Modèle de colonnes 593
   - 4.1 Définition 593
   - 4.2 Classes et interfaces pour créer un modèle de colonnes 594
   - 4.3 Modèle de table et modèle de colonnes 594
5. JTable éditable 599
   - 5.1 Fenêtre affichée 599
   - 5.2 Conception des classes ModeleTable et ModeleTableContact 599
   - 5.3 Autres classes de l'application 603
   - 5.4 En résumé 603
6. Le package utilitairesMG.graphique.table 603
   - 6.1 ModeleTable 603
   - 6.2 ModeleColonneTable 604
7. Couche d'accès aux données - DAO (Data Access Object) 606
   - 7.1 Application JTable éditable (paragraphe 5) 606
   - 7.2 La couche d'accès aux données : la DAO 607
8. Modèle MVC (Modèle - Vue - Contrôleur) 608
   - 8.1 Définition 608
   - 8.2 Application "JTable éditable", version finale 608
   - 8.3 Diagramme de séquence 610
   - 8.4 Classe Controleur 610
   - 8.5 Classe Fenetre 611
   - 8.6 Classe ModeleTableContact 612

8.7 Classe ContactDAO 614
9. Travail pratique : Projet GestionContactLocal 615
9.1 Objectif 615
9.2 Sujet 615
9.3 GestionContactLocal : proposition de correction 618
9.4 GestionContactLocal : intérêt du MVC et de la couche DAO 627

Chapitre 2-14
JDBC - Mapping Objet/Relationnel

1. Objectifs du chapitre 629
2. Définition 629
3. Communication Java - Base de données 629
3.1 Pilotes (drivers) JDBC 629
3.2 Schéma de communication 630
4. Travailler avec JDBC 630
4.1 Exécution d'une requête SQL de type SELECT 630
4.2 Structure générale du programme pour une requête SELECT 631
4.3 Exemple : base de données utilisée 632
4.4 Exécution d'un SELECT 632
4.5 Correspondance entre types SQL et Java 637
4.6 Exécution d'une requête SQL de type UPDATE 638
5. Présentation des classes utilitaires 639
5.1 JeuResultat 639
5.2 JeuResultatXML 641
5.3 Classe BaseDeDonnees 643
5.4 Classe AccesBase 644
5.5 Exécution d'une requête SQL de type SELECT avec les utilitaires . . 645
6. Travail pratique : Projet GestionContactJdbc - Version 1 646
6.1 Objectif 646
6.2 Architecture du projet : 2 tiers 647
6.3 Sujet 647
6.4 Conception 647
6.5 Programmation 647
6.6 GestionContactJdbc - Version 1 : proposition de correction 648

7. JDBC : compléments 651
7.1 Table utilisée dans les exemples 651
7.2 Exécution de requêtes INSERT et DELETE 652
7.3 Requêtes exécutées en Batch 653
7.4 Transactions 654
8. Le Mapping Objet/Relationnel 655
8.1 Exposé du problème à résoudre 655
8.2 Comment traduire l'association entre les deux objets ? 659
9. Travail pratique : Projet Mapping 661
9.1 Objectif 661
9.2 Sujet 661
9.3 Documents joints 662
9.4 Mapping : proposition de correction 665
9.5 Packages metierMapping, daoJdbcMapping 673
10. Travail pratique : Projet GestionContactJdbc 673
10.1 Objectifs 673
10.2 Sujet 673
10.3 GestionContactJdbc : Proposition de correction 674

Chapitre 2-15
Programmation réseau - Architecture 3-tiers

1. Objectifs du chapitre 679
2. Définitions 679
2.1 Serveur 679
2.2 Protocole 680
3. Programmation des "sockets" 680
3.1 Présentation 680
3.2 Programme d'accès à un serveur web 680
4. Implémentation de serveurs d'applications 683
4.1 Choix de dénomination des variables 683
4.2 ServerSocket 683
4.3 Un serveur simple et son client 684
4.4 Un serveur d'objets sérialisés et son client 688
4.5 Un serveur d'objets sérialisés capable de servir plusieurs clients 693

5. Travail pratique : Projet ServeurObjets 699
5.1 Objectifs 699
5.2 Architecture du projet : 3-tiers 699
5.3 Sujet 699
5.4 Projet ServeurObjets : proposition de correction 702
6. Travail pratique : Projet GestionContactReseau 709
6.1 Objectif 709
6.2 Écrans du client et du serveur 709
6.3 Sujet 709
6.4 Projet GestionContactReseau : proposition de correction 711

Chapitre 2-16
XML - Architecture 3-tiers

1. Objectifs du chapitre 721
2. XML : eXtensible Markup Language 721
2.1 Le langage XML 721
2.2 Document XML 722
3. Java et XML 723
3.1 Le DOM (Document Object Model) 723
3.2 La hiérarchie des interfaces Node (extrait) 724
4. Parseurs XML 725
4.1 Parseur DOM 725
4.2 La DTD (Document Type Definition) 728
4.3 Le parseur DOM et la DTD 729
4.4 Le parseur SAX 734
5. Travail pratique : Projet ServeurXML 736
5.1 Objectifs 736
5.2 Architecture du projet : 3-tiers 736
5.3 Sujet 736
5.4 Projet ServeurXML : proposition de correction 741

6. Travail pratique : Projet GestionContactXML 743
6.1 Objectif 743
6.2 Écrans du client et du serveur 743
6.3 Sujet 744
6.4 Projet GestionContactXML : proposition de correction 745

Partie 3 : Java EE

Chapitre 3-1
Présentation

1. Java EE (Java Entreprise Edition) 753
1.1 Les classes de Java EE 753
1.2 Le serveur Java EE 754
2. Architectures réparties mises en œuvre dans la partie JEE 754
2.1 3 tiers client léger 754
2.2 3 tiers client lourd 755
2.3 4 tiers client léger 755

Chapitre 3-2
Servlet

1. Objectif des chapitres sur les Servlets et les JSP 757
2. Servlets : généralités 758
2.1 Définition 758
2.2 Traitement d'une requête par une Servlet 758
2.3 Classes et interfaces pour l'utilisation d'une HttpServlet 759
2.4 Servlets et ServletContext 760
2.5 Cycle de vie d'une Servlet 760
2.6 Déploiement, fichier web.xml 761
3. Première Servlet : projet Servlet1 762
3.1 Écrans du projet Servlet1 762
3.2 Enchaînement des programmes et des écrans du projet Servlet1 . . . 762
3.3 index.html 763
3.4 Classe ServletValidation 764
3.5 Classe Validation 769

3.6 Classe Utilisateur 770
3.7 Remarques de conception. 771

Chapitre 3-3
JSP - MVC

1. Présentation 773
2. Première JSP : projet ServletJsp1. 773
2.1 Projet ServletJsp1 773
2.2 Enchaînement des programmes et des écrans du projet Servlet1 . . . 774
2.3 La JSP resultat2.jsp 774
3. Servlet ou JSP ? 776
3.1 Servlet/JSP : comment les utiliser ? 776
3.2 Servlet/JSP : un choix de conception 777
4. Le modèle MVC (Modèle-Vue-Contrôleur) en Web 777
4.1 Rappel : architecture MVC pour une application client lourd. 777
4.2 Architecture MVC pour une application web 777
4.3 L'application ServletJsp1 respecte le modèle MVC. 778

Chapitre 3-4
Servlet - JSP : compléments

1. Transmission de données entre Servlet et JSP : projet ServletJsp2. 781
1.1 Écrans du projet ServletJsp2. 781
1.2 Enchaînement des programmes et des écrans du projet ServletJsp2 . 783
1.3 Transmission de données entre Servlet et JSP par session 784
1.4 Transmission de données entre Servlet et JSP par l'objet ServletContext. 788
1.5 Transmission de données entre Servlet et JSP par cookies. 790
1.6 Transmission de données entre Servlet et JSP : en résumé 793
2. La réentrance 793
2.1 Projet ReEntrance 794
2.2 Conclusion sur la réentrance 797

3. Le MVC2 797
3.1 Définition 797
3.2 Application au projet ServletJsp2 : projet ServletJsp2MVC2 797
4. Travail pratique : projet GestionContactMVC2 801
4.1 Objectifs 801
4.2 Sujet 802
4.3 GestionContactMVC2 : Proposition de correction 806

Chapitre 3-5
Objets distants - RMI - EJB

1. Objectifs 825
2. Principe des objets distants 825
2.1 Schéma 826
2.2 Fonctionnement général 826
3. Création et utilisation d'objets distants en Java (RMI) 828
3.1 RMI 828
3.2 Exemple : un objet distant de tri 828
4. Objets distants et JEE : Enterprise Java Beans (EJB 3) 833
5. Qu'est-ce qu'un EJB ? 833
5.1 Les différents types d'EJB 834
5.2 L'annuaire pour les EJB Session 834
5.3 Conteneur EJB - conteneur web 834
5.4 Différence JavaBeans/EJB 834

Chapitre 3-6
Les EJB Session

1. Objectif des chapitres sur les EJB (Enterprise Java Beans) 835
2. Types d'EJB Session 836
2.1 EJB Session avec état (stateful) 836
2.2 EJB Session sans état (stateless) 836
2.3 Utilisation Remote/Local 837

3. Outils et conventions utilisés dans ce chapitre 838
3.1 Outils 838
3.2 Conventions de nommage utilisées 838
4. Exemple : un objet distant de tri 838
4.1 Création de l'EJB Session 838
4.2 Déploiement 839
4.3 Création d'un projet client 840
5. Cycle de vie de l'EJB Session : @PostConstruct, @PreDestroy 842
6. Travail pratique : Contact distant 843
6.1 Objectifs 843
6.2 Sujet 843
6.3 Contact distant : proposition de correction 844
7. Pools de connexions 848
7.1 Le problème des connexions à un serveur de base de données 848
7.2 Les pools de connexions 849
7.3 Utilisation d'un pool de connexions pour l'accès à la base de données 850
8. Travail pratique : Projet GestionContactEJB 852
8.1 Objectifs 852
8.2 Sujet 853
8.3 GestionContactEJB : proposition de correction 855

Chapitre 3-7
Les EJB Entity

1. Introduction 861
1.1 Définition 861
1.2 Framework de persistance 861
1.3 Contexte de persistance 861
1.4 Interface EntityManager 862
1.5 Unité de persistance 863
1.6 Transactions 864

2. Exemple complet : gestion des contacts. 864
2.1 Objectif 864
2.2 Généreration des classes EBJ Entity. 865
2.3 Création de l'EJB Session MappingEntite 868
2.4 Application client 877
3. Travail pratique : Projet GestionContactEJBEntite 878
3.1 Objectifs 878
3.2 Sujet. 878
3.3 GestionContactEJBEntite : proposition de correction 879

Index

Langage C - Algorithmique 885
Programmation objet - Java 892
Java EE 901

Chapitre 1-9
Algorithmique - Présentation de la méthode

1. Objectifs du chapitre

- Décomposer hiérarchiquement un ensemble de données.
- À partir des décompositions des données écrites (données de sortie) et des données lues (données d'entrée) par le programme, établir l'organigramme structuré de la logique de l'application.
- Écrire le programme à partir de l'organigramme.

2. Présentation de la méthode

2.1 Pourquoi une méthode de programmation ?

Dans un projet informatique, la première chose qu'exprime en général le futur utilisateur, c'est le résultat qu'il souhaite. "Je veux un programme pour gérer mon stock, je souhaite un site web pour présenter ma société...". Il faut alors regarder en quoi consiste le stock, quels sont les renseignements que le site web doit présenter ? Etc.

Enfin, ayant établi avec précision la liste des données disponibles, et l'ensemble des résultats à obtenir, le développeur programme la logique qui permet d'arriver au résultat voulu. On dit qu'il conçoit l'**algorithme** de la solution.

Remarque

Une recette de cuisine est un algorithme. Quel est le plat que je veux réaliser ? Ai-je les ingrédients pour le réaliser ? Enfin, comment m'y prendre ? Quelle est la recette ?

L'art de l'informaticien est de découvrir la recette, l'algorithme à programmer.

Dans les chapitres précédents, nous avons réalisé des programmes sans utiliser de méthode. Un raisonnement logique, traduit convenablement en instructions structurées du langage (`if`, `while`...) suffit souvent pour résoudre les problèmes.

Quand la complexité augmente, que l'ensemble des données à traiter est important, l'algorithme devient moins évident à trouver.

– Le premier intérêt d'une méthode est d'aider l'informaticien à concevoir l'algorithme.
– Le deuxième avantage, c'est qu'une méthode propose des étapes pour arriver au résultat.

Remarque

Si deux personnes suivent les mêmes étapes pour trouver un algorithme, elles arriveront à des programmes semblables. Au sein d'une entreprise, ***une méthode facilite la communication et la maintenance des programmes****.*

2.2 Quelle méthode, pour quels types d'applications ?

La méthode et les notations proposées dans les chapitres concernant l'algorithmique, sont inspirées des travaux de Jean Dominique Warnier, auteur de nombreux travaux sur les systèmes d'information. Plus précisément, les notations (accolades par exemple) sont tirées de sa méthode de Logique de Construction de Programmes.

– La méthode consiste à :
  – étudier les données de sortie (impressions, mises à jour), en examinant la façon dont elles sont organisées et structurées ;
  – étudier les données d'entrée (fichiers), en examinant la façon dont elles sont organisées et structurées ;
  – déduire de ces deux études l'algorithme à programmer.
– Les applications que nous allons réaliser en suivant cette méthode sont connues sous le nom de traitements "**batch**". C'est une partie de l'informatique parfois qualifiée "d'ancienne", mais qui reste d'actualité.
  – Une banque doit mettre à jour les comptes de ses clients en fonction de leurs dépenses et de leurs recettes. Une commune doit éditer son budget. Une entreprise de vente par Internet doit mettre à jour ses stocks. Les quantités d'informations manipulées sont parfois considérables.

– Les informations sont stockées dans des fichiers (ou des bases de données), préparés en vue du traitement.
– Une fois le traitement de ces données lancé, il n'y a plus d'intervention humaine avant l'obtention du résultat. **C'est un traitement batch**. Cette absence de dialogue avec l'utilisateur s'oppose aux traitements conversationnels, où l'utilisateur dialogue avec le programme par l'intermédiaire d'écrans de saisie et d'affichage.

Remarque

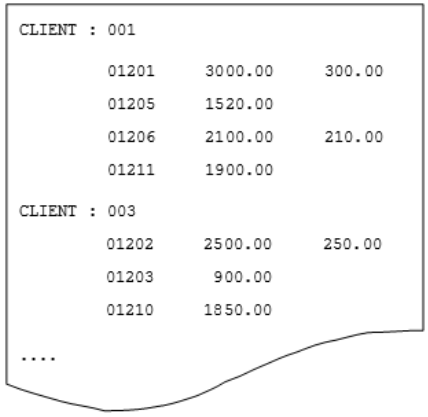
Le langage C est bien adapté aux applications batch. Pour le conversationnel, il est possible d'utiliser des bibliothèques graphiques spécialisées, ou d'utiliser un des nombreux enfants du C : C++, PHP, C#, Java.

2.3 La démarche hiérarchique

Il s'agit de trouver la logique avec laquelle un ensemble de données est organisé, en allant du général au particulier.

3. Exemple : édition de factures

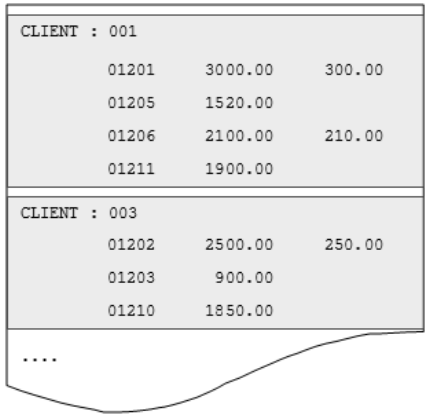
3.1 État de sortie à obtenir



Pour chaque client, on imprime des lignes "facture". Chacune d'entre elles contient un numéro de facture, le montant de la facture, et une remise de 10 % sur le montant de la facture si celui-ci dépasse 2.000 €.

3.2 Décomposition de l'état de sortie

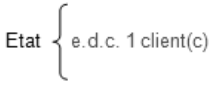
Quand on examine l'état, on constate qu'il est organisé "client par client" :



Chaque sous-ensemble grisé est un ensemble de données concernant un client.

L'ensemble de l'état peut être décrit comme **un ensemble de données concernant un client autant de fois qu'il y a de clients**.

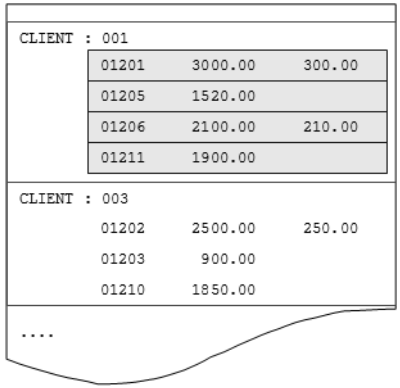
On peut le noter ainsi :



`e.d.c.` : abréviation pour « ensemble de données concernant ».

`(c)` : cardinalité répétitive pour : autant de fois qu'il y a de clients.

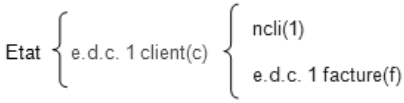
On s'intéresse maintenant à l'organisation des données d'un client, par exemple à celles du `client 001` :



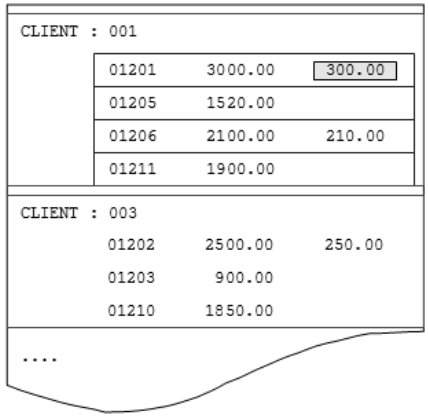
Chaque sous-ensemble grisé est un ensemble de données concernant une facture.

L'ensemble des données d'un client peut être décrit comme : un numéro de client (`ncli`) présent 1 fois, puis un **ensemble de données concernant une facture autant de fois qu'il y a de factures**.

On peut le noter ainsi :



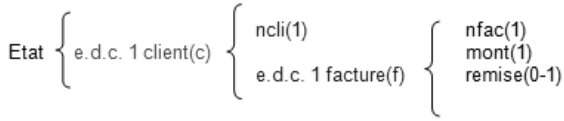
On s'intéresse maintenant à l'organisation des données d'une facture :



Le sous-ensemble grisé correspond à une remise.

L'ensemble des données d'une facture peut être décrit comme : un numéro de facture (`nfac`) présent 1 fois, un montant de facture (`mont`) présent une fois et **une remise présente 0 ou 1 fois**.

Décomposition complète :



3.3 Le fichier des factures

Les informations nécessaires à l'édition des factures sont lues dans un fichier d'entrée dont les enregistrements ont la structure suivante :

N° CLIENT	N° FACTURE	MONTANT

Ce fichier séquentiel est trié sur le N° CLIENT. Il ne comporte qu'un enregistrement par facture.

Structure correspondante :

```
typedef struct
{
    char  ncli[4];
    char  nfac[6];
    float mont;
} FACTURE;
```

On considère que le fichier des factures n'est pas vide.