Master Réseaux Informatiques d'Entreprise

Projet Base De Données

Rapport du Projet : Base De Données

Groupe

- Moïse Kahotwa Kyoghero
- Nathan Vidal Fages
- Samuel Monsempes
- Guillaume Fernandez

Juin 2021

Table des matières

1 Contexte

L'objectif du projet base de données est de se familiariser à la modélisation d'un Système de Gestion de Base de Données à partir d'un cahier des charges, le besoin demandé est la création d'un système d'enchère. Le modèle sera ensuite concrétisé par la mise en place dans une base de données Oracle.

Le devellopement sera fait en Java en utilisant l'API JDBC.

2 Dictionnaire

- **Creation d'un utilistaur** : À la creation de l'utilisateur, on demande l'email, le nom et le prénom de l'utilisateur.
- **Mise en place d'une enchère** : Pour une enchère, il est nécessaire de demander la somme mise par l'utilisateur et de préciser le nombre de produit qu'il faut enchérir.
  La date de l'enchère sera inscrite automatiquement.
- **Creation d'un produit** : Lors de la création d'un produit, nous devons demander le nom du produit, le prix de depart du produit et la quantité.
- **Creation d'une vente** : Pour la vente d'un produit, on voit le prix de la validation de la vente, si la vente est montante ou descendante, si la vente est revocable ou non, si la duré de la vente est libre ou non, avec sa date de fin. S'il est possible d'encherir plusieurs fois.

3 Schéma conceptuel entité-association

Le modèle entité-association (en anglais Entity-Relation model) fournit une description graphique pour représenter des modèles conceptuels de données sous la forme de diagrammes contenant des entités et des associations.

Il permet d'identifier et de caractériser les objets du domaine d'application et d'établir leurs liens, les cardinalités donnent des renseignements sur le minimum et le maximum d'occurrences d'une association liant une entité à une autre.

Au niveau conceptuel, le modèle distingue les objets et les relations. Les objets sont représentés par des rectangles, les relations des ellipses, les entités, objets ou relations, ont des propriétés ou attributs. Une relation (m,n) se traduit par un segment logique.

3.1 Liste des entités

— Utilisateur

— Vente

— Enchere

— Produit

3.2 Liste des attributs

1. **Utilisateur :**
   - **NumUtilisateur** : Nombre entier qui permet de rendre unique l'utilisateur
   - **emailUtilisateur** : Contient l'adresse mail de l'utilisateur
   - **nomUtilisateur** : Contient le nom de l'utilisateur
   - **prenomUtilisateur** : Contient le prenom de l'utilisateur
   - **adresseUtilisateur** : Contient l'adresse postal de l'utilisateur
2. **Enchere :**
   - **numEnchere** : Identifie à l'aide d'un entier l'enchère effectué par un utilisateur
   - **prixachatEnchere** : Prix founie par l'utilisateur
   - **dateEnchere** : La date de l'enchere effectué par l'utilisateur
   - **qteproduitEnchere** : La quantité de produit encherie

3. **Vente :**
    - **numVente** : Référence une vente par un numéro
    - **prixVente** : Le prix de la validation de la vente
    - **montantVente** : Si la vente et montante ou pas
    - **revocableVente** : Savoir si la vente et revocable ou pas
    - **dureelibreVente** : La durée de la vente
    - **datefinVente** : La date de fin de la vente
    - **encherirplusieurfoisVente** : est-il possible de réencherir?
4. **Produit :**
    - **numProduit** : numéro qui référence le produit
    - **nomProduit** : Le nom du produit
    - **prixrevientProduit** : Le prix de mise de départ du produit
    - **stockProduit** : Le stock du produit

3.3 Associations entre entités

Associations	Entités liées
• enchéri	• Utilisateur • Enchere
• sélectionne	• Enchere • Vente

3.4 Caractéristiques des associations

- **Association enchéri** : Sans attribut propre et de dimension 2, elle relie un utilisateur à une enchere. Elle est de cardinalité de 1-N car l'utilisateur peut encherir sur une ou plusieurs enchère.
- **Association Sélectionne** : Sans attribut propre et de dimension 2, elle relie une vente à une enchère. Elle est de cardinalité de 1-N car une vente peut avoir plusieurs enchère.

3.5 Construire le schéma Entité Association

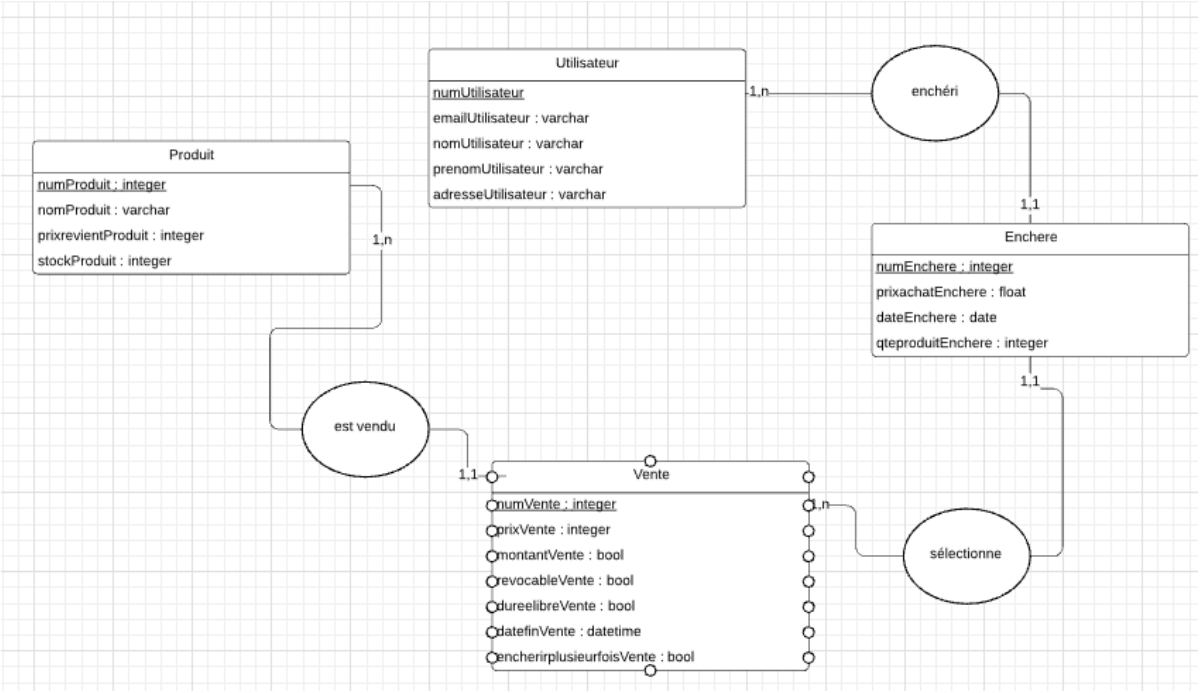


Schéma Entité Association

4 Passage du schéma conceptuel Entité-Association au schéma relationnel

D'après le schéma conceptuel Entité-Association, on a créé pour chaque entité une table.

Ainsi les premières tables qui ont été créées sont les suivantes :

- **Utilisateur**
- **Enchere**
- **Vente**
- **Produit**

Les clés primaires sont des nombres pour éviter de se soucier si une clé est belle et bien unique.

4.1 Les cardinalités 1-N :

Pour des raisons de cardinalité, l'association binaire **un à plusieurs** est traduit par un report de clé.

En effet, la clé primaire de l'entité participant côté N est ajoutée à l'autre entité en tant que clé étrangère. Les associations enchéries et sélectionnées, ont toutes des cardinalités N-1, ce qui implique que les clés primaires des premières entités seront des clés étrangères dans les deuxièmes entités :

- La clef primaire de l'entité Utilisateur (numUtilisateur) se retrouve en clef étrangère dans l'entité Enchere
- La même chose pour la clef primaire de l'entité vente (numVente), elle se retrouve en clef étrangère dans l'entité Enchere
- La même chose encore pour la clef primaire de l'entité produit(numProduit), elle se retrouve en clef étrangère dans l'entité Vente

4.2 Le modèle relationnel :

- **Produit**(numProduit,nomProduit,prixrevientProduit,stockProduit)
- **Utilisateur**(emailUtilisateur,nomUtilisateur,prenomUtilisateur,adresseUtilisateur)
- **Vente**(numVente,prixVente,montantVente,revocableVente,dureelibreVente, datefinVente,encherirplusieurfoisVente,#numProduit)
- **Enchere**(numEnchere,prixachatEnchere,dateEnchere,heureEchere, qteproduitEnchere,#emailUtilisateur,#numVente)

5 Création de la base de données

5.1 Creation des tables de la base de données

- La création des tables s'est obtenue grâce schéma relationnel logique fait dans la partie 3
- On a crée les tables avec la commande : CREATE TABLE nomDeLaTable (les attributs dont on a besoin)
- Les tables ne peuvent pas être créées dans n'importe quel ordre puisqu'elles sont reliées entre elles par des clés primaires et/ou étrangères

5.2 Contraintes métier

Plusieurs contraintes métier ont été identifiées :

- Les enchères peuvent être montantes ou descendantes. Dans les enchères montantes un prix de départ est fixé et les acheteurs surenchérissent les uns après les après les autres. Dans les enchères descendantes, aussi appelé enchères hollandaises [1], un prix de départ est fixé et est régulièrement abaissé jusqu'à qu'un acheteur se manifeste. Un booléen permet de différencier les deux et d'adapter le fonctionnement de l'application.
- Un objet est défini par la quantité de ce produit en stock. Si un même modèle est vendu plusieurs fois les enchères doivent s'adapter au stock et à la demande
- La vente peut être révoquée si le prix de vente est inférieur au prix de revient du produit
- L'enchère peut-être définie dans le temps. Une date de fin est ainsi déclarée au début de l'enchère. Si aucune limite n'est définie, l'enchère pourra être clôturée manuellement ou si un certain montant est atteint
- Les adresses e-mail des utilisateurs doivent être uniques. Ainsi une vérification est réalisée à la création ou à la modification de ce champ
- Après la clôture d'une enchère, les stocks du produit doivent être impactés par la quantité vendue

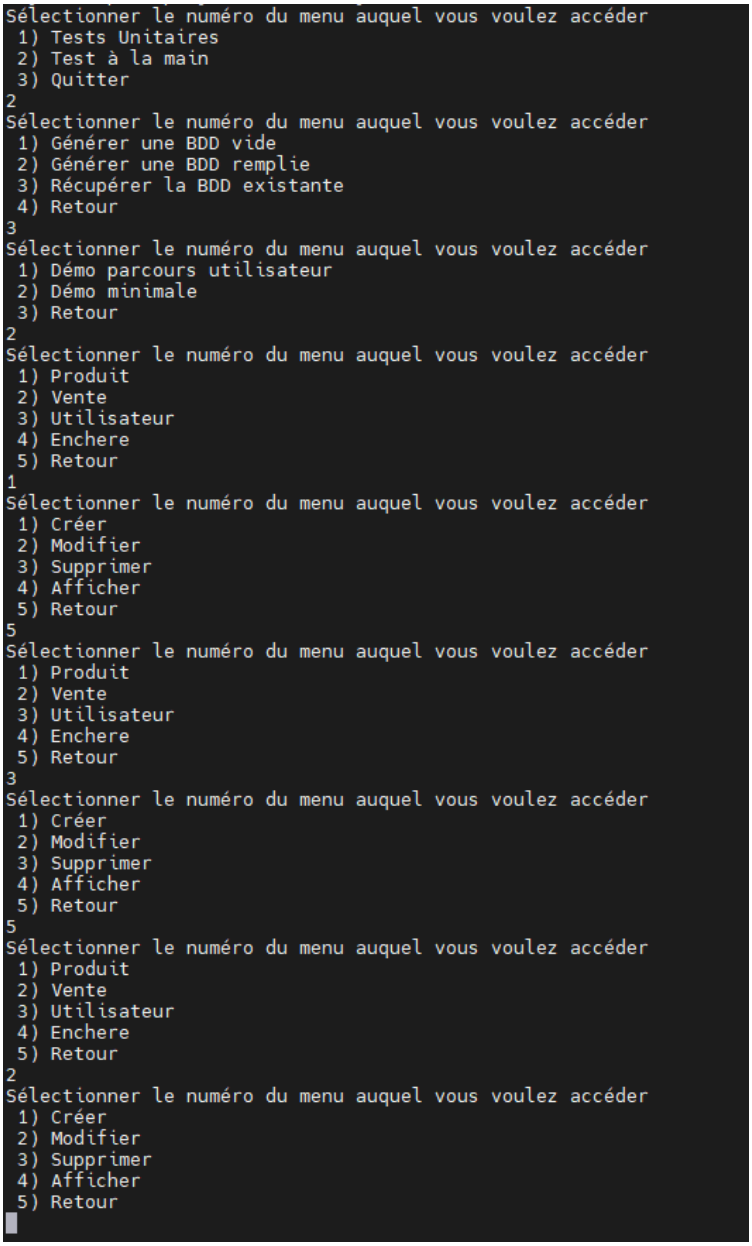
1. Dans ce mécanisme, l'organisateur de la vente annonce un prix de départ élevé, puis l'abaisse par étapes, jusqu'à ce qu'un enchérisseur se déclare preneur. Le bien est alors attribué à cet enchérisseur « le plus offrant », à un prix de cession égal à son offre, appelé « premier prix » (les offres des autres candidats restent, dans cette procédure, inconnues).

6 Conclusion

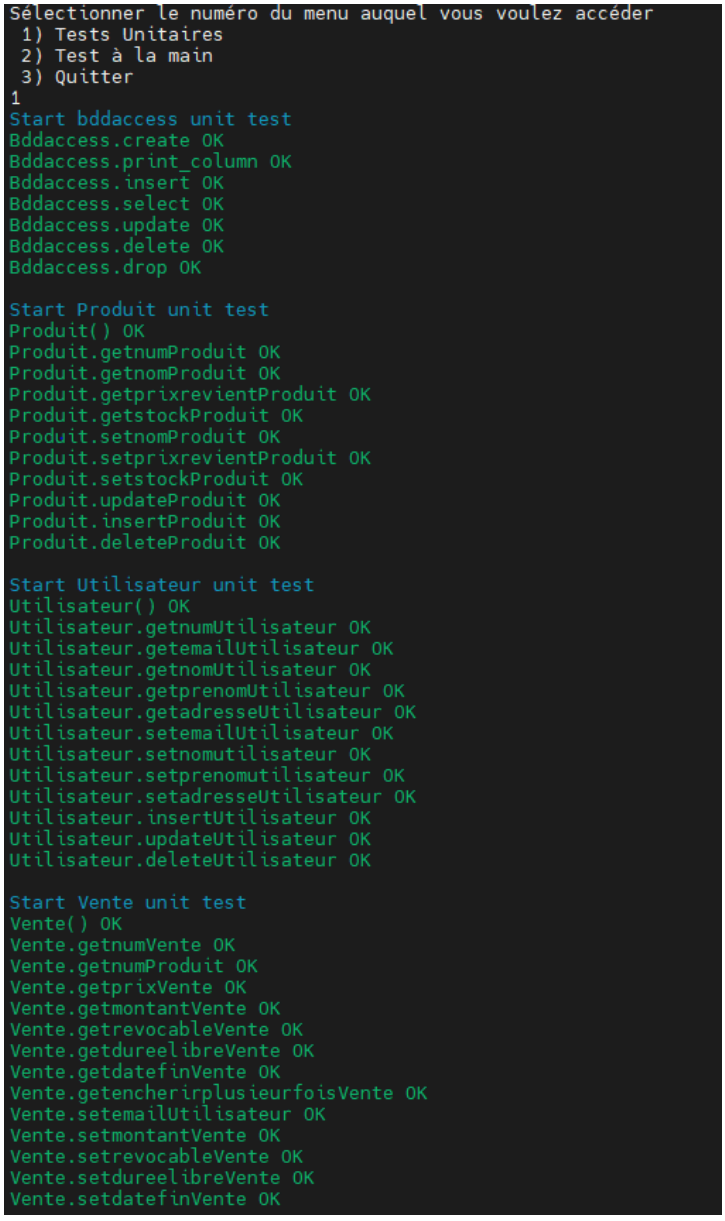
Ce projet, nous a permis d'appliquer les méthodologies vues en cours de base de données et en cours de java.

Vous pouvez trouver le code du projet ICI

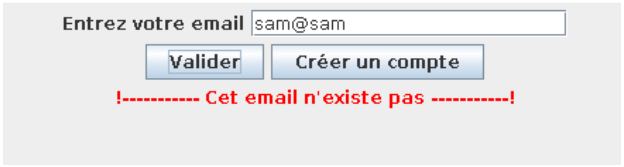
7 Annexe 1 : Images du programme Java



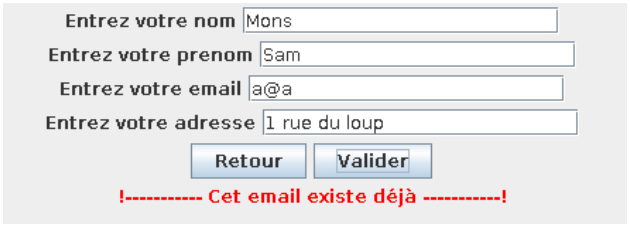
Menu CLI



Tests unitaires



Connection

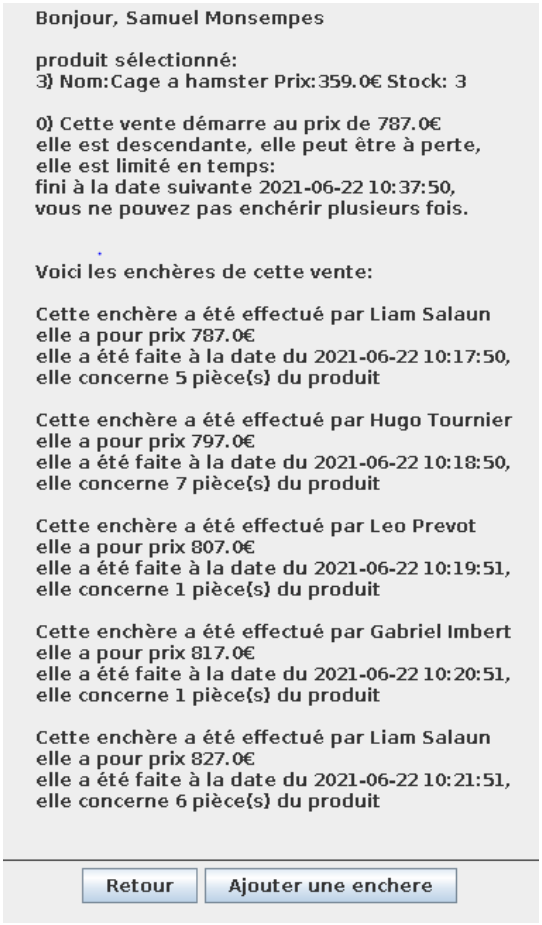


Créer un utilisateur

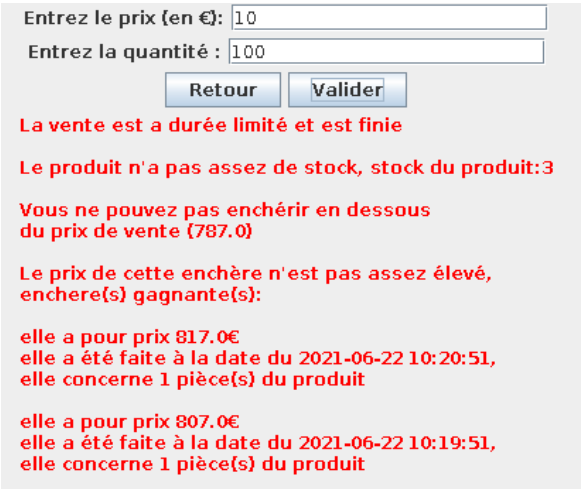
Selection ou creation d'un produit

Créer un produit

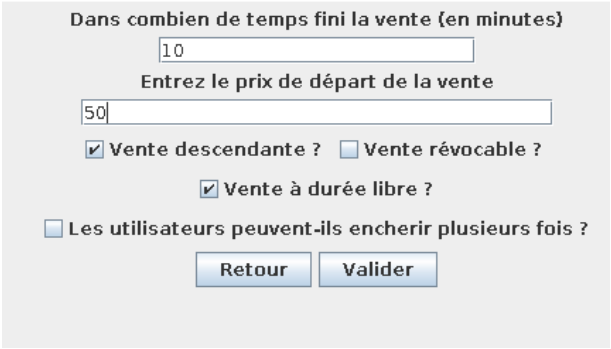
Vente en cours



Liste des enchères



Enchérir



Paramètres de l'enchère

8 Annexe 2 : Programme Java

Vous pouvez trouver les sources à l'adresse suivante :

```
1  import java.text.ParseException;
2  import javax.swing.*;
3  import java.awt.*;
4
5  class Main {
6      public static void main (String[] args)throws ParseException {
7          DemoMinimale demoMinimale = new DemoMinimale();
8          demoMinimale.menuPrincipal();
9      }
10 }
```

java/main.java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import java.util.ArrayList;


public class Bddaccess{

  public String[] select(String table){
    Connection conn = null;
    Statement stmt = null;
    String[] result = null;
    try {
            conn = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xe","system","djamel.2a");
            //System.out.println("Connection etablie.....");

            stmt = conn.createStatement() ;
            //System.out.println("Create  Statement.....");
      ResultSet rs = stmt.executeQuery("Select * From " + table);
      int size = rs.getMetaData().getColumnCount();
      ArrayList<String> auxresult = new ArrayList<String>();
      while(rs.next()){
        for (int i = 1; i <= size; i++){
          auxresult.add(rs.getString(i));
        }
      }
      result = auxresult.toArray(new String[0]);
      rs.close();
      //System.out.println("Table "+table+" selected");
    }catch(SQLException se){
      //System.out.println(se.getMessage());
    }
    finally{
      try{
        if(stmt!=null)
        conn.close();
      }catch(SQLException se){
      }
    }
```

```
43      return result;
44    }
45
46    public void drop(String table){
47      Connection conn = null;
48      Statement stmt = null;
49      try {
50              conn = DriverManager.getConnection(
51                      "jdbc:oracle:thin:@localhost:1521:xe","system","djamel.2a");
52              //System.out.println("Connection etablie.....");
53
54              stmt = conn.createStatement() ;
55              //System.out.println("Create  Statement.....");
56        stmt.execute("DROP TABLE "+table+" CASCADE CONSTRAINTS");
57        //System.out.println("Table "+table+" deleted");
58      }catch(SQLException se){
59        //System.out.println(se.getMessage());
60      }
61      finally{
62        try{
63          if(stmt!=null)
64          conn.close();
65        }catch(SQLException se){
66        }
67      }
68    }
69
70    public String[] print_column(String table){
71      Connection conn = null;
72      Statement stmt = null;
73      String[] result = null;
74      try {
75              conn = DriverManager.getConnection(
76                      "jdbc:oracle:thin:@localhost:1521:xe","system","djamel.2a");
77              //System.out.println("Connection etablie.....");
78
79              stmt = conn.createStatement() ;
80              //System.out.println("Create  Statement.....");
81        ResultSet rs = stmt.executeQuery("Select COLUMN_NAME  From all_tab_columns
      WHERE table_name like '" + table.toUpperCase()+"'");
82        int size = rs.getMetaData().getColumnCount();
83        ArrayList<String> auxresult = new ArrayList<String>();
84        while(rs.next()){
85          for (int i = 1; i <= size; i++){
86            auxresult.add(rs.getString(i));
87          }
```

```
88          }
89          result = auxresult.toArray(new String[0]);
90          rs.close();
91          //System.out.println("Table "+table+" selected");
92        }catch(SQLException se){
93          //System.out.println(se.getMessage());
94        }
95        finally{
96          try{
97            if(stmt!=null)
98            conn.close();
99          }catch(SQLException se){
100         }
101       }
102       return result;
103     }
104
105     public void insert(String table, String[] option){
106       Connection conn = null;
107       Statement stmt = null;
108       try{
109               conn = DriverManager.getConnection(
110                       "jdbc:oracle:thin:@localhost:1521:xe","system","djamel.2a");
111               //System.out.println("Connection etablie.....");
112
113               stmt = conn.createStatement() ;
114               //System.out.println("Create  Statement.....");
115         String column = "";
116         String value = "";
117         int i = 0;
118         for (i = 0; i < option.length/2; i++){
119           column += option[i];
120           if (i != (option.length/2)-1) column += ", ";
121         }
122         for (i = i; i < option.length; i++){
123           value += option[i];
124           if (i != (option.length-1)) value += ", ";
125         }
126         stmt.executeUpdate("INSERT INTO "+table+" ("+column+") VALUES ("+value+")")
        ;
127         //System.out.println("Insert line in table "+table);
128       }catch(SQLException se){
129         //System.out.println(se.getMessage());
130       }
131       finally{
132         try{
```

```
133           if(stmt!=null)
134           conn.close();
135         }catch(SQLException se){
136         }
137       }
138     }
139
140     public void update(String table, String[] option){
141       Connection conn = null;
142       Statement stmt = null;
143       try {
144               conn = DriverManager.getConnection(
145                       "jdbc:oracle:thin:@localhost:1521:xe","system","djamel.2a");
146               //System.out.println("Connection etablie.....");
147
148               stmt = conn.createStatement() ;
149               //System.out.println("Create  Statement.....");
150         String inSet = "";
151         int i = 0;
152         for (i = 0; i < option.length; i+=2){
153           inSet += option[i] +" = " + option[i+1] ;
154           if (i != (option.length-2)) inSet += ", ";
155         }
156         stmt.executeUpdate("UPDATE "+table+" SET "+inSet+" WHERE "+option[0]+" = "+
      option[1]);
157         //System.out.println("Insert line in table "+table);
158       }catch(SQLException se){
159         //System.out.println(se.getMessage());
160       }
161       finally{
162         try{
163           if(stmt!=null)
164           conn.close();
165         }catch(SQLException se){
166         }
167       }
168     }
169
170     public void delete(String table, String primarykey){
171       Connection conn = null;
172       Statement stmt = null;
173       try {
174               conn = DriverManager.getConnection(
175                       "jdbc:oracle:thin:@localhost:1521:xe","system","djamel.2a");
176               //System.out.println("Connection etablie.....");
177
```

```
178             stmt = conn.createStatement() ;
179             //System.out.println("Create  Statement......");
180       stmt.executeUpdate("DELETE FROM "+table+" WHERE "+primarykey);
181       //System.out.println("Delete line into table "+table);
182     }catch(SQLException se){
183       //System.out.println(se.getMessage());
184     }
185     finally{
186       try{
187         if(stmt!=null)
188         conn.close();
189       }catch(SQLException se){
190       }
191     }
192   }
193
194   public void create(String table, String option){
195     Connection conn = null;
196     Statement stmt = null;
197     try {
198             conn = DriverManager.getConnection(
199                     "jdbc:oracle:thin:@localhost:1521:xe","system","djamel.2a");
200             //System.out.println("Connection etablie......");
201
202             stmt = conn.createStatement() ;
203             //System.out.println("Create  Statement......");
204       stmt.execute("CREATE TABLE "+table+"( "+option+")");
205       //System.out.println("create table "+table);
206     }catch(SQLException se){
207       //System.out.println(se.getMessage());
208     }
209     finally{
210       try{
211         if(stmt!=null)
212         conn.close();
213       }catch(SQLException se){
214       }
215     }
216   }
217 }
```

java/Bddaccess.java

```
1 import java.util.Scanner;
2 import java.text.ParseException;
```

```
3  import java.util.ArrayList;
4  import javax.swing.*;
5  import java.awt.*;
6
7  public class DemoMinimale {
8
9    private Scanner saisieUtilisateur;
10   private Init init;
11
12   public DemoMinimale(){
13     this.saisieUtilisateur = new Scanner(System.in);
14     }
15
16   public void menuPrincipal()throws ParseException{
17     String menuPrincipal="Sélectionner le numéro du menu auquel vous voulez accé
       der \n"+
18                          " 1) Tests Unitaires \n 2) Test à la main \n 3) Quitter
       ";
19     int reponse;
20     do{
21       System.out.println(menuPrincipal);
22       reponse = saisieUtilisateur.nextInt();
23       switch (reponse) {
24         case 1:
25           Test test = new Test();
26           test.test_unitaires();
27           break;
28         case 2:
29           menuchoixBDDDemo();
30           break;
31         case 3:
32           reponse = 5;
33           break;
34         default:
35           reponse = 4;
36       }
37     }while (reponse < 5);
38   }
39
40
41
42
43
44   public void menuchoixBDDDemo()throws ParseException{
45     String menuPrincipalDemo ="Sélectionner le numéro du menu auquel vous voulez
       accéder \n"+
```

```
46                          " 1) Générer une BDD vide \n 2) Générer une BDD remplie
       \n 3) Récupérer la BDD existante \n 4) Retour";
47      int reponse;
48      do{
49        this.init = new Init();
50        System.out.println(menuPrincipalDemo);
51        reponse = saisieUtilisateur.nextInt();
52        switch (reponse) {
53          case 1:
54            init.genererBDD();
55            menuchoixdeDemo();
56            break;
57          case 2:
58            init.genererBDD();
59            init.RemplirBDD();
60            menuchoixdeDemo();
61            break;
62          case 3:
63            init.remplirfromBDD();
64            menuchoixdeDemo();
65            break;
66          case 4:
67            reponse = 6;
68            break;
69          default:
70            reponse = 5;
71        }
72      }while (reponse < 6);
73    }
74
75    public void menuchoixdeDemo()throws ParseException{
76      String menuPrincipalDemo ="Sélectionner le numéro du menu auquel vous voulez
       accéder \n"+
77                          " 1) Démo parcours utilisateur \n 2) Démo minimale \n
       3) Retour";
78      int reponse;
79      do{
80        this.init = new Init();
81        System.out.println(menuPrincipalDemo);
82        reponse = saisieUtilisateur.nextInt();
83        switch (reponse) {
84          case 1:
85            Init init2 = new Init();
86            init2.remplirfromBDD();
87            Menunormal frame = new Menunormal(init2);
88            frame.setLayout(new GridLayout(0,1));
```

```
89              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
90              frame.pack();
91              frame.setSize(450, 900);
92              frame.setVisible(true);
93              break;
94            case 2:
95              menuPrincipalDemo();
96              break;
97            case 3:
98              reponse = 5;
99              break;
100           default:
101             reponse = 4;
102       }
103     }while (reponse < 5);
104   }
105
106   public void menuPrincipalDemo()throws ParseException{
107     String menuPrincipalDemo ="Sélectionner le numéro du menu auquel vous voulez
        accéder \n"+
108                               " 1) Produit \n 2) Vente \n 3) Utilisateur \n 4)
        Enchere \n 5) Retour";
109     int reponse;
110     do{
111       System.out.println(menuPrincipalDemo);
112       reponse = saisieUtilisateur.nextInt();
113       switch (reponse) {
114         case 1:
115           demoProduit();
116           break;
117         case 2:
118           demoVente();
119           break;
120         case 3:
121           demoUtilisateur();
122           break;
123         case 4:
124           demoEnchere();
125           break;
126         case 5:
127           reponse = 7;
128           break;
129         default:
130           reponse = 6;
131       }
132     }while (reponse < 7);
```

```
133     }
134
135     public void demoUtilisateur(){
136       String menuPrincipalDemo ="Sélectionner le numéro du menu auquel vous voulez
        accéder \n"+
137                                 " 1) Créer \n 2) Modifier \n 3) Supprimer \n 4)
        Afficher \n 5) Retour";
138       int reponse;
139       do{
140         System.out.println(menuPrincipalDemo);
141         reponse = saisieUtilisateur.nextInt();
142         switch (reponse) {
143           case 1:
144             init.creerUtilisateur();
145             break;
146           case 2:
147             init.modifierUtilisateur();
148             break;
149           case 3:
150             init.supprimerUtilisateur();
151             break;
152           case 4:
153             init.afficherUtilisateur();
154             break;
155           case 5:
156             reponse = 7;
157             break;
158           default:
159             reponse = 6;
160         }
161       }while (reponse < 7);
162     }
163
164     public void demoProduit(){
165       String menuPrincipalDemo ="Sélectionner le numéro du menu auquel vous voulez
        accéder \n"+
166                                 " 1) Créer \n 2) Modifier \n 3) Supprimer \n 4)
        Afficher \n 5) Retour";
167       int reponse;
168       do{
169         System.out.println(menuPrincipalDemo);
170         reponse = saisieUtilisateur.nextInt();
171         switch (reponse) {
172           case 1:
173             init.creerProduit();
174             break;
```

```
175           case 2:
176             init.modifierProduit();
177             break;
178           case 3:
179             init.supprimerProduit();
180             break;
181           case 4:
182             init.afficherProduit();
183             break;
184           case 5:
185             reponse = 7;
186             break;
187           default:
188             reponse = 6;
189         }
190       }while (reponse < 7);
191     }
192
193     public void demoVente()throws ParseException{
194       String menuPrincipalDemo ="Sélectionner le numéro du menu auquel vous voulez
          accéder \n"+
195                                 " 1) Créer \n 2) Modifier \n 3) Supprimer \n 4)
          Afficher \n 5) Retour";
196       int reponse;
197       do{
198         System.out.println(menuPrincipalDemo);
199         reponse = saisieUtilisateur.nextInt();
200         switch (reponse) {
201           case 1:
202             init.creerVente();
203             break;
204           case 2:
205             init.modifierVente();
206             break;
207           case 3:
208             init.supprimerVente();
209             break;
210           case 4:
211             init.afficherVente();
212             break;
213           case 5:
214             reponse = 7;
215             break;
216           default:
217             reponse = 6;
218         }
```

```
219       }while (reponse < 7);
220     }
221
222     public void demoEnchere(){
223       String menuPrincipalDemo ="Sélectionner le numéro du menu auquel vous voulez
          accéder \n"+
224                                 " 1) Créer \n 2) Modifier \n 3) Supprimer \n 4)
          Afficher \n 5) Retour";
225       int reponse;
226       do{
227         System.out.println(menuPrincipalDemo);
228         reponse = saisieUtilisateur.nextInt();
229         switch (reponse) {
230           case 1:
231             init.creerEnchere();
232             break;
233           case 2:
234             init.modifierEnchere();
235             break;
236           case 3:
237             init.supprimerEnchere();
238             break;
239           case 4:
240             init.afficherEnchere();
241             break;
242           case 5:
243             reponse = 7;
244             break;
245           default:
246             reponse = 6;
247         }
248       }while (reponse < 7);
249     }
250   }
```

java/DemoMinimale.java

```
1   import java.util.Calendar;
2   import java.util.Date;
3   import java.text.SimpleDateFormat;
4   import java.text.ParseException;
5
6   public class Enchere {
7
8     private int numEnchere;
```

```
9     private int numUtilisateur;
10    private int numVente;
11    private float prixachatEnchere;
12    private Calendar dateEnchere;
13    private int qteproduitEnchere;
14    private Bddaccess bddaccess;
15    private String table = "Enchere";
16
17    public Enchere(String numEnchere, String numUtilisateur, String numVente,
        String prixachatEnchere, String dateEnchere, String qteproduitEnchere)throws
        ParseException {
18      this.numEnchere = Integer.parseInt(numEnchere);
19      this.numUtilisateur = Integer.parseInt(numUtilisateur);
20      this.numVente = Integer.parseInt(numVente);
21      this.prixachatEnchere = Float.parseFloat(prixachatEnchere);
22      SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
23      this.dateEnchere = Calendar.getInstance();
24      this.dateEnchere.setTime(sdf.parse(dateEnchere));
25      this.qteproduitEnchere = Integer.parseInt(qteproduitEnchere);
26      this.bddaccess = new Bddaccess();
27    }
28
29    public String stringEnchere(){
30      String enchere = "Cette enchère numéro "+numEnchere+" a pour prix " + String.
        valueOf(prixachatEnchere) + "euro, elle a été faite à la date du " +
        getdateEnchere() + ", elle concerne "+qteproduitEnchere+" pièce(s) du produit
        ";
31      return enchere;
32    }
33
34    public String stringEncheremenu(){
35      String enchere = "elle a pour prix " + String.valueOf(prixachatEnchere) + "
        euro <br/>elle a été faite à la date du " + getdateEnchere() + ", <br/>elle
        concerne "+qteproduitEnchere+" pièce(s) du produit<br/><br/>";
36      return enchere;
37    }
38
39    public int getnumEnchere(){
40      return this.numEnchere;
41    }
42
43    public int getnumUtilisateur(){
44      return this.numUtilisateur;
45    }
46
47    public int getnumVente(){
```

```
48        return this.numVente;
49      }
50
51      public float getprixachatEnchere(){
52        return this.prixachatEnchere;
53      }
54
55      public String getdateEnchere(){
56        SimpleDateFormat format1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
57        String res = format1.format(this.dateEnchere.getTime());
58        return res;
59      }
60
61      public int getqteproduitEnchere(){
62        return this.qteproduitEnchere;
63      }
64
65      public void setprixachatEnchere(float prixachatEnchere){
66        updateEnchere(getnumEnchere(), getnumUtilisateur(), getnumVente(),
          prixachatEnchere, getdateEnchere(), getqteproduitEnchere());
67        this.prixachatEnchere = prixachatEnchere;
68      }
69
70      public void setdateEnchere(String dateEnchere)throws ParseException {
71        updateEnchere(getnumEnchere(), getnumUtilisateur(), getnumVente(),
          getprixachatEnchere(), dateEnchere, getqteproduitEnchere());
72        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
73        this.dateEnchere.setTime(sdf.parse(dateEnchere));
74      }
75
76      public void setqteproduitEnchere(int qteproduitEnchere){
77        updateEnchere(getnumEnchere(), getnumUtilisateur(), getnumVente(),
          getprixachatEnchere(), getdateEnchere(), qteproduitEnchere);
78        this.qteproduitEnchere = qteproduitEnchere;
79      }
80
81      public void updateEnchere(int numEnchere, int numUtilisateur, int numVente,
          float prixachatEnchere, String dateEnchere, int setqteproduitEnchere){
82        String[] update_value = {
83          "numEnchere",
84          String.valueOf(numEnchere),
85          "numUtilisateur",
86          String.valueOf(numUtilisateur),
87          "numVente",
88          String.valueOf(numVente),
89          "prixachatEnchere",
```

```
90          String.valueOf(prixachatEnchere),
91          "dateEnchere",
92          "TO_DATE('"+getdateEnchere()+"','YYYY-MM-DD HH24:MI:SS')",
93          "qteproduitEnchere",
94          String.valueOf(setqteproduitEnchere)
95        };
96        this.bddaccess.update(table, update_value);
97      }
98
99      public void deleteEnchere(){
100       this.bddaccess.delete(this.table, "numEnchere="+String.valueOf(this.
          numEnchere));
101     }
102
103     public void insertEnchere(){
104       String[] column_value = {
105         "numEnchere",
106         "numUtilisateur",
107         "numVente",
108         "prixachatEnchere",
109         "dateEnchere",
110         "qteproduitEnchere",
111         String.valueOf(numEnchere),
112         String.valueOf(numUtilisateur),
113         String.valueOf(numVente),
114         String.valueOf(prixachatEnchere),
115         "TO_DATE('"+getdateEnchere()+"','YYYY-MM-DD HH24:MI:SS')",
116         String.valueOf(qteproduitEnchere)
117       };
118       this.bddaccess.insert(this.table, column_value);
119     }
120 }
```

java/Enchere.java

```
1  import java.util.concurrent.ThreadLocalRandom;
2  import java.text.ParseException;
3  import java.util.ArrayList;
4  import java.util.Date;
5  import java.text.SimpleDateFormat;
6  import java.util.Calendar;
7  import java.util.Scanner;
8
9  public class Init {
10
```

```
11    private Bddaccess bddaccess;
12    private ArrayList<Utilisateur> utilisateurs;
13    private ArrayList<Produit> produits;
14    private ArrayList<Vente> ventes;
15    private ArrayList<Enchere> encheres;
16    private Scanner saisieUtilisateur;
17
18    public Init(){
19      this.bddaccess = new Bddaccess();
20      this.utilisateurs = new ArrayList<Utilisateur>();
21      this.produits = new ArrayList<Produit>();
22      this.ventes = new ArrayList<Vente>();
23      this.encheres = new ArrayList<Enchere>();
24      this.saisieUtilisateur = new Scanner(System.in);
25    }
26
27    public ArrayList<Utilisateur> getutilisateurs(){
28      return this.utilisateurs;
29    }
30    public ArrayList<Produit> getproduits(){
31      return this.produits;
32    }
33    public ArrayList<Vente> getventes(){
34      return this.ventes;
35    }
36    public ArrayList<Enchere> getencheres(){
37      return this.encheres;
38    }
39
40    public void genererBDD(){
41      String table = "Produit";
42      String option = " numProduit INTEGER NOT NULL,"
43      + " nomProduit varchar(30) NOT NULL,"
44      + " prixrevientProduit float NOT NULL,"
45      + " stockProduit INTEGER NOT NULL,"
46      + " primary key (numProduit)";
47      this.bddaccess.drop(table);
48      this.bddaccess.create(table, option);
49      table = "Utilisateur";
50      option = " numUtilisateur INTEGER,"
51      + " emailUtilisateur varchar(30) NOT NULL UNIQUE,"
52      + " nomUtilisateur varchar(30) NOT NULL,"
53      + " prenomUtilisateur varchar(30) NOT NULL,"
54      + " adresseUtilisateur varchar(30) NOT NULL,"
55      + " primary key (numUtilisateur)";
56      this.bddaccess.drop(table);
```

```
57      this.bddaccess.create(table, option);
58      table = "Vente";
59      option = " numVente INTEGER NOT NULL,"
60        + " numProduit INTEGER NOT NULL,"
61        + " prixVente float NOT NULL,"
62        + " montantVente number(1) not null check (montantVente in (1,0)),"
63        + " revocableVente number(1) not null check (revocableVente in (1,0)),"
64        + " dureelibreVente number(1) not null check (dureelibreVente in (1,0)),"
65        + " datefinVente DATE,"
66        + " encherirplusieurfoisVente number(1) not null check (
        encherirplusieurfoisVente in (1,0)),"
67        + " PRIMARY KEY (numVente),"
68        + " CONSTRAINT FK_produit FOREIGN KEY (numProduit) REFERENCES Produit(
        numProduit) ON DELETE CASCADE";
69      this.bddaccess.drop(table);
70      this.bddaccess.create(table, option);
71      table = "Enchere";
72      option = " numEnchere INTEGER NOT NULL,"
73        + " numUtilisateur INTEGER NOT NULL,"
74        + " numVente INTEGER NOT NULL,"
75        + " prixachatEnchere float NOT NULL,"
76        + " dateEnchere DATE,"
77        + " qteproduitEnchere INTEGER NOT NULL,"
78        + " PRIMARY KEY (numEnchere),"
79        + " CONSTRAINT FK_Utilisateur FOREIGN KEY (numUtilisateur) REFERENCES
        Utilisateur(numUtilisateur) ON DELETE CASCADE,"
80        + " CONSTRAINT FK_Vente FOREIGN KEY (numVente) REFERENCES Vente(numVente)
       ON DELETE CASCADE";
81      this.bddaccess.drop(table);
82      this.bddaccess.create(table, option);
83    }
84
85    public void RemplirBDD()throws ParseException{
86      genererBDD();
87      String[] nomproduits ={
88        "Petit stepper",
89        "Bateau",
90        "Guirlande lumineuse",
91        "Cage a hamster",
92        "Cable parrallele",
93        "Tapis",
94        "Lentilles de contact",
95        "Matelas gonflable",
96        "Tablette Wacom",
97        "Protection pour bras casse",
98        "Trotteur",
```

```
99          "Affiches"
100       };
101       String[] rue ={
102         "Rue du lise",
103         "Place du lise",
104         "Grande Rue",
105         "Rue du Moulin",
106         "Place de la Mairie",
107         "Rue du Château",
108         "Rue des Écoles",
109         "Rue de la Gare",
110         "Rue de la Mairie",
111         "Rue Principale",
112         "Rue du Stade",
113         "Rue de la Fontaine"
114       };
115       String[] prenom ={
116         "Gabriel",
117         "Leo",
118         "Raphael",
119         "Arthur",
120         "Louis",
121         "Lucas",
122         "Adam",
123         "Jules",
124         "Hugo",
125         "Mael",
126         "Liam",
127         "Noah"
128       };
129       String[] nom ={
130           "Imbert",
131         "Prevot",
132           "Toussaint",
133         "Cornu",
134           "Maillet",
135         "Lelievre",
136           "Bonneau",
137         "Flament",
138           "Tournier",
139         "Merlin",
140           "Salaun",
141         "Vial"
142       };
143       for(int i = 0; i<nomproduits.length; i++){
144         String randomprix = String.valueOf(ThreadLocalRandom.current().nextInt(20,
```

```
        1000));
145       String randomstock = String.valueOf(ThreadLocalRandom.current().nextInt(2,
        10));
146       Produit produit = new Produit(String.valueOf(i), nomproduits[i], randomprix
        , randomstock);
147       produit.insertProduit();
148       produits.add(produit);
149     }
150     for(int i = 0; i<prenom.length; i++){
151       String numrue = String.valueOf(ThreadLocalRandom.current().nextInt(1, 20));
152       String numUtilisateur = String.valueOf(i);
153       String emailUtilisateur = prenom[i] + "." + nom[i] + "@gmail.com";
154       String nomUtilisateur = nom[i];
155       String prenomUtilisateur = prenom[i];
156       String adresseUtilisateur = numrue + " " + rue[i];
157       Utilisateur utilisateur = new Utilisateur(numUtilisateur, emailUtilisateur,
         nomUtilisateur, prenomUtilisateur, adresseUtilisateur);
158       utilisateur.insertUtilisateur();
159       utilisateurs.add(utilisateur);
160       }
161     for(int i = 0; i<nomproduits.length/2; i++){
162       String numVente = String.valueOf(i);
163       String numProduit = String.valueOf(ThreadLocalRandom.current().nextInt(0,
        nomproduits.length-1));
164       Produit prodaux = produits.get(Integer.parseInt(numProduit));
165       String prixVente = String.valueOf(prodaux.getprixrevientProduit()+
        ThreadLocalRandom.current().nextInt(20, 1000));
166       String montantVente = String.valueOf(ThreadLocalRandom.current().nextInt(0,
         1));
167       String revocableVente = String.valueOf(ThreadLocalRandom.current().nextInt
        (0, 1));
168       String dureelibreVente = String.valueOf(ThreadLocalRandom.current().nextInt
        (0, 1));
169       Date date = new Date();
170       Calendar cal = Calendar.getInstance();
171       cal.setTime(date);
172       cal.add(Calendar.MINUTE, 10);
173       date = cal.getTime();
174       SimpleDateFormat format1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
175       String datefinVente = format1.format(cal.getTime());
176       String encherirplusieurfoisVente = String.valueOf(ThreadLocalRandom.current
        ().nextInt(0, 1));
177       Vente vente = new Vente(numVente, numProduit, prixVente, montantVente,
        revocableVente, dureelibreVente, datefinVente, encherirplusieurfoisVente);
178       vente.insertVente();
179       ventes.add(vente);
```

```
180       }
181       int j =0;
182     for(int i = 0; i<nomproduits.length/2; i++){
183       for(int y = 0; y<5; y++){
184         String numEnchere = String.valueOf(j);
185         String numUtilisateur = String.valueOf(ThreadLocalRandom.current().nextInt
        (0, utilisateurs.size()-1));
186         String numVente = String.valueOf(i);
187         String prixachatEnchere = String.valueOf(ventes.get(i).getprixVente() + j
        *10);
188         Date date = new Date();
189         Calendar cal = Calendar.getInstance();
190         cal.setTime(date);
191         cal.add(Calendar.MINUTE, y-10);
192         date = cal.getTime();
193         SimpleDateFormat format1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
194         String dateEnchere = format1.format(cal.getTime());
195         String qteproduitEnchere = String.valueOf(ThreadLocalRandom.current().
        nextInt(1 , produits.get(i).getstockProduit()));
196         j++;
197         Enchere enchere = new Enchere( numEnchere,  numUtilisateur,  numVente,
        prixachatEnchere,  dateEnchere,  qteproduitEnchere);
198         enchere.insertEnchere();
199         encheres.add(enchere);
200       }
201     }
202   }
203
204     public void remplirfromBDD()throws ParseException{
205       remplirproduit();
206       remplirutilisateur();
207       remplirvente();
208       remplirenchere();
209     }
210
211     public void remplirproduit()throws ParseException{
212       produits.clear();
213       String table = "Produit";
214       String[] resultSet = this.bddaccess.select(table);
215       int size = resultSet.length;
216       for(int i = 0; i<size; i+=4){
217         Produit produit = new Produit(resultSet[i], resultSet[i+1], resultSet[i+2],
         resultSet[i+3]);
218         produits.add(produit);
219       }
220     }
```

```
221
222   public void remplirutilisateur()throws ParseException{
223     utilisateurs.clear();
224     String table = "Utilisateur";
225     String[] resultSet = this.bddaccess.select(table);
226     int size = resultSet.length;
227     for(int i = 0; i<size; i+=5){
228       Utilisateur utilisateur = new Utilisateur(resultSet[i], resultSet[i+1],
      resultSet[i+2], resultSet[i+3], resultSet[i+4]);
229       utilisateurs.add(utilisateur);
230     }
231   }
232
233   public void remplirvente()throws ParseException{
234     ventes.clear();
235     for (int j=0; j<produits.size(); j++){
236       produits.get(j).getventes().clear();
237     }
238     String table = "Vente";
239     String[] resultSet = this.bddaccess.select(table);
240     int size = resultSet.length;
241     for(int i = 0; i<size; i+=8){
242       Vente vente = new Vente(resultSet[i], resultSet[i+1], resultSet[i+2],
      resultSet[i+3], resultSet[i+4], resultSet[i+5], resultSet[i+6], resultSet[i
      +7]);
243       ventes.add(vente);
244       for (int j=0; j<produits.size(); j++){
245         if(vente.getnumProduit() == produits.get(j).getnumProduit()){
246           produits.get(j).addVente(vente);
247         }
248       }
249     }
250   }
251
252   public void remplirenchere()throws ParseException{
253     encheres.clear();
254     for (int j=0; j<ventes.size(); j++){
255       ventes.get(j).getencheres().clear();
256     }
257     for (int j=0; j<utilisateurs.size(); j++){
258       utilisateurs.get(j).getencheres().clear();
259     }
260     String table = "Enchere";
261     String[] resultSet = this.bddaccess.select(table);
262     int size = resultSet.length;
263     for(int i = 0; i<size; i+=6){
```

```
264         Enchere enchere = new Enchere(resultSet[i], resultSet[i+1], resultSet[i+2],
            resultSet[i+3], resultSet[i+4], resultSet[i+5]);
265         encheres.add(enchere);
266         for (int j=0; j<ventes.size(); j++){
267           if(enchere.getnumVente() == ventes.get(j).getnumVente()){
268             ventes.get(j).addEnchere(enchere);
269           }
270         }
271         for (int j=0; j<utilisateurs.size(); j++){
272           if(enchere.getnumUtilisateur() == utilisateurs.get(j).getnumUtilisateur()
            ){
273             utilisateurs.get(j).addEnchere(enchere);
274           }
275         }
276       }
277     }
278
279
280
281     public void creerUtilisateur(){
282       System.out.println("Rentrer l'email de utilisateur");
283       String emailUtilisateur = saisieUtilisateur.nextLine();
284       System.out.println("Rentrer le nom de utilisateur");
285       String nomUtilisateur = saisieUtilisateur.nextLine();
286       System.out.println("Rentrer le prénom de utilisateur");
287       String prenomUtilisateur = saisieUtilisateur.nextLine();
288       System.out.println("Rentrer l'adresse de utilisateur");
289       String adresseUtilisateur = saisieUtilisateur.nextLine();
290       creerUtilisateurBDD( emailUtilisateur, nomUtilisateur, prenomUtilisateur,
          adresseUtilisateur);
291     }
292
293     public String creerUtilisateurBDD(String emailUtilisateur, String
          nomUtilisateur, String prenomUtilisateur, String adresseUtilisateur){
294       String numUtilisateur;
295       if(produits.size()==0){
296         numUtilisateur = "0";
297       }else{
298         numUtilisateur = String.valueOf(utilisateurs.get(utilisateurs.size()-1).
          getnumUtilisateur()+1);
299       }
300       Utilisateur utilisateur = new Utilisateur(numUtilisateur, emailUtilisateur,
          nomUtilisateur, prenomUtilisateur,
301       adresseUtilisateur);
302       String res = verifUtilisateur(emailUtilisateur);
303       if (res.equals("")){
```

```
304         utilisateur.insertUtilisateur();
305         utilisateurs.add(utilisateur);
306       }
307       return res;
308     }
309
310     public void modifierUtilisateur(){
311       System.out.println("Rentrer le numéro de l'utilisateur");
312       int num = saisieUtilisateur.nextInt();
313       int numUtilisateur = chercheUtilisateur(num);
314       System.out.println("Rentrer l'email de utilisateur");
315       saisieUtilisateur.nextLine();
316       String emailUtilisateur = saisieUtilisateur.nextLine();
317       System.out.println("Rentrer le nom de utilisateur");
318       String nomUtilisateur = saisieUtilisateur.nextLine();
319       System.out.println("Rentrer le prénom de utilisateur");
320       String prenomUtilisateur = saisieUtilisateur.nextLine();
321       System.out.println("Rentrer l'adresse de utilisateur");
322       String adresseUtilisateur = saisieUtilisateur.nextLine();
323       utilisateurs.get(numUtilisateur).setemailUtilisateur(emailUtilisateur);
324       utilisateurs.get(numUtilisateur).setnomUtilisateur(nomUtilisateur);
325       utilisateurs.get(numUtilisateur).setprenomUtilisateur(prenomUtilisateur);
326       utilisateurs.get(numUtilisateur).setadresseUtilisateur(adresseUtilisateur);
327     }
328
329     public int chercheUtilisateur(int num){
330       int res = 0;
331       for (int i=0;i<utilisateurs.size() ;i++ ) {
332         if(num == utilisateurs.get(i).getnumUtilisateur())res=i;
333       }
334       return res;
335     }
336
337
338     public void supprimerUtilisateur(){
339       System.out.println("Rentrer le numéro de l'utilisateur");
340       int num = saisieUtilisateur.nextInt();
341       int numUtilisateur = chercheUtilisateur(num);
342       utilisateurs.get(numUtilisateur).deleteUtilisateur();
343       utilisateurs.remove(utilisateurs.get(numUtilisateur));
344     }
345
346     public void afficherUtilisateur(){
347       utilisateurs.forEach((n) -> System.out.println(n.stringUtilisateur()));
348     }
349
```

```
350     public void creerProduit(){
351       System.out.println("Rentrer le nom du produit");
352       String nomProduit = saisieUtilisateur.nextLine();
353       System.out.println("Rentrer le prix de revient du produit");
354       String prixrevientProduit = saisieUtilisateur.nextLine();
355       System.out.println("Rentrer le stock du produit");
356       String stockProduit = saisieUtilisateur.nextLine();
357       creerProduitBDD(nomProduit, prixrevientProduit, stockProduit);
358     }
359
360     public void creerProduitBDD(String nomProduit, String prixrevientProduit,
        String stockProduit){
361       String numProduit;
362       if(produits.size()==0){
363         numProduit = "0";
364       }else{
365         numProduit = String.valueOf(produits.get(produits.size()-1).getnumProduit()
        +1);
366       }
367       Produit produit = new Produit(numProduit, nomProduit, prixrevientProduit,
        stockProduit);
368       produit.insertProduit();
369       produits.add(produit);
370     }
371
372     public void modifierProduit(){
373       System.out.println("Rentrer le numéro du produit");
374       int num = saisieUtilisateur.nextInt();
375       int numProduit = chercheProduit(num);
376       System.out.println("Rentrer le nom du produit");
377       saisieUtilisateur.nextLine();
378       String nomProduit = saisieUtilisateur.nextLine();
379       System.out.println("Rentrer le prix de revient du produit");
380       String prixrevientProduit = saisieUtilisateur.nextLine();
381       System.out.println("Rentrer le stock du produit");
382       String stockProduit = saisieUtilisateur.nextLine();
383       produits.get(numProduit).setnomProduit(nomProduit);
384       produits.get(numProduit).setprixrevientProduit(Float.parseFloat(
        prixrevientProduit));
385       produits.get(numProduit).setstockProduit(Integer.parseInt(stockProduit));
386     }
387
388     public int chercheProduit(int num){
389       int res = 0;
390       for (int i=0;i<produits.size() ;i++ ) {
391         if(num == produits.get(i).getnumProduit())res=i;
```

```
392       }
393       return res;
394     }
395 
396     public void supprimerProduit(){
397       System.out.println("Rentrer le numéro du produit");
398       int num = saisieUtilisateur.nextInt();
399       int numProduit = chercheProduit(num);
400       produits.get(numProduit).deleteProduit();
401       produits.remove(produits.get(numProduit));
402     }
403 
404     public void afficherProduit(){
405       produits.forEach((n) -> System.out.println(n.stringProduit()));
406     }
407 
408     public void creerVente()throws ParseException {
409       System.out.println("Rentrer le numéro du produit auquel correspond la vente")
          ;
410       afficherProduit();
411       int numProduit = saisieUtilisateur.nextInt();
412       System.out.println("Quel est le prix de départ de ce produit ?");
413       saisieUtilisateur.nextLine();
414       String prixVente = saisieUtilisateur.nextLine();
415       System.out.println("Cette vente est elle montante ? O/N");
416       String montantVente = (saisieUtilisateur.nextLine().equals("O"))?"1":"0";
417       System.out.println("Cette vente peut elle être à perte ? O/N");
418       String revocableVente = (saisieUtilisateur.nextLine().equals("O"))?"1":"0";
419       System.out.println("Cette vente est elle à durée libre ? O/N");
420       String dureelibreVente = (saisieUtilisateur.nextLine().equals("O"))?"1":"0";
421       System.out.println("Un utilisateur peut il enchérir plusieurs fois sur cette
          vente ? O/N");
422       String encherirplusieurfoisVente = (saisieUtilisateur.nextLine().equals("O"))
          ?"1":"0";
423       String datefinVente = "2020-06-06 12:12:12";
424       if(dureelibreVente.equals("0")){
425         System.out.println("A quelle date fini la vente ? format yyyy-MM-dd HH:mm:
          ss");
426         datefinVente = saisieUtilisateur.nextLine();
427       }
428       creerVenteBDD(String.valueOf(numProduit), prixVente, montantVente,
          revocableVente, dureelibreVente, datefinVente, encherirplusieurfoisVente);
429     }
430 
431   public void creerVenteBDD(String numProduit, String prixVente, String
        montantVente, String revocableVente, String dureelibreVente, String
```

```
      datefinVente, String encherirplusieurfoisVente){
432     String numVente;
433     if(ventes.size()==0){
434       numVente = "0";
435     }else{
436       numVente = String.valueOf(ventes.get(ventes.size()-1).getnumVente()+1);
437     }
438     try{
439       Vente vente = new Vente(numVente, numProduit, prixVente, montantVente,
      revocableVente, dureelibreVente, datefinVente, encherirplusieurfoisVente);
440       vente.insertVente();
441       ventes.add(vente);
442     } catch(Exception exep) {}
443   }
444
445   public void modifierVente()throws ParseException {
446     System.out.println("Rentrer le numéro de la vente");
447     int num = saisieUtilisateur.nextInt();
448     int numVente = chercheVente(num);
449     System.out.println("Quel est le prix de départ de ce produit ?");
450     String prixVente = saisieUtilisateur.nextLine();
451     System.out.println("Cette vente est elle montante ? O/N");
452     String montantVente = (saisieUtilisateur.nextLine().equals("O"))?"1":"0";
453     System.out.println("Cette vente peut elle être à perte ? O/N");
454     String revocableVente = (saisieUtilisateur.nextLine().equals("O"))?"1":"0";
455     System.out.println("Cette vente est elle à durée libre ? O/N");
456     String dureelibreVente = (saisieUtilisateur.nextLine().equals("O"))?"1":"0";
457     System.out.println("Un utilisateur peut il enchérir plusieurs fois sur cette
      vente ? O/N");
458     String encherirplusieurfoisVente = (saisieUtilisateur.nextLine().equals("O"))
      ?"1":"0";
459     String datefinVente = "2020-06-06 12:12:12";
460     if(dureelibreVente.equals("0")){
461       System.out.println("A quelle date fini la vente ? format yyyy-MM-dd HH:mm:
      ss");
462       datefinVente = saisieUtilisateur.nextLine();
463     }
464     ventes.get(numVente).setprixVente(Float.parseFloat(prixVente));
465     ventes.get(numVente).setmontantVente(!montantVente.equals("0"));
466     ventes.get(numVente).setrevocableVente(!revocableVente.equals("0"));
467     ventes.get(numVente).setdureelibreVente(!dureelibreVente.equals("0"));
468     ventes.get(numVente).setdatefinVente(datefinVente);
469     ventes.get(numVente).setencherirplusieurfoisVente( !encherirplusieurfoisVente
      .equals("0"));
470   }
471
```

```
472   public int chercheVente(int num){
473     int res = 0;
474     for (int i=0;i<ventes.size() ;i++ ) {
475       if(num == ventes.get(i).getnumVente())res=i;
476     }
477     return res;
478   }
479
480   public void supprimerVente(){
481     System.out.println("Rentrer le numéro de la vente");
482     int num = saisieUtilisateur.nextInt();
483     int numVente = chercheVente(num);
484     ventes.get(numVente).deleteVente();
485     ventes.remove(ventes.get(numVente));
486   }
487
488   public void afficherVente(){
489     ventes.forEach((n) -> System.out.println(n.stringVente()));
490   }
491
492   public void creerEnchere(){
493   }
494
495   public void modifierEnchere(){
496   }
497
498   public int chercheEnchere(int num){
499     int res = 0;
500     for (int i=0;i<encheres.size() ;i++ ) {
501       if(num == encheres.get(i).getnumEnchere())res=i;
502     }
503     return res;
504   }
505
506   public void supprimerEnchere(){
507     System.out.println("Rentrer le numéro du enchere");
508     int num = saisieUtilisateur.nextInt();
509     int numEnchere = chercheEnchere(num);
510     encheres.get(numEnchere).deleteEnchere();
511     encheres.remove(encheres.get(numEnchere));
512   }
513
514   public void afficherEnchere(){
515     encheres.forEach((n) -> System.out.println(n.stringEnchere()));
516   }
517
```

```
518   public String verifUtilisateur(String emailUtilisateur){
519       String res = "";
520       for (int i=0; i<utilisateurs.size(); i++) {
521         if(utilisateurs.get(i).getemailUtilisateur().equals(emailUtilisateur))res
      ="!---------- Cet email existe déjà ----------!";
522       }
523       return res;
524   }
525
526   public int verifUtilisateurMenu(String emailUtilisateur){
527       int res = -1;
528       for (int i=0; i<utilisateurs.size(); i++) {
529         if(utilisateurs.get(i).getemailUtilisateur().equals(emailUtilisateur))res
      =i;
530       }
531       return res;
532   }
533
534 }
```

java/Init.java

```
1  import java.text.ParseException;
2  import java.util.ArrayList;
3
4  public class Test {
5    public static final String ANSI_RESET = "\u001B[0m";
6    public static final String ANSI_RED = "\u001B[31m";
7    public static final String ANSI_GREEN = "\u001B[32m";
8    public static final String ANSI_BLUE = "\u001B[34m";
9
10   private Bddaccess bddaccess;
11
12   public Test(){
13     this.bddaccess = new Bddaccess();
14   }
15
16   public void test_unitaires()throws ParseException {
17     this.test_complet_Bddaccess();
18     this.test_complet_Produit();
19     this.test_complet_Utilisateur();
20     this.test_complet_Vente();
21     this.test_complet_Enchere();
22     this.test_complet_Init();
23   }
```

```
24
25    private void test_complet_Init()throws ParseException {
26      System.out.println(ANSI_BLUE+"Start Init unit test"+ANSI_RESET);
27      Init init = new Init();
28      init.genererBDD();
29      init.RemplirBDD();
30      ArrayList<Utilisateur> utilisateurs = init.getutilisateurs();
31      ArrayList<Produit> produits = init.getproduits();
32      ArrayList<Vente> ventes = init.getventes();
33      ArrayList<Enchere> encheres = init.getencheres();
34      if(utilisateurs.size() != 0 && produits.size() != 0 && ventes.size() != 0 &&
        encheres.size() != 0){
35        System.out.println(ANSI_GREEN+"Init() OK"+ANSI_RESET);
36        System.out.println(ANSI_GREEN+"Init.getutilisateurs OK"+ANSI_RESET);
37        System.out.println(ANSI_GREEN+"Init.getproduits OK"+ANSI_RESET);
38        System.out.println(ANSI_GREEN+"Init.getventes OK"+ANSI_RESET);
39        System.out.println(ANSI_GREEN+"Init.getencheres OK"+ANSI_RESET);
40        System.out.println(ANSI_GREEN+"Init.genererBDD OK"+ANSI_RESET);
41        System.out.println(ANSI_GREEN+"Init.RemplirBDD OK"+ANSI_RESET);
42      }else {
43        System.out.println(ANSI_GREEN+"Init() Fail"+ANSI_RESET);
44        System.out.println(ANSI_GREEN+"Init.getutilisateurs Fail"+ANSI_RESET);
45        System.out.println(ANSI_GREEN+"Init.getproduits Fail"+ANSI_RESET);
46        System.out.println(ANSI_GREEN+"Init.getventes Fail"+ANSI_RESET);
47        System.out.println(ANSI_GREEN+"Init.getencheres Fail"+ANSI_RESET);
48        System.out.println(ANSI_GREEN+"Init.genererBDD Fail"+ANSI_RESET);
49        System.out.println(ANSI_GREEN+"Init.RemplirBDD Fail"+ANSI_RESET);
50      }
51      init = new Init();
52      init.remplirfromBDD();
53      utilisateurs = init.getutilisateurs();
54      produits = init.getproduits();
55      ventes = init.getventes();
56      encheres = init.getencheres();
57      if(utilisateurs.size() != 0 && produits.size() != 0 && ventes.size() != 0 &&
        encheres.size() != 0){
58        System.out.println(ANSI_GREEN+"Init.remplirfromBDD OK"+ANSI_RESET);
59      }else{
60        System.out.println(ANSI_GREEN+"Init.remplirfromBDD Fail"+ANSI_RESET);
61      }
62
63      System.out.println();
64    }
65
66    private void test_complet_Produit(){
67      String table = "Produit";
```

```
68      System.out.println(ANSI_BLUE+"Start "+table+" unit test"+ANSI_RESET);
69      String option = " numProduit INTEGER,"
70      + " nomProduit varchar(30) NOT NULL,"
71      + " prixrevientProduit INTEGER NOT NULL,"
72      + " stockProduit INTEGER,"
73      + " primary key (numProduit)";
74      this.bddaccess.drop(table);
75      this.bddaccess.create(table, option);
76      Produit produit = new Produit("0", "Chocolat", "50", "20");
77      produit.insertProduit();
78      if (produit.getnumProduit() == 0) {
79        System.out.println(ANSI_GREEN+"Produit() OK"+ANSI_RESET);
80        System.out.println(ANSI_GREEN+"Produit.getnumProduit OK"+ANSI_RESET);
81      }else {
82        System.out.println(ANSI_RED+"Produit() Fail"+ANSI_RESET);
83        System.out.println(ANSI_RED+"Produit.getnumProduit Fail"+ANSI_RESET);
84      }
85      if (produit.getnomProduit().equals("Chocolat")) {
86        System.out.println(ANSI_GREEN+"Produit.getnomProduit OK"+ANSI_RESET);
87      }else {
88        System.out.println(ANSI_RED+"Produit.getnomProduit Fail"+ANSI_RESET);
89      }
90      if (produit.getprixrevientProduit() == 50) {
91        System.out.println(ANSI_GREEN+"Produit.getprixrevientProduit OK"+ANSI_RESET
        );
92      }else {
93        System.out.println(ANSI_RED+"Produit.getprixrevientProduit Fail"+ANSI_RESET
        );
94      }
95      if (produit.getstockProduit() == 20) {
96        System.out.println(ANSI_GREEN+"Produit.getstockProduit OK"+ANSI_RESET);
97      }else {
98        System.out.println(ANSI_RED+"Produit.getstockProduit Fail"+ANSI_RESET);
99      }
100     produit.setnomProduit("Chocopops");
101     if (produit.getnomProduit().equals("Chocopops")) {
102       System.out.println(ANSI_GREEN+"Produit.setnomProduit OK"+ANSI_RESET);
103     }else {
104       System.out.println(ANSI_RED+"Produit.setnomProduit Fail"+ANSI_RESET);
105     }
106     produit.setprixrevientProduit(40);
107     if (produit.getprixrevientProduit() == 40) {
108       System.out.println(ANSI_GREEN+"Produit.setprixrevientProduit OK"+ANSI_RESET
        );
109     }else {
110       System.out.println(ANSI_RED+"Produit.setprixrevientProduit Fail"+ANSI_RESET
```

```
      );
111   }
112   produit.setstockProduit(30);
113   if (produit.getstockProduit() == 30) {
114     System.out.println(ANSI_GREEN+"Produit.setstockProduit OK"+ANSI_RESET);
115   }else {
116     System.out.println(ANSI_RED+"Produit.setstockProduit Fail"+ANSI_RESET);
117   }
118   String[] resultSet = bddaccess.select(table);
119   String test = "";
120   for (int i=0; i<resultSet.length;i++){
121     test+=resultSet[i];
122   }
123   if(test.equals("0Chocopops4030")){
124     System.out.println(ANSI_GREEN+"Produit.updateProduit OK"+ANSI_RESET);
125     System.out.println(ANSI_GREEN+"Produit.insertProduit OK"+ANSI_RESET);
126   }else {
127     System.out.println(ANSI_RED+"Produit.updateProduit Fail"+ANSI_RESET);
128     System.out.println(ANSI_RED+"Produit.insertProduit Fail"+ANSI_RESET);
129   }
130   produit.deleteProduit();
131   resultSet = bddaccess.select(table);
132   test = "";
133   for (int i=0; i<resultSet.length;i++){
134     test+=resultSet[i];
135   }
136   if(test.equals("")){;
137     System.out.println(ANSI_GREEN+"Produit.deleteProduit OK"+ANSI_RESET);
138   }else {
139     System.out.println(ANSI_RED+"Produit.deleteProduit Fail"+ANSI_RESET);
140   }
141   System.out.println();
142 }
143
144 private void test_complet_Utilisateur(){
145   String table = "Utilisateur";
146   System.out.println(ANSI_BLUE+"Start "+table+" unit test"+ANSI_RESET);
147   String option = " numUtilisateur INTEGER,"
148   + " emailUtilisateur varchar(30) NOT NULL UNIQUE,"
149   + " nomUtilisateur varchar(30) NOT NULL,"
150   + " prenomUtilisateur varchar(30) NOT NULL,"
151   + " adresseUtilisateur varchar(30) NOT NULL,"
152   + " primary key (numUtilisateur)";
153   this.bddaccess.drop(table);
154   this.bddaccess.create(table, option);
155   Utilisateur utilisateur = new Utilisateur("0", "antoine.malo@toto.fr", "malo"
```

```
        , "antoine", "1 rue du chanvre");
156     utilisateur.insertUtilisateur();
157     if (utilisateur.getnumUtilisateur() == 0) {
158       System.out.println(ANSI_GREEN+"Utilisateur() OK"+ANSI_RESET);
159       System.out.println(ANSI_GREEN+"Utilisateur.getnumUtilisateur OK"+ANSI_RESET
        );
160     }else {
161       System.out.println(ANSI_RED+"Utilisateur() Fail"+ANSI_RESET);
162       System.out.println(ANSI_RED+"Utilisateur.getnumUtilisateur Fail"+ANSI_RESET
        );
163     }
164     if (utilisateur.getemailUtilisateur().equals("antoine.malo@toto.fr")) {
165       System.out.println(ANSI_GREEN+"Utilisateur.getemailUtilisateur OK"+
        ANSI_RESET);
166     }else {
167       System.out.println(ANSI_RED+"Utilisateur.getemailUtilisateur Fail"+
        ANSI_RESET);
168     }
169     if (utilisateur.getnomUtilisateur().equals("malo")) {
170       System.out.println(ANSI_GREEN+"Utilisateur.getnomUtilisateur OK"+ANSI_RESET
        );
171     }else {
172       System.out.println(ANSI_RED+"Utilisateur.getnomUtilisateur Fail"+ANSI_RESET
        );
173     }
174     if (utilisateur.getprenomUtilisateur().equals("antoine")) {
175       System.out.println(ANSI_GREEN+"Utilisateur.getprenomUtilisateur OK"+
        ANSI_RESET);
176     }else {
177       System.out.println(ANSI_RED+"Utilisateur.getprenomUtilisateur Fail"+
        ANSI_RESET);
178     }
179     if (utilisateur.getadresseUtilisateur().equals("1 rue du chanvre")) {
180       System.out.println(ANSI_GREEN+"Utilisateur.getadresseUtilisateur OK"+
        ANSI_RESET);
181     }else {
182       System.out.println(ANSI_RED+"Utilisateur.getadresseUtilisateur Fail"+
        ANSI_RESET);
183     }
184     utilisateur.setemailUtilisateur("antoine2.malo@toto.fr");
185     if (utilisateur.getemailUtilisateur().equals("antoine2.malo@toto.fr")) {
186       System.out.println(ANSI_GREEN+"Utilisateur.setemailUtilisateur OK"+
        ANSI_RESET);
187     }else {
188       System.out.println(ANSI_RED+"Utilisateur.setemailUtilisateur Fail"+
        ANSI_RESET);
```

```
189       }
190       utilisateur.setnomUtilisateur("Malo");
191       if (utilisateur.getnomUtilisateur().equals("Malo")) {
192         System.out.println(ANSI_GREEN+"Utilisateur.setnomutilisateur OK"+ANSI_RESET
          );
193       }else {
194         System.out.println(ANSI_RED+"Utilisateur.setnomutilisateur Fail"+ANSI_RESET
          );
195       }
196       utilisateur.setprenomUtilisateur("Antoine");
197       if (utilisateur.getprenomUtilisateur().equals("Antoine")) {
198         System.out.println(ANSI_GREEN+"Utilisateur.setprenomutilisateur OK"+
          ANSI_RESET);
199       }else {
200         System.out.println(ANSI_RED+"Utilisateur.setprenomutilisateur Fail"+
          ANSI_RESET);
201       }
202       utilisateur.setadresseUtilisateur("2 rue du chanvre");
203       if (utilisateur.getadresseUtilisateur().equals("2 rue du chanvre")) {
204         System.out.println(ANSI_GREEN+"Utilisateur.setadresseUtilisateur OK"+
          ANSI_RESET);
205       }else {
206         System.out.println(ANSI_RED+"Utilisateur.setadresseUtilisateur Fail"+
          ANSI_RESET);
207       }
208       String[] resultSet = bddaccess.select(table);
209       String test = "";
210       for (int i=0; i<resultSet.length;i++){
211         test+=resultSet[i];
212       }
213       if(test.equals("0antoine2.malo@toto.frMaloAntoine2 rue du chanvre")){
214         System.out.println(ANSI_GREEN+"Utilisateur.insertUtilisateur OK"+ANSI_RESET
          );
215         System.out.println(ANSI_GREEN+"Utilisateur.updateUtilisateur OK"+ANSI_RESET
          );
216       }else {
217         System.out.println(ANSI_RED+"Utilisateur.insertUtilisateur Fail"+ANSI_RESET
          );
218         System.out.println(ANSI_RED+"Utilisateur.updateUtilisateur Fail"+ANSI_RESET
          );
219       }
220       utilisateur.deleteUtilisateur();
221       resultSet = bddaccess.select(table);
222       test = "";
223       for (int i=0; i<resultSet.length;i++){
224         test+=resultSet[i];
```

```
225       }
226       if(test.equals("")){
227         System.out.println(ANSI_GREEN+"Utilisateur.deleteUtilisateur OK"+ANSI_RESET
          );
228       }else {
229         System.out.println(ANSI_RED+"Utilisateur.deleteUtilisateur Fail"+ANSI_RESET
          );
230       }
231       System.out.println();
232     }
233
234     private void test_complet_Vente()throws ParseException  {
235       String table = "Vente";
236       System.out.println(ANSI_BLUE+"Start "+table+" unit test"+ANSI_RESET);
237       String option = " numVente INTEGER NOT NULL,"
238         + " numProduit INTEGER NOT NULL,"
239         + " prixVente float NOT NULL,"
240         + " montantVente number(1) not null check (montantVente in (1,0)),"
241         + " revocableVente number(1) not null check (revocableVente in (1,0)),"
242         + " dureelibreVente number(1) not null check (dureelibreVente in (1,0)),"
243         + " datefinVente DATE,"
244         + " encherirplusieurfoisVente number(1) not null check (
          encherirplusieurfoisVente in (1,0)),"
245         + " PRIMARY KEY (numVente),"
246         + " CONSTRAINT FK_produit FOREIGN KEY (numProduit) REFERENCES Produit(
          numProduit)";
247       this.bddaccess.drop(table);
248       this.bddaccess.create(table, option);
249       Produit produit = new Produit("0", "Chocolat", "50", "20");
250       produit.insertProduit();
251       Vente vente = new Vente("0", "0", "20.20", "1", "1", "1", "2021-04-01
          00:00:00","1");
252       vente.insertVente();
253       if (vente.getnumVente() == 0) {
254         System.out.println(ANSI_GREEN+"Vente() OK"+ANSI_RESET);
255         System.out.println(ANSI_GREEN+"Vente.getnumVente OK"+ANSI_RESET);
256       }else {
257         System.out.println(ANSI_RED+"Vente() Fail"+ANSI_RESET);
258         System.out.println(ANSI_RED+"Vente.getnumVente Fail"+ANSI_RESET);
259       }
260       if (vente.getnumProduit() == 0) {
261         System.out.println(ANSI_GREEN+"Vente.getnumProduit OK"+ANSI_RESET);
262       }else {
263         System.out.println(ANSI_RED+"Vente.getnumProduit Fail"+ANSI_RESET);
264       }
265       if (vente.getprixVente() == (float)20.2) {
```

```
266         System.out.println(ANSI_GREEN+"Vente.getprixVente OK"+ANSI_RESET);
267       }else {
268         System.out.println(ANSI_RED+"Vente.getprixVente Fail"+ANSI_RESET);
269       }
270       if (vente.getmontantVente() == true) {
271         System.out.println(ANSI_GREEN+"Vente.getmontantVente OK"+ANSI_RESET);
272       }else {
273         System.out.println(ANSI_RED+"Vente.getmontantVente Fail"+ANSI_RESET);
274       }
275       if (vente.getrevocableVente() == true) {
276         System.out.println(ANSI_GREEN+"Vente.getrevocableVente OK"+ANSI_RESET);
277       }else {
278         System.out.println(ANSI_RED+"Vente.getrevocableVente Fail"+ANSI_RESET);
279       }
280       if (vente.getdureelibreVente() == true) {
281         System.out.println(ANSI_GREEN+"Vente.getdureelibreVente OK"+ANSI_RESET);
282       }else {
283         System.out.println(ANSI_RED+"Vente.getdureelibreVente Fail"+ANSI_RESET);
284       }
285       if (vente.getdatefinVente().equals("2021-04-01 00:00:00")) {
286         System.out.println(ANSI_GREEN+"Vente.getdatefinVente OK"+ANSI_RESET);
287       }else {
288         System.out.println(ANSI_RED+"Vente.getdatefinVente Fail"+ANSI_RESET);
289       }
290       if (vente.getencherirplusieurfoisVente() == true) {
291         System.out.println(ANSI_GREEN+"Vente.getencherirplusieurfoisVente OK"+
      ANSI_RESET);
292       }else {
293         System.out.println(ANSI_RED+"Vente.getencherirplusieurfoisVente Fail"+
      ANSI_RESET);
294       }
295       vente.setprixVente((float)40.44);
296       if (vente.getprixVente()==(float)40.44) {
297         System.out.println(ANSI_GREEN+"Vente.setemailUtilisateur OK"+ANSI_RESET);
298       }else {
299         System.out.println(ANSI_RED+"Vente.setemailUtilisateur Fail"+ANSI_RESET);
300       }
301       vente.setmontantVente(false);
302       if (vente.getmontantVente()==false) {
303         System.out.println(ANSI_GREEN+"Vente.setmontantVente OK"+ANSI_RESET);
304       }else {
305         System.out.println(ANSI_RED+"Vente.setmontantVente Fail"+ANSI_RESET);
306       }
307       vente.setrevocableVente(false);
308       if (vente.getrevocableVente()==false) {
309         System.out.println(ANSI_GREEN+"Vente.setrevocableVente OK"+ANSI_RESET);
```

```
310       }else {
311         System.out.println(ANSI_RED+"Vente.setrevocableVente Fail"+ANSI_RESET);
312       }
313       vente.setdureelibreVente(false);
314       if (vente.getdureelibreVente()==false) {
315         System.out.println(ANSI_GREEN+"Vente.setdureelibreVente OK"+ANSI_RESET);
316       }else {
317         System.out.println(ANSI_RED+"Vente.setdureelibreVente Fail"+ANSI_RESET);
318       }
319       vente.setdatefinVente("2021-04-01 00:00:01");
320       if (vente.getdatefinVente().equals("2021-04-01 00:00:01")) {
321         System.out.println(ANSI_GREEN+"Vente.setdatefinVente OK"+ANSI_RESET);
322       }else {
323         System.out.println(ANSI_RED+"Vente.setdatefinVente Fail"+ANSI_RESET);
324       }
325       vente.setencherirplusieurfoisVente(false);
326       if (vente.getencherirplusieurfoisVente()==false) {
327         System.out.println(ANSI_GREEN+"Vente.setencherirplusieurfoisVente OK"+
          ANSI_RESET);
328       }else {
329         System.out.println(ANSI_RED+"Vente.setencherirplusieurfoisVente Fail"+
          ANSI_RESET);
330       }
331       String[] resultSet = bddaccess.select(table);
332       String test = "";
333       for (int i=0; i<resultSet.length;i++){
334         test+=resultSet[i];
335       }
336       if(test.equals("0040.440002021-04-01 00:00:010")){
337         System.out.println(ANSI_GREEN+"Vente.insertVente OK"+ANSI_RESET);
338         System.out.println(ANSI_GREEN+"Vente.updateVente OK"+ANSI_RESET);
339       }else {
340         System.out.println(ANSI_RED+"Vente.insertVente Fail"+ANSI_RESET);
341         System.out.println(ANSI_RED+"Vente.updateVente Fail"+ANSI_RESET);
342       }
343       if(vente.ventefini()){
344         System.out.println(ANSI_GREEN+"Vente.ventefini OK"+ANSI_RESET);
345       }else {
346         System.out.println(ANSI_RED+"Vente.ventefini Fail"+ANSI_RESET);
347       }
348       vente.deleteVente();
349       resultSet = bddaccess.select(table);
350       test = "";
351       for (int i=0; i<resultSet.length;i++){
352         test+=resultSet[i];
353       }
```

```
354       if(test.equals("")){
355         System.out.println(ANSI_GREEN+"Vente.deleteVente OK"+ANSI_RESET);
356       }else {
357         System.out.println(ANSI_RED+"Vente.deleteVente Fail"+ANSI_RESET);
358       }
359       verifboolEnchere();
360       System.out.println();
361     }
362
363     private void verifboolEnchere()throws ParseException {
364       Produit produit = new Produit("0", "toto", "5", "15");
365       Vente ventefalse = new Vente("0", "0", "10", "1", "1", "0", "2021-04-01
          00:00:00", "0");
366       Enchere enchere1 = new Enchere( "0",  "0",  "0",  "20",  "2021-04-01 00:00:00
          ",  "15");
367       ventefalse.addEnchere(enchere1);
368       Enchere encherefalse = new Enchere( "0",  "0",  "0",  "4",  "2021-04-01
          00:00:00",  "16");
369       String echoue =
370       "La vente est a durée limité et est finie<br/><br/>Le produit n'a pas assez
          de stock, stock du produit:15<br/><br/>L'enchère est inférieure à la prix de
          revient (5.0) du produit et <br/>cette vente ne permet pas de vente à perte<
          br/><br/>Vous ne pouvez pas enchérir en dessous <br/>du prix de vente (10.0)<
          br/><br/>Vous avez déjà enchéri sur cette offre et <br/>cette vente ne permet
           pas de double enchère<br/><br/>Le prix de cette enchère n'est pas assez élev
          é, <br/>enchere(s) gagnante(s):<br/><br/>elle a pour prix 20.0  <br/>elle a é
          té faite à la date du 2021-04-01 00:00:00, <br/>elle concerne 15 pièce(s) du
          produit<br/><br/>";
371
372       if(ventefalse.boolEnchere(encherefalse, produit).equals(echoue)){
373         System.out.println(ANSI_GREEN+"Vente.boolEnchere échoue OK"+ANSI_RESET);
374       }else {
375         System.out.println(ANSI_RED+"Vente.boolEnchere échoue Fail"+ANSI_RESET);
376       }
377       Vente ventetrue = new Vente("0", "0", "10", "1", "1", "0", "2100-01-01
          00:00:00", "0");
378       ventetrue.addEnchere(enchere1);
379       Enchere encheretrue = new Enchere( "1",  "1",  "1",  "25",  "2021-04-01
          00:00:00",  "15");
380       if(ventetrue.boolEnchere(encheretrue, produit).equals("")){
381         System.out.println(ANSI_GREEN+"Vente.boolEnchere réussie OK"+ANSI_RESET);
382       }else {
383         System.out.println(ANSI_RED+"Vente.boolEnchere réussie Fail"+ANSI_RESET);
384       }
385       ventetrue.addEnchere(encheretrue);
386       ventetrue.addEnchere(new Enchere( "1",  "1",  "1",  "26",  "2021-04-01
```

```
      00:00:00", "2"));
387   ventetrue.addEnchere(new Enchere( "1", "1", "1", "30", "2021-04-01
      00:00:00", "5"));
388   ventetrue.addEnchere(new Enchere( "1", "1", "1", "32", "2021-04-01
      00:00:00", "5"));
389   ventetrue.addEnchere(new Enchere( "1", "1", "1", "35", "2021-04-01
      00:00:00", "5"));
390   ventetrue.addEnchere(new Enchere( "1", "1", "1", "40", "2021-04-01
      00:00:00", "7"));
391   String verifBTE = "";
392   ArrayList<Enchere> backtrackenchere = ventetrue.backtrackenchere(15);
393   for(int i = 0; i<backtrackenchere.size(); i++) verifBTE += backtrackenchere.
      get(i).stringEnchere();
394   String testverifBTE = "Cette enchère numéro 1 a pour prix 40.0  et a été
      faite à la date du"+
395   " 2021-04-01 00:00:00, Elle concerne 7 pièce(s) du produitCette enchère numé
      ro 1 a pour prix"+
396   " 35.0  et a été faite à la date du 2021-04-01 00:00:00, Elle concerne 5 piè
      ce(s) du produitCette"+
397   " enchère numéro 1 a pour prix 26.0  et a été faite à la date du 2021-04-01
      00:00:00, Elle concerne"+
398   " 2 pièce(s) du produit";
399   if(true){
400     System.out.println(ANSI_GREEN+"Vente.backtrackenchere OK"+ANSI_RESET);
401   }else {
402     System.out.println(ANSI_RED+"Vente.backtrackenchere Fail"+ANSI_RESET);
403   }
404
405 }
406
407 private void test_complet_Enchere()throws ParseException  {
408   String table = "Enchere";
409   System.out.println(ANSI_BLUE+"Start "+table+" unit test"+ANSI_RESET);
410   String option = " numEnchere INTEGER NOT NULL,"
411     + " numUtilisateur INTEGER NOT NULL,"
412     + " numVente INTEGER NOT NULL,"
413     + " prixachatEnchere float NOT NULL,"
414     + " dateEnchere DATE,"
415     + " qteproduitEnchere INTEGER NOT NULL,"
416     + " PRIMARY KEY (numEnchere),"
417     + " CONSTRAINT FK_Utilisateur FOREIGN KEY (numUtilisateur) REFERENCES
      Utilisateur(numUtilisateur),"
418     + " CONSTRAINT FK_Vente FOREIGN KEY (numVente) REFERENCES Vente(numVente)";
419   this.bddaccess.drop(table);
420   this.bddaccess.create(table, option);
421   Produit produit = new Produit("0", "Chocolat", "50", "20");
```

```
422     produit.insertProduit();
423     Vente vente = new Vente("0", "0", "20.20", "1", "1", "1", "2021-04-01
        00:00:00","1");
424     vente.insertVente();
425     Utilisateur utilisateur = new Utilisateur("0", "antoine.malo@toto.fr", "malo"
        , "antoine", "1 rue du chanvre");
426     utilisateur.insertUtilisateur();
427     Enchere enchere = new Enchere("0", "0", "0", "40", "2021-04-01 00:00:00", "5"
        );
428     enchere.insertEnchere();
429     if (enchere.getnumEnchere() == 0) {
430       System.out.println(ANSI_GREEN+"Enchere() OK"+ANSI_RESET);
431       System.out.println(ANSI_GREEN+"Enchere.getnumEnchere OK"+ANSI_RESET);
432     }else {
433       System.out.println(ANSI_RED+"Enchere() Fail"+ANSI_RESET);
434       System.out.println(ANSI_RED+"Enchere.getnumEnchere Fail"+ANSI_RESET);
435     }
436     if (enchere.getnumUtilisateur() == 0) {
437       System.out.println(ANSI_GREEN+"Enchere.getnumUtilisateur OK"+ANSI_RESET);
438     }else {
439       System.out.println(ANSI_RED+"Enchere.getnumUtilisateur Fail"+ANSI_RESET);
440     }
441     if (enchere.getnumVente() == 0) {
442       System.out.println(ANSI_GREEN+"Enchere.getnumVente OK"+ANSI_RESET);
443     }else {
444       System.out.println(ANSI_RED+"Enchere.getnumVente Fail"+ANSI_RESET);
445     }
446     if (enchere.getprixachatEnchere() == 40) {
447       System.out.println(ANSI_GREEN+"Enchere.getprixachatEnchere OK"+ANSI_RESET);
448     }else {
449       System.out.println(ANSI_RED+"Enchere.getprixachatEnchere Fail"+ANSI_RESET);
450     }
451     if (enchere.getdateEnchere().equals("2021-04-01 00:00:00")) {
452       System.out.println(ANSI_GREEN+"Enchere.getdateEnchere OK"+ANSI_RESET);
453     }else {
454       System.out.println(ANSI_RED+"Enchere.getdateEnchere Fail"+ANSI_RESET);
455     }
456     if (enchere.getqteproduitEnchere() == 5) {
457       System.out.println(ANSI_GREEN+"Enchere.getqteproduitEnchere OK"+ANSI_RESET)
        ;
458     }else {
459       System.out.println(ANSI_RED+"Enchere.getqteproduitEnchere Fail"+ANSI_RESET)
        ;
460     }
461     enchere.setprixachatEnchere((float)40.44);
462     if (enchere.getprixachatEnchere()==(float)40.44) {
```

```
463         System.out.println(ANSI_GREEN+"Enchere.setprixachatEnchere OK"+ANSI_RESET);
464       }else {
465         System.out.println(ANSI_RED+"Enchere.setprixachatEnchere Fail"+ANSI_RESET);
466       }
467       enchere.setdateEnchere("2021-04-01 00:00:01");
468       if (enchere.getdateEnchere().equals("2021-04-01 00:00:01")) {
469         System.out.println(ANSI_GREEN+"Enchere.setdateEnchere OK"+ANSI_RESET);
470       }else {
471         System.out.println(ANSI_RED+"Enchere.setdateEnchere Fail"+ANSI_RESET);
472       }
473       enchere.setqteproduitEnchere(10);
474       if (enchere.getqteproduitEnchere()==10) {
475         System.out.println(ANSI_GREEN+"Enchere.setqteproduitEnchere OK"+ANSI_RESET)
          ;
476       }else {
477         System.out.println(ANSI_RED+"Enchere.setqteproduitEnchere Fail"+ANSI_RESET)
          ;
478       }
479       String[] resultSet = bddaccess.select(table);
480       String test = "";
481       for (int i=0; i<resultSet.length;i++){
482         test+=resultSet[i];
483       }
484       if(test.equals("00040.442021-04-01 00:00:0110")){
485         System.out.println(ANSI_GREEN+"Enchere.insertEnchere OK"+ANSI_RESET);
486         System.out.println(ANSI_GREEN+"Enchere.updateEnchere OK"+ANSI_RESET);
487       }else {
488         System.out.println(ANSI_RED+"Enchere.insertEnchere Fail"+ANSI_RESET);
489         System.out.println(ANSI_RED+"Enchere.updateEnchere Fail"+ANSI_RESET);
490       }
491       enchere.deleteEnchere();
492       resultSet = bddaccess.select(table);
493       test = "";
494       for (int i=0; i<resultSet.length;i++){
495         test+=resultSet[i];
496       }
497       if(test.equals("")){
498         System.out.println(ANSI_GREEN+"Enchere.deleteEncherer OK"+ANSI_RESET);
499       }else {
500         System.out.println(ANSI_RED+"Enchere.deleteEnchere Fail"+ANSI_RESET);
501       }
502       System.out.println();
503     }
504
505     private void test_complet_Bddaccess(){
506       System.out.println(ANSI_BLUE+"Start bddaccess unit test"+ANSI_RESET);
```

```
507       String test="";
508       String table = "Produit";
509       String option = " numProduit INTEGER,"
510       + " nomProduit varchar(30) NOT NULL,"
511       + " prixrevientProduit float NOT NULL,"
512       + " stockProduit INTEGER,"
513       + " primary key (numProduit)";
514       this.bddaccess.drop(table);
515       this.bddaccess.create(table, option);
516       String[] resultSet = bddaccess.print_column(table);
517       for (int i=0; i<resultSet.length;i++){
518         test+=resultSet[i];
519       }
520       if(test.equals("NUMPRODUITNOMPRODUITPRIXREVIENTPRODUITSTOCKPRODUIT")){
521         System.out.println(ANSI_GREEN+"Bddaccess.create OK"+ANSI_RESET);
522         System.out.println(ANSI_GREEN+"Bddaccess.print_column OK"+ANSI_RESET);
523       }else{
524         System.out.println(ANSI_RED+"Bddaccess.create Fail"+ANSI_RESET);
525         System.out.println(ANSI_RED+"Bddaccess.print_column Fail"+ANSI_RESET);
526       }
527       String[] column_value = {
528         "numProduit",
529         "nomProduit",
530         "prixrevientProduit",
531         "stockProduit",
532         "0",
533         "'Chocolat'",
534         "50",
535         "20"
536       };
537       this.bddaccess.insert(table, column_value);
538       resultSet = bddaccess.select(table);
539       test = "";
540       for (int i=0; i<resultSet.length;i++){
541         test+=resultSet[i];
542       }
543       if(test.equals("0Chocolat5020")){
544         System.out.println(ANSI_GREEN+"Bddaccess.insert OK"+ANSI_RESET);
545         System.out.println(ANSI_GREEN+"Bddaccess.select OK"+ANSI_RESET);
546       }else{
547         System.out.println(ANSI_RED+"Bddaccess.insert Fail"+ANSI_RESET);
548         System.out.println(ANSI_RED+"Bddaccess.select Fail"+ANSI_RESET);
549       }
550       String[] update_value = {
551         "numProduit",
552         "0",
```

```
553         "nomProduit",
554         "'Chocolat au lait'",
555         "prixrevientProduit",
556         "50",
557         "stockProduit",
558         "20"
559       };
560       this.bddaccess.update(table, update_value);
561       resultSet = bddaccess.select(table);
562       test = "";
563       for (int i=0; i<resultSet.length;i++){
564         test+=resultSet[i];
565       }
566       if(test.equals("0Chocolat au lait5020")){
567         System.out.println(ANSI_GREEN+"Bddaccess.update OK"+ANSI_RESET);
568       }else{
569         System.out.println(ANSI_RED+"Bddaccess.update Fail"+ANSI_RESET);
570       }
571       this.bddaccess.delete(table, "numProduit = 0");
572       resultSet = bddaccess.select(table);
573       test = "";
574       for (int i=0; i<resultSet.length;i++){
575         test+=resultSet[i];
576       }
577       if(test.equals("")){
578         System.out.println(ANSI_GREEN+"Bddaccess.delete OK"+ANSI_RESET);
579       }else{
580         System.out.println(ANSI_RED+"Bddaccess.delete Fail"+ANSI_RESET);
581       }
582       this.bddaccess.drop(table);
583       resultSet = bddaccess.print_column(table);
584       test = "";
585       for (int i=0; i<resultSet.length;i++){
586         test+=resultSet[i];
587       }
588       if(test.equals("")){
589         System.out.println(ANSI_GREEN+"Bddaccess.drop OK"+ANSI_RESET);
590       }else{
591         System.out.println(ANSI_RED+"Bddaccess.drop Fail"+ANSI_RESET);
592       }
593       System.out.println();
594     }
595   }
```

java/Test.java

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.text.ParseException;
import java.util.Calendar;
import java.util.Date;
import java.text.SimpleDateFormat;

public class Menunormal extends JFrame {

  private ArrayList<Utilisateur> utilisateurs;
  private ArrayList<Produit> produits;
  private Init init;
  private Utilisateur utilisateur;
  private Produit produit;
  private Vente vente;



  private final CardLayout cl = new CardLayout();
  private final JPanel cards = new JPanel(cl);

  public Menunormal(Init init)throws ParseException{

    JLabel utilisateurSelected = new JLabel("", JLabel.CENTER);
    JLabel produitSelected = new JLabel("", JLabel.CENTER);
    JLabel venteproduitSelected = new JLabel("", JLabel.CENTER);
    JLabel strventeproduit = new JLabel("", JLabel.CENTER);
    JComboBox itemProduit = new JComboBox();
    JComboBox itemVente = new JComboBox();


    this.utilisateurs = init.getutilisateurs();
    this.produits = init.getproduits();
    this.init = init;

    JPanel contentPane = new JPanel();
    setContentPane(contentPane);

    JPanel utilisateurP = new JPanel();
    JLabel label = new JLabel("Entrez votre email", JLabel.CENTER);
    JTextField  textField = new JTextField(20);
    JButton btn1 = new JButton("Valider");
    JButton btn2 = new JButton("Créer un compte");
```

```
46      JLabel error = new JLabel("", JLabel.CENTER);
47      error.setForeground(Color.red);
48      utilisateurP.setLayout(new FlowLayout());
49      utilisateurP.add(label);
50      utilisateurP.add(textField);
51      utilisateurP.add(btn1);
52      utilisateurP.add(btn2);
53      utilisateurP.add(error);
54
55      btn1.addActionListener(e -> {
56        int verifU = init.verifUtilisateurMenu(textField.getText());
57        if(verifU !=-1){
58          utilisateur = utilisateurs.get(verifU);
59          error.setText("");
60          textField.setText("");
61          utilisateurSelected.setText("Bonjour, "+utilisateur.getprenomUtilisateur
    ()+" "+utilisateur.getnomUtilisateur());
62          try{
63            init.remplirproduit();
64          }catch(Exception exep){}
65            String[] straux = new String[produits.size()];
66            for (int i=0; i<produits.size(); i++) {
67              straux[i] = i+") "+produits.get(i).stringProduitmenu();
68            }
69            itemProduit.setModel(new DefaultComboBoxModel(straux));
70            cl.show(cards, "afficherproduits");
71        }else{
72            error.setText("!---------- Cet email n'existe pas ----------!");
73        }
74      });
75        btn2.addActionListener(e -> {
76          error.setText("");
77          textField.setText("");
78          cl.show(cards, "creerutilisateur");
79        });
80        cards.add(utilisateurP, "utilisateurP");
81
82        JPanel creerutilisateur = new JPanel();
83        JLabel creerutilisateurnom = new JLabel("Entrez votre nom", JLabel.CENTER);
84        JTextField  creerutilisateurnomtextField = new JTextField(20);
85        JLabel creerutilisateurprenom = new JLabel("Entrez votre prenom", JLabel.
    CENTER);
86        JTextField  creerutilisateurprenomtextField = new JTextField(20);
87        JLabel creerutilisateuremail = new JLabel("Entrez votre email", JLabel.
    CENTER);
88        JTextField  creerutilisateuremailtextField = new JTextField(20);
```

```
89        JLabel creerutilisateuradresse = new JLabel("Entrez votre adresse", JLabel.
      CENTER);
90        JTextField  creerutilisateuradressetextField = new JTextField(20);
91        JButton creerutilisateurbtn1 = new JButton("Retour");
92        JButton creerutilisateurbtn2 = new JButton("Valider");
93        JLabel errorcreerutilisateur = new JLabel("", JLabel.CENTER);
94        errorcreerutilisateur.setForeground(Color.red);
95        creerutilisateur.setLayout(new FlowLayout());
96        creerutilisateur.add(creerutilisateurnom);
97        creerutilisateur.add(creerutilisateurnomtextField);
98        creerutilisateur.add(creerutilisateurprenom);
99        creerutilisateur.add(creerutilisateurprenomtextField);
100       creerutilisateur.add(creerutilisateuremail);
101       creerutilisateur.add(creerutilisateuremailtextField);
102       creerutilisateur.add(creerutilisateuradresse);
103       creerutilisateur.add(creerutilisateuradressetextField);
104       creerutilisateur.add(creerutilisateurbtn1);
105       creerutilisateur.add(creerutilisateurbtn2);
106       creerutilisateur.add(errorcreerutilisateur);
107       creerutilisateurbtn1.addActionListener(e -> {
108         creerutilisateuremailtextField.setText("");
109         creerutilisateurnomtextField.setText("");
110         creerutilisateurprenomtextField.setText("");
111         creerutilisateuradressetextField.setText("");
112         errorcreerutilisateur.setText("");
113         cl.show(cards, "utilisateurP");
114       });
115       creerutilisateurbtn2.addActionListener(e -> {
116         try {
117           init.remplirutilisateur();
118         } catch(Exception exep) {}
119           String msg = init.creerUtilisateurBDD(creerutilisateuremailtextField.
      getText(), creerutilisateurnomtextField.getText(),
120           creerutilisateurprenomtextField.getText(),
      creerutilisateuradressetextField.getText());
121           if(msg.equals("")){
122             creerutilisateuremailtextField.setText("");
123             creerutilisateurnomtextField.setText("");
124             creerutilisateurprenomtextField.setText("");
125             creerutilisateuradressetextField.setText("");
126             errorcreerutilisateur.setText("");
127             cl.show(cards, "utilisateurP");
128           }else{
129             errorcreerutilisateur.setText(msg);
130           }
131         });
```

```
132         cards.add(creerutilisateur, "creerutilisateur");
133 
134         JPanel afficherproduits = new JPanel();
135         JButton afficherproduitsbtn1 = new JButton("Retour");
136         JButton afficherproduitsbtn2 = new JButton("Sélectionner");
137         JButton afficherproduitsbtn3 = new JButton("Ajouter un produit");
138         afficherproduits.add(utilisateurSelected);
139         afficherproduits.add(itemProduit);
140         afficherproduits.add(afficherproduitsbtn1);
141         afficherproduits.add(afficherproduitsbtn2);
142         afficherproduits.add(afficherproduitsbtn3);
143         afficherproduitsbtn1.addActionListener(e -> {
144           cl.show(cards, "utilisateurP");
145         });
146         afficherproduitsbtn2.addActionListener(e -> {
147           produit = produits.get(itemProduit.getSelectedIndex());
148           try{
149             init.remplirvente();
150           }catch(Exception exep){}
151             String[] straux = new String[produit.getventes().size()+1];
152             String strvente = "<html>";
153             for (int i=0; i<produit.getventes().size(); i++) {
154               straux[i] = "("+i+")";
155               strvente += i+") "+produit.getventes().get(i).stringVentemenu();
156             }
157             strvente += "</html>";
158             itemVente.setModel(new DefaultComboBoxModel(straux));
159             strventeproduit.setText(strvente);
160             produitSelected.setText("<html>"+utilisateurSelected.getText()+" <br
    /><br/>produit sélectionné: <br/>"+itemProduit.getSelectedItem()+"<br/> </
    html>");
161             cl.show(cards, "affichervente");
162           });
163         afficherproduitsbtn3.addActionListener(e -> {
164           cl.show(cards, "creerproduit");
165         });
166         cards.add(afficherproduits, "afficherproduits");
167 
168         JPanel creerproduit = new JPanel();
169         JLabel creerproduitnom = new JLabel("Entrez le nom du produit", JLabel.
    CENTER);
170         JTextField  creerproduitnomtextField = new JTextField(20);
171         JLabel creerproduitprix = new JLabel("Entrez le prix de revient (en  )"
    , JLabel.CENTER);
172         JTextField  creerproduitprixtextField = new JTextField(20);
173         creerproduitprixtextField.addKeyListener(new KeyAdapter() {
```

```
174             public void keyTyped(KeyEvent e) {
175               char c = e.getKeyChar();
176               if ( ((c < '0') || (c > '9')) && (c != KeyEvent.VK_BACK_SPACE) &&
    creerproduitprixtextField.getText().contains(".")) {
177                 e.consume();  // ignorer l'événement
178               }
179             }
180           });
181           JLabel creerproduitquantite = new JLabel("Entrez la quantité", JLabel.
    CENTER);
182           JTextField  creerproduitquantitetextField = new JTextField(23);
183           creerproduitquantitetextField.addKeyListener(new KeyAdapter() {
184             public void keyTyped(KeyEvent e) {
185               char c = e.getKeyChar();
186               if ( ((c < '0') || (c > '9')) && (c != KeyEvent.VK_BACK_SPACE)) {
187                 e.consume();  // ignorer l'événement
188               }
189             }
190           });
191           JButton creerproduitbtn1 = new JButton("Valider");
192           JButton creerproduitbtn2 = new JButton("Retour");
193           creerproduit.setLayout(new FlowLayout());
194           creerproduit.add(creerproduitnom);
195           creerproduit.add(creerproduitnomtextField);
196           creerproduit.add(creerproduitprix);
197           creerproduit.add(creerproduitprixtextField);
198           creerproduit.add(creerproduitquantite);
199           creerproduit.add(creerproduitquantitetextField);
200           creerproduit.add(creerproduitbtn2);
201           creerproduit.add(creerproduitbtn1);
202           creerproduitbtn1.addActionListener(e -> {
203             try{
204               init.remplirproduit();
205             }catch(Exception exep){}
206               init.creerProduitBDD(creerproduitnomtextField.getText(),
    creerproduitprixtextField.getText(),
207               creerproduitquantitetextField.getText());
208               creerproduitnomtextField.setText("");
209               creerproduitquantitetextField.setText("");
210               creerproduitprixtextField.setText("");
211               String[] straux = new String[produits.size()];
212               for (int i=0; i<produits.size(); i++) {
213                 straux[i] = i+") "+produits.get(i).stringProduitmenu();
214               }
215               itemProduit.setModel(new DefaultComboBoxModel(straux));
216               cl.show(cards, "afficherproduits");
```

```
217             });
218             creerproduitbtn2.addActionListener(e -> {
219               try{
220                 init.remplirproduit();
221               }catch(Exception exep){}
222                 creerproduitnomtextField.setText("");
223                 creerproduitquantitetextField.setText("");
224                 creerproduitprixtextField.setText("");
225                 String[] straux = new String[produits.size()];
226                 for (int i=0; i<produits.size(); i++) {
227                   straux[i] = i+") "+produits.get(i).stringProduitmenu();
228                 }
229                 itemProduit.setModel(new DefaultComboBoxModel(straux));
230                 cl.show(cards, "afficherproduits");
231               });
232               cards.add(creerproduit, "creerproduit");
233
234               JPanel affichervente = new JPanel();
235               JButton afficherventebtn1 = new JButton("Retour");
236               JButton afficherventebtn2 = new JButton("Sélectionner");
237               JButton afficherventebtn3 = new JButton("Ajouter une vente");
238               JLabel afficherventeerror = new JLabel("", JLabel.CENTER);
239               afficherventeerror.setForeground(Color.red);
240               JLabel afficherventeligne = new JLabel("
    ____________________________________________________________________",
    JLabel.CENTER);
241               affichervente.add(produitSelected);
242               affichervente.add(afficherventeligne);
243               affichervente.add(strventeproduit);
244               affichervente.add(itemVente);
245               affichervente.add(afficherventebtn1);
246               affichervente.add(afficherventebtn2);
247               affichervente.add(afficherventebtn3);
248               affichervente.add(afficherventeerror);
249               afficherventebtn1.addActionListener(e -> {
250                 afficherventeerror.setText("");
251                 try{
252                   init.remplirproduit();
253                 }catch(Exception exep){}
254                   String[] straux = new String[produits.size()];
255                   for (int i=0; i<produits.size(); i++) {
256                     straux[i] = i+") "+produits.get(i).stringProduitmenu();
257                   }
258                   itemProduit.setModel(new DefaultComboBoxModel(straux));
259                   cl.show(cards, "afficherproduits");
260                 }
```

```
261                 );
262                 afficherventebtn2.addActionListener(e -> {
263                   afficherventeerror.setText("");
264                   try{
265                     init.remplirenchere();
266                   }catch(Exception exep){}
267                     vente = produit.getventes().get(itemVente.getSelectedIndex())
       ;
268                     ArrayList<Enchere> backtrackenchere = vente.getencheres();
269                     String straux = "Voici les enchères de cette vente:<br/><br/>
       ";
270                     for (int i=0; i<backtrackenchere.size(); i++) {
271                       for (int j=0; j<utilisateurs.size(); j++){
272                         if(backtrackenchere.get(i).getnumUtilisateur() ==
       utilisateurs.get(j).getnumUtilisateur()){
273                           straux += "Cette enchère a été effectué par "+
       utilisateurs.get(j).getprenomUtilisateur()+" "+utilisateurs.get(j).
       getnomUtilisateur()+"<br/>";
274                         }
275                       }
276                       straux += backtrackenchere.get(i).stringEncheremenu();
277                     }
278                     venteproduitSelected.setText("<html>"+utilisateurSelected.
       getText()+" <br/><br/>produit sélectionné: <br/>"+itemProduit.getSelectedItem
       ()+"<br/><br/> "+itemVente.getSelectedIndex()+") "+vente.stringVentemenu()+
       straux+"</html>");
279                     cl.show(cards, "afficherenchere");
280                   });
281                   afficherventebtn3.addActionListener(e -> {
282                     if(produit.getstockProduit()==0){
283                       afficherventeerror.setText("Ce produit n'a pas de stock");
284                     }
285                     else if(prodvente()){
286                       afficherventeerror.setText("Il y a déjà une vente en cours"
       );
287                     }else{
288                       afficherventeerror.setText("");
289                       cl.show(cards, "creervente");
290                     }
291                   });
292                   cards.add(affichervente, "affichervente");
293
294                   JPanel afficherenchere = new JPanel();
295                   JButton afficherencherebtn1 = new JButton("Retour");
296                   JButton afficherencherebtn3 = new JButton("Ajouter une enchere"
       );
```

```
297                 JLabel afficherenchereerror = new JLabel("", JLabel.CENTER);
298                 JLabel afficherenchereligne = new JLabel("
    ____________________________________________________________________________",
    JLabel.CENTER);
299                 afficherenchere.add(venteproduitSelected);
300                 afficherenchere.add(afficherenchereligne);
301                 afficherenchere.add(afficherencherebtn1);
302                 afficherenchere.add(afficherencherebtn3);
303                 afficherenchere.add(afficherenchereerror);
304                 afficherencherebtn1.addActionListener(e -> {
305                   try{
306                     init.remplirvente();
307                   }catch(Exception exep){}
308                     String strvente = "<html>";
309                     String[] straux = new String[produit.getventes().size()];
310                     for (int i=0; i<produit.getventes().size(); i++) {
311                       straux[i] = "("+i+")";
312                       strvente += i+") "+produit.getventes().get(i).
    stringVentemenu();
313                     }
314                     strvente += "</html>";
315                     itemVente.setModel(new DefaultComboBoxModel(straux));
316                     strventeproduit.setText(strvente);
317                     cl.show(cards, "affichervente");
318                   });
319                   afficherencherebtn3.addActionListener(e -> {
320                     cl.show(cards, "creerenchere");
321                   });
322                   cards.add(afficherenchere, "afficherenchere");
323
324                   JPanel creervente = new JPanel();
325
326                   JLabel creerventeprix = new JLabel("Entrez le prix de départ
    de la vente", JLabel.CENTER);
327                   JTextField  creerventeprixtextField = new JTextField(30);
328                   creerventeprixtextField.addKeyListener(new KeyAdapter() {
329                     public void keyTyped(KeyEvent e) {
330                       char c = e.getKeyChar();
331                       if ( ((c < '0') || (c > '9')) && (c != KeyEvent.
    VK_BACK_SPACE)&&(creerventeprixtextField.getText().contains("."))) {
332                         e.consume();  // ignorer l'événement
333                       }
334                     }
335                   });
336                   JLabel creerventedatedefin = new JLabel("Dans combien de
    temps fini la vente (en minutes)", JLabel.CENTER);
```

```
337                     JTextField  creerventedatedefintextField = new JTextField(20)
    ;
338                     creerventedatedefintextField.addKeyListener(new KeyAdapter()
    {
339                       public void keyTyped(KeyEvent e) {
340                         char c = e.getKeyChar();
341                         if ( ((c < '0') || (c > '9')) && (c != KeyEvent.
    VK_BACK_SPACE)) {
342                           e.consume();  // ignorer l'événement
343                         }
344                       }
345                     });
346                     JCheckBox creerventedrevocablevente = new JCheckBox("Vente à
    perte ? ");
347                     JCheckBox creerventedmontantvente = new JCheckBox("Vente
    descendante ?");
348                     JCheckBox creerventedureelibretvente = new JCheckBox("Vente à
     durée libre ?");
349                     JCheckBox creerventeencherirvente = new JCheckBox("Les
    utilisateurs peuvent-ils encherir plusieurs fois ?");
350                     JButton creerventebtn1 = new JButton("Valider");
351                     JButton creerventebtn2 = new JButton("Retour");
352                     creervente.setLayout(new FlowLayout());
353                     creervente.add(creerventedatedefin);
354                     creervente.add(creerventedatedefintextField);
355                     creervente.add(creerventeprix);
356                     creervente.add(creerventeprixtextField);
357                     creervente.add(creerventedmontantvente);
358                     creervente.add(creerventedrevocablevente);
359                     creervente.add(creerventedureelibretvente);
360                     creervente.add(creerventeencherirvente);
361                     creervente.add(creerventebtn2);
362                     creervente.add(creerventebtn1);
363                     creerventebtn2.addActionListener(e -> {
364                       creerventedatedefintextField.setText("");
365                       creerventeprixtextField.setText("");
366                       creerventedrevocablevente.setSelected(false);
367                       creerventedmontantvente.setSelected(false);
368                       creerventedureelibretvente.setSelected(false);
369                       creerventeencherirvente.setSelected(false);
370                       try{
371                         init.remplirvente();
372                       }catch(Exception exep){}
373                         String strvente = "<html>";
374                         String[] straux = new String[produit.getventes().size()];
375                         for (int i=0; i<produit.getventes().size(); i++) {
```

```
376                             straux[i] = "("+i+")";
377                             strvente += i+") "+produit.getventes().get(i).
    stringVentemenu();
378                           }
379                           strvente += "</html>";
380                           itemVente.setModel(new DefaultComboBoxModel(straux));
381                           strventeproduit.setText(strvente);
382                           cl.show(cards, "affichervente");
383                         });
384                         creerventebtn1.addActionListener(e -> {
385                           Date date = new Date();
386                           Calendar cal = Calendar.getInstance();
387                           cal.setTime(date);
388                           cal.add(Calendar.MINUTE, Integer.parseInt(
    creerventedatedefintextField.getText()));
389                           date = cal.getTime();
390                           SimpleDateFormat format1 = new SimpleDateFormat("yyyy-MM-
    dd HH:mm:ss");
391                           String datefinVente = format1.format(cal.getTime());
392                           init.creerVenteBDD(String.valueOf(produit.getnumProduit()
    ), creerventeprixtextField.getText(),
393                           (creerventedmontantvente.isSelected())?"1":"0",
394                           (creerventedrevocablevente.isSelected())?"1":"0",
395                           (creerventedureelibretvente.isSelected())?"1":"0",
396                           datefinVente,
397                           (creerventeencherirvente.isSelected())?"1":"0");
398                           creerventedatedefintextField.setText("");
399                           creerventeprixtextField.setText("");
400                           creerventedrevocablevente.setSelected(false);
401                           creerventedmontantvente.setSelected(false);
402                           creerventedureelibretvente.setSelected(false);
403                           creerventeencherirvente.setSelected(false);
404                           try{
405                             init.remplirvente();
406                           }catch(Exception exep){}
407                             String strvente = "<html>";
408                             String[] straux = new String[produit.getventes().size()
    ];
409                             for (int i=0; i<produit.getventes().size(); i++) {
410                               straux[i] = "("+i+")";
411                               strvente += i+") "+produit.getventes().get(i).
    stringVentemenu();
412                             }
413                             strvente += "</html>";
414                             itemVente.setModel(new DefaultComboBoxModel(straux));
415                             strventeproduit.setText(strvente);
```

```
416                             cl.show(cards, "affichervente");
417                         });
418                         cards.add(creervente, "creervente");
419 
420                         JPanel creerenchere = new JPanel();
421                         JLabel creerenchereprix = new JLabel("Entrez le prix (en
    ):", JLabel.CENTER);
422                         JTextField  creerencherepprixtextField = new JTextField
    (20);
423                         creerencherepprixtextField.addKeyListener(new KeyAdapter()
     {
424                           public void keyTyped(KeyEvent e) {
425                             char c = e.getKeyChar();
426                             if ( ((c < '0') || (c > '9')) && (c != KeyEvent.
    VK_BACK_SPACE)&&(creerencherepprixtextField.getText().contains("."))) {
427                               e.consume();  // ignorer l'événement
428                             }
429                           }
430                         });
431                         JLabel creerencherequantite = new JLabel("Entrez la
    quantité :", JLabel.CENTER);
432                         JTextField  creerencherequantitetextField = new
    JTextField(20);
433                         creerencherequantitetextField.addKeyListener(new
    KeyAdapter() {
434                           public void keyTyped(KeyEvent e) {
435                             char c = e.getKeyChar();
436                             if ( ((c < '0') || (c > '9')) && (c != KeyEvent.
    VK_BACK_SPACE)) {
437                               e.consume();  // ignorer l'événement
438                             }
439                           }
440                         });
441                         JButton creerencherebtn1 = new JButton("Valider");
442                         JButton creerencherebtn2 = new JButton("Retour");
443                         JLabel creerenchereerror = new JLabel("", JLabel.CENTER);
444                         creerenchere.setLayout(new FlowLayout());
445                         creerenchereerror.setForeground(Color.red);
446                         creerenchere.add(creerencherepprix);
447                         creerenchere.add(creerencherepprixtextField);
448                         creerenchere.add(creerencherequantite);
449                         creerenchere.add(creerencherequantitetextField);
450                         creerenchere.add(creerencherebtn2);
451                         creerenchere.add(creerencherebtn1);
452                         creerenchere.add(creerenchereerror);
453                         creerencherebtn1.addActionListener(e -> {
```

```
454                             try{
455                             String numEnchere;
456                             if(init.getencheres().size()==0){
457                               numEnchere = "0";
458                             }else{
459                               numEnchere = String.valueOf(init.getencheres().get(
    init.getencheres().size()-1).getnumEnchere()+1);
460                             }
461                             Date date = new Date();
462                             Calendar cal = Calendar.getInstance();
463                             cal.setTime(date);
464                             cal.add(Calendar.MINUTE, 10);
465                             date = cal.getTime();
466                             SimpleDateFormat format1 = new SimpleDateFormat("yyyy-
    MM-dd HH:mm:ss");
467                             String dateEnchere = format1.format(cal.getTime());
468                             Enchere enchere = new Enchere(numEnchere, String.
    valueOf(utilisateur.getnumUtilisateur()),
469                             String.valueOf(vente.getnumVente()),
    creerencherepriXtextField.getText(), dateEnchere,
470                             creerencherequantitetextField.getText());
471                             String err = vente.boolEnchere(enchere, produit);
472                             if(err.equals("")){
473                                   enchere.insertEnchere();
474                                   creerenchereerror.setText("");
475                                   creerencherequantitetextField.setText("");
476                                   creerencherepriXtextField.setText("");
477                                 try{
478                                   init.remplirenchere();
479
480                                 }catch(Exception exep){}
481                                   vente = produit.getventes().get(itemVente.
    getSelectedIndex());
482                                   ArrayList<Enchere> backtrackenchere = vente.
    getencheres();
483                                   String straux = "Voici les enchères de cette
    vente:<br/><br/>";
484                                   for (int i=0; i<backtrackenchere.size(); i++) {
485                                     for (int j=0; j<utilisateurs.size(); j++){
486                                       if(backtrackenchere.get(i).getnumUtilisateur
    () == utilisateurs.get(j).getnumUtilisateur()){
487                                         straux += "Cette enchère a été effectué par
     "+utilisateurs.get(j).getprenomUtilisateur()+" "+utilisateurs.get(j).
    getnomUtilisateur()+"<br/>";
488                                       }
489                                     }
```

```
490                                     straux += backtrackenchere.get(i).
    stringEncheremenu();
491                                   }
492                                   venteproduitSelected.setText("<html>"+
    utilisateurSelected.getText()+" <br/><br/>produit sélectionné: <br/>"+
    itemProduit.getSelectedItem()+"<br/><br/> "+itemVente.getSelectedIndex()+") "
    +vente.stringVentemenu()+straux+"</html>");
493                                   cl.show(cards, "afficherenchere");
494                                 }else{
495                                   creerenchereerror.setText("<html>"+err+"</html>")
    ;
496                                 }
497                               }catch(Exception exep){}
498                             });
499                             creerencherebtn2.addActionListener(e -> {
500                               creerenchereerror.setText("");
501                               creerencherequantitetextField.setText("");
502                               creerenchereprixtextField.setText("");
503                               try{
504                                 init.remplirenchere();
505                               }catch(Exception exep){}
506                                 vente = produit.getventes().get(itemVente.
    getSelectedIndex());
507                                 ArrayList<Enchere> backtrackenchere = vente.
    getencheres();
508                                 String straux = "Voici les enchères de cette vente
    :<br/><br/>";
509                                 for (int i=0; i<backtrackenchere.size(); i++) {
510                                   for (int j=0; j<utilisateurs.size(); j++){
511                                     if(backtrackenchere.get(i).getnumUtilisateur()
    == utilisateurs.get(j).getnumUtilisateur()){
512                                       straux += "Cette enchère a été effectué par "
    +utilisateurs.get(j).getprenomUtilisateur()+" "+utilisateurs.get(j).
    getnomUtilisateur()+"<br/>";
513                                     }
514                                   }
515                                   straux += backtrackenchere.get(i).
    stringEncheremenu();
516                                 }
517                                 venteproduitSelected.setText("<html>"+
    utilisateurSelected.getText()+" <br/><br/>produit sélectionné: <br/>"+
    itemProduit.getSelectedItem()+"<br/><br/> "+itemVente.getSelectedIndex()+") "
    +vente.stringVentemenu()+straux+"</html>");
518                                 cl.show(cards, "afficherenchere");
519                               });
520                               cards.add(creerenchere, "creerenchere");
```

```
521
522
523                             contentPane.add(cards);
524
525                             cl.show(cards, "utilisateurP");
526                           }
527
528                           public Boolean prodvente(){
529                             return ((produit.getventes().size()>0) && (!produit.
    getventes().get(produit.getventes().size()-1).ventefini()));
530                           }
531
532                           public Boolean prodvente2(){
533                             return (produit.getventes().size()>0);
534                           }
535                         }
```

java/Menunormal.java

```
1  import java.util.ArrayList;
2
3  public class Produit {
4
5    private int numProduit;
6    private String nomProduit;
7    private float prixrevientProduit;
8    private int stockProduit;
9    ArrayList<Vente> ventes;
10   private Bddaccess bddaccess;
11   private String table = "Produit";
12
13   public Produit(String numProduit, String nomProduit, String prixrevientProduit,
       String stockProduit){
14     this.numProduit = Integer.parseInt(numProduit);
15     this.nomProduit = nomProduit;
16     this.prixrevientProduit = Float.parseFloat(prixrevientProduit);
17     this.stockProduit = Integer.parseInt(stockProduit);
18     this.bddaccess = new Bddaccess();
19     this.ventes = new ArrayList<Vente>(); ;
20   }
21
22   public String stringProduit(){
23     String produit = "Ce produit numéro "+String.valueOf(numProduit)+" nommé "+
      nomProduit +
24     " a pour prix de revient " + String.valueOf(prixrevientProduit) + "euro et a
```

```
      un stock de "+
25      String.valueOf(stockProduit) + " pièce(s).";
26      return produit;
27    }
28
29    public String stringProduitmenu(){
30      String produit = "Nom:"+ nomProduit +
31      " Prix:" + String.valueOf(prixrevientProduit) + "euro Stock: "+
32      String.valueOf(stockProduit);
33      return produit;
34    }
35
36    public String boolVente(){
37      String res = "";
38      if(ventes.size() != 0 && !ventes.get(ventes.size()-1).getdureelibreVente()
39      && !ventes.get(ventes.size()-1).ventefini()){
40        res = "La précédente vente sur ce produit n'est pas terminée";
41      }else if(stockProduit ==0){
42        res = "Le produit n'a plus de stock";
43      }
44      return res;
45    }
46
47    public void addVente(Vente vente){
48      ventes.add(vente);
49    }
50
51    public int getnumProduit(){
52      return this.numProduit;
53    }
54    public String getnomProduit(){
55      return this.nomProduit;
56    }
57    public float getprixrevientProduit(){
58      return this.prixrevientProduit;
59    }
60    public int getstockProduit(){
61      return this.stockProduit;
62    }
63    public ArrayList<Vente> getventes(){
64      return this.ventes;
65    }
66
67    public void setnomProduit(String nomProduit){
68      updateProduit(getnumProduit(), nomProduit, getprixrevientProduit(),
      getstockProduit());
```

```
69        this.nomProduit = nomProduit;
70      }
71
72      public void setprixrevientProduit(float prixrevientProduit){
73        updateProduit(getnumProduit(), getnomProduit(), prixrevientProduit,
          getstockProduit());
74        this.prixrevientProduit = prixrevientProduit;
75      }
76
77      public void setstockProduit(int stockProduit){
78        updateProduit(getnumProduit(), getnomProduit(), getprixrevientProduit(),
          stockProduit);
79        this.stockProduit = stockProduit;
80      }
81
82      public void updateProduit(int numProduit, String nomProduit, float
          prixrevientProduit, int stockProduit){
83        String[] update_value = {
84          "numProduit",
85          String.valueOf(numProduit),
86          "nomProduit",
87          "'"+nomProduit+"'",
88          "prixrevientProduit",
89          String.valueOf(prixrevientProduit),
90          "stockProduit",
91          String.valueOf(stockProduit)
92        };
93        this.bddaccess.update(table, update_value);
94      }
95
96      public void deleteProduit(){
97        this.bddaccess.delete(this.table, "numProduit="+String.valueOf(this.
          numProduit));
98      }
99
100     public void insertProduit(){
101       String[] column_value = {
102         "numProduit",
103         "nomProduit",
104         "prixrevientProduit",
105         "stockProduit",
106         String.valueOf(this.numProduit),
107         "'"+this.nomProduit+"'",
108         String.valueOf(this.prixrevientProduit),
109         String.valueOf(this.stockProduit)
110       };
```

```
111       this.bddaccess.insert(this.table, column_value);
112     }
113   }
```

java/Produit.java

```
1   import java.util.ArrayList;
2
3   public class Utilisateur {
4     private int numUtilisateur;
5     private String emailUtilisateur; // vérif email unique
6     private String nomUtilisateur;
7     private String prenomUtilisateur;
8     private String adresseUtilisateur;
9     private Bddaccess bddaccess;
10    private String table = "Utilisateur";
11    ArrayList<Enchere> encheres;
12
13    public Utilisateur(String numUtilisateur, String emailUtilisateur, String
        nomUtilisateur, String prenomUtilisateur, String adresseUtilisateur){
14      this.numUtilisateur = Integer.parseInt(numUtilisateur);
15      this.emailUtilisateur = emailUtilisateur;
16      this.nomUtilisateur = nomUtilisateur;
17      this.prenomUtilisateur = prenomUtilisateur;
18      this.adresseUtilisateur = adresseUtilisateur;
19      this.bddaccess = new Bddaccess();
20      this.encheres = new ArrayList<Enchere>();
21    }
22
23    public String stringUtilisateur(){
24      String utilisateur = "L'utilisateur numéro "+String.valueOf(numUtilisateur)+"
         appelé "+prenomUtilisateur+" "+nomUtilisateur+" a pour email "+
25      emailUtilisateur+" et pour adresse "+adresseUtilisateur;
26      return utilisateur;
27    }
28
29
30    public void addEnchere(Enchere enchere){
31      encheres.add(enchere);
32    }
33
34    public int getnumUtilisateur(){
35      return this.numUtilisateur;
36    }
37
```

```
38    public String getemailUtilisateur(){
39      return this.emailUtilisateur;
40    }
41
42    public String getnomUtilisateur(){
43      return this.nomUtilisateur;
44    }
45
46    public String getprenomUtilisateur(){
47      return this.prenomUtilisateur;
48    }
49
50    public String getadresseUtilisateur(){
51      return this.adresseUtilisateur;
52    }
53
54    public ArrayList<Enchere> getencheres(){
55      return this.encheres;
56    }
57
58    public void setemailUtilisateur(String emailUtilisateur){
59      updateUtilisateur(getnumUtilisateur(), emailUtilisateur, getnomUtilisateur(),
         getprenomUtilisateur(), getadresseUtilisateur());
60      this.emailUtilisateur = emailUtilisateur;
61    }
62
63    public void setnomUtilisateur(String nomUtilisateur){
64      updateUtilisateur(getnumUtilisateur(), getemailUtilisateur(), nomUtilisateur,
         getprenomUtilisateur(), getadresseUtilisateur());
65      this.nomUtilisateur = nomUtilisateur;
66    }
67
68    public void setprenomUtilisateur(String prenomUtilisateur){
69      updateUtilisateur(getnumUtilisateur(), getemailUtilisateur(),
        getnomUtilisateur(), prenomUtilisateur, getadresseUtilisateur());
70      this.prenomUtilisateur = prenomUtilisateur;
71    }
72
73    public void setadresseUtilisateur(String adresseUtilisateur){
74      updateUtilisateur(getnumUtilisateur(), getemailUtilisateur(),
        getnomUtilisateur(), getprenomUtilisateur(), adresseUtilisateur);
75      this.adresseUtilisateur = adresseUtilisateur;
76    }
77
78    public void updateUtilisateur(int numUtilisateur,String emailUtilisateur,
        String nomUtilisateur, String prenomUtilisateur, String adresseUtilisateur){
```

```
79      String[] update_value = {
80        "numUtilisateur",
81        "'"+numUtilisateur+"'",
82        "emailUtilisateur",
83        "'"+emailUtilisateur+"'",
84        "nomUtilisateur",
85        "'"+nomUtilisateur+"'",
86        "prenomUtilisateur",
87        "'"+prenomUtilisateur+"'",
88        "adresseUtilisateur",
89        "'"+adresseUtilisateur+"'",
90      };
91      this.bddaccess.update(table, update_value);
92    }
93
94    public void deleteUtilisateur(){
95      this.bddaccess.delete(this.table, "numUtilisateur="+String.valueOf(this.
        numUtilisateur));
96    }
97
98    public void insertUtilisateur(){
99      String[] column_value = {
100       "numUtilisateur",
101       "emailUtilisateur",
102       "nomUtilisateur",
103       "prenomUtilisateur",
104       "adresseUtilisateur",
105       String.valueOf(this.numUtilisateur),
106       "'"+this.emailUtilisateur+"'",
107       "'"+this.nomUtilisateur+"'",
108       "'"+this.prenomUtilisateur+"'",
109       "'"+this.adresseUtilisateur+"'"
110     };
111     this.bddaccess.insert(this.table, column_value);
112   }
113 }
```

java/Utilisateur.java

```
1  import java.util.Calendar;
2  import java.util.ArrayList;
3  import java.util.Date;
4  import java.text.SimpleDateFormat;
5  import java.text.ParseException;
6
```

```
7  public class Vente {
8
9    private int numVente;
10   private int numProduit;
11   private float prixVente;
12   private Boolean montantVente;
13   private Boolean revocableVente;
14   private Boolean dureelibreVente;
15   private Calendar datefinVente;
16   private Boolean encherirplusieurfoisVente;
17   ArrayList<Enchere> encheres;
18   private Bddaccess bddaccess;
19   private String table = "Vente";
20
21
22   public Vente(String numVente, String numProduit, String prixVente, String
       montantVente, String revocableVente, String dureelibreVente, String
       datefinVente, String encherirplusieurfoisVente)throws ParseException {
23     this.numVente = Integer.parseInt(numVente);
24     this.numProduit = Integer.parseInt(numProduit);
25     this.prixVente = Float.parseFloat(prixVente);
26     this.montantVente = !montantVente.equals("0");
27     this.revocableVente = !revocableVente.equals("0");
28     this.dureelibreVente = !dureelibreVente.equals("0");
29     SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
30     this.datefinVente = Calendar.getInstance();
31     this.datefinVente.setTime(sdf.parse(datefinVente));
32     this.encherirplusieurfoisVente = !encherirplusieurfoisVente.equals("0");
33     this.bddaccess = new Bddaccess();
34     this.encheres = new ArrayList<Enchere>();
35   }
36
37   public String stringVente(){
38     String montante = (montantVente)?"montante":"descendante";
39     String revocable = (revocableVente)?"n'est pas pas perte":"peut être à perte"
       ;
40     String libre = (dureelibreVente)?"n'est pas":"est";
41     String date = (!dureelibreVente)?": fini à la date suivante "+getdatefinVente
       ():"";
42     String enchere = (encherirplusieurfoisVente)?"pouvez":"ne pouvez pas";
43     String vente ="Cette vente numéro "+String.valueOf(numVente)+" démarre au
       prix de " +String.valueOf(prixVente)+
44     "  cette vente est "+ montante+ ", elle "+revocable+", elle "+libre+ " limité
        en temps"+
45     date+", vous "+enchere+" enchérir plusieurs fois.";
46     return vente;
```

```
47    }
48
49    public String stringVentemenu(){
50      String montante = (montantVente)?"montante":"descendante";
51      String revocable = (revocableVente)?"n'est pas pas perte":"peut être à perte"
        ;
52      String libre = (dureelibreVente)?"n'est pas":"est";
53      String date = ":<br/> fini à la date suivante "+getdatefinVente();
54      String enchere = (encherirplusieurfoisVente)?"pouvez":"ne pouvez pas";
55      String vente ="Cette vente démarre au prix de " +String.valueOf(prixVente)+
56      "  <br/> elle est "+ montante+ ", elle "+revocable+",<br/> elle "+libre+ "
        limité en temps"+
57      date+",<br/> vous "+enchere+" enchérir plusieurs fois.<br/><br/><br/>";
58      return vente;
59
60    }
61
62    private Boolean trouveUtilisateurdoublon(int num){
63      Boolean res = false;
64      for (int i=0; i<encheres.size(); i++){
65        res = res || encheres.get(i).getnumUtilisateur() == num;
66      }
67      return res;
68    }
69
70    public String boolEnchere(Enchere enchere, Produit produit){
71      String res = "";
72      if(!getdureelibreVente()&&ventefini()){
73        res += "La vente est a durée limité et est finie<br/><br/>";
74      }
75      if(enchere.getqteproduitEnchere()>produit.getstockProduit()){
76        res += "Le produit n'a pas assez de stock, stock du produit:"+ String.
        valueOf(produit.getstockProduit())+"<br/><br/>";
77      }
78      if (getrevocableVente() && enchere.getprixachatEnchere()<produit.
        getprixrevientProduit()) {
79        res += "L'enchère est inférieure à la prix de revient ("+produit.
        getprixrevientProduit()+") du produit et <br/>cette vente ne permet pas de
        vente à perte<br/><br/>";
80      }
81      if (enchere.getprixachatEnchere()<getprixVente()) {
82        res += "Vous ne pouvez pas enchérir en dessous <br/>du prix de vente ("+
        getprixVente()+")<br/><br/>";
83      }
84      if(!getencherirplusieurfoisVente()&&trouveUtilisateurdoublon(enchere.
        getnumUtilisateur())){
```

```
85        res += "Vous avez déjà enchéri sur cette offre et <br/>cette vente ne
      permet pas de double enchère<br/><br/>";
86      }
87      ArrayList<Enchere> backtrackenchere = backtrackenchere(produit.
      getstockProduit());
88      if (!verifStock( produit, enchere, backtrackenchere)) {
89        res += "Le prix de cette enchère n'est pas assez élevé, <br/>enchere(s)
      gagnante(s):<br/><br/>";
90        for (int i=0; i<backtrackenchere.size(); i++) {
91          res += backtrackenchere.get(i).stringEncheremenu();
92        }
93      }
94      return res;
95    }
96
97    public Boolean verifStock(Produit produit, Enchere enchere, ArrayList<Enchere>
      backtrackenchere){
98        int stock = produit.getstockProduit();
99        Boolean res = false;
100       for (int i = 0; i<backtrackenchere.size(); i++) {
101         stock-=backtrackenchere.get(i).getqteproduitEnchere();
102         if(enchere.getqteproduitEnchere()<= stock||(backtrackenchere.get(i).
      getprixachatEnchere()<enchere.getprixachatEnchere()))res=true;
103       }
104       if(backtrackenchere.size()==0)res=true;
105       return res;
106   }
107
108   public ArrayList<Enchere> backtrackenchere(int stock){
109     ArrayList<Enchere> res = new ArrayList<Enchere>();
110     if(encheres.size()!=0){
111       for (int i = encheres.size()-1; i>=0; i--) {
112         if(encheres.get(i).getqteproduitEnchere()<=stock){
113           stock -= encheres.get(i).getqteproduitEnchere();
114           res.add(encheres.get(i));
115         }
116       }
117     }
118     return res;
119   }
120
121   public void addEnchere(Enchere enchere){
122     encheres.add(enchere);
123   }
124
125   public Boolean ventefini(){
```

```
126       Date date=new java.util.Date();
127       Calendar calendar = Calendar.getInstance();
128       calendar.setTime(date);
129       return this.datefinVente.compareTo(calendar) == -1;
130     }
131
132     public ArrayList<Enchere> getencheres(){
133       return this.encheres;
134     }
135
136     public int getnumVente(){
137       return this.numVente;
138     }
139
140     public int getnumProduit(){
141       return this.numProduit;
142     }
143
144     public float getprixVente(){
145       return this.prixVente;
146     }
147
148     public Boolean getmontantVente(){
149       return this.montantVente;
150     }
151
152     public Boolean getrevocableVente(){
153       return this.revocableVente;
154     }
155
156     public Boolean getdureelibreVente(){
157       return this.dureelibreVente;
158     }
159
160     public String getdatefinVente(){
161       SimpleDateFormat format1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
162       String res = format1.format(this.datefinVente.getTime());
163       return res;
164     }
165
166     public Boolean getencherirplusieurfoisVente(){
167       return this.encherirplusieurfoisVente;
168     }
169
170     public void setprixVente(float prixVente){
171       updateVente(getnumVente(), getnumProduit(), prixVente, getmontantVente(),
```

```
        getrevocableVente(), getdureelibreVente(), getdatefinVente(),
        getencherirplusieurfoisVente());
172     this.prixVente = prixVente;
173   }
174
175   public void setmontantVente(Boolean montantVente){
176     updateVente(getnumVente(), getnumProduit(), getprixVente(), montantVente,
        getrevocableVente(), getdureelibreVente(), getdatefinVente(),
        getencherirplusieurfoisVente());
177     this.montantVente = montantVente;
178   }
179
180   public void setrevocableVente(Boolean revocableVente){
181     updateVente(getnumVente(), getnumProduit(), getprixVente(), getmontantVente()
        , revocableVente, getdureelibreVente(), getdatefinVente(),
        getencherirplusieurfoisVente());
182     this.revocableVente = revocableVente;
183   }
184
185   public void setdureelibreVente(Boolean dureelibreVente){
186     updateVente(getnumVente(), getnumProduit(), getprixVente(), getmontantVente()
        , getrevocableVente(), dureelibreVente, getdatefinVente(),
        getencherirplusieurfoisVente());
187     this.dureelibreVente = dureelibreVente;
188   }
189
190   public void setdatefinVente(String datefinVente)throws ParseException {
191     updateVente(getnumVente(), getnumProduit(), getprixVente(), getmontantVente()
        , getrevocableVente(), getdureelibreVente(), datefinVente,
        getencherirplusieurfoisVente());
192     SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
193     this.datefinVente.setTime(sdf.parse(datefinVente));
194   }
195
196   public void setencherirplusieurfoisVente(Boolean encherirplusieurfoisVente){
197     updateVente(getnumVente(), getnumProduit(), getprixVente(), getmontantVente()
        , getrevocableVente(), getdureelibreVente(), getdatefinVente(),
        encherirplusieurfoisVente);
198     this.encherirplusieurfoisVente = encherirplusieurfoisVente;
199   }
200
201   public void updateVente(int numVente, int numProduit, float prixVente, Boolean
        montantVente, Boolean revocableVente, Boolean dureelibreVente, String
        datefinVente, Boolean encherirplusieurfoisVente){
202     String[] update_value = {
203       "numVente",
```

```
204           String.valueOf(numVente),
205           "numProduit",
206           String.valueOf(numProduit),
207           "prixVente",
208           String.valueOf(prixVente),
209           "montantVente",
210           (montantVente)?"1":"0",
211           "revocableVente",
212           (revocableVente)?"1":"0",
213           "dureelibreVente",
214           (dureelibreVente)?"1":"0",
215           "datefinVente",
216           "TO_DATE('"+getdatefinVente()+"','YYYY-MM-DD HH24:MI:SS')",
217           "encherirplusieurfoisVente",
218           (encherirplusieurfoisVente)?"1":"0"
219         };
220         this.bddaccess.update(table, update_value);
221       }
222
223       public void deleteVente(){
224         this.bddaccess.delete(this.table, "numVente="+String.valueOf(this.numVente));
225       }
226
227       public void insertVente(){
228         String[] column_value = {
229           "numVente",
230           "numProduit",
231           "prixVente",
232           "montantVente",
233           "revocableVente",
234           "dureelibreVente",
235           "datefinVente",
236           "encherirplusieurfoisVente",
237           String.valueOf(this.numVente),
238           String.valueOf(this.numProduit),
239           String.valueOf(this.prixVente),
240           (this.montantVente)?"1":"0",
241           (this.revocableVente)?"1":"0",
242           (this.dureelibreVente)?"1":"0",
243           "TO_DATE('"+getdatefinVente()+"','YYYY-MM-DD HH24:MI:SS')",
244           (this.encherirplusieurfoisVente)?"1":"0"
245         };
246         this.bddaccess.insert(this.table, column_value);
247       }
248     }
```

java/Vente.java

Références

[1] Site officiel codejava. https ://www.codejava.net/java-se/swing/jcombobox-basic-tutorial-and-examples.

[2] Site officiel oracle. https ://www.oracle.com/fr/database/technologies/appdev/jdbc.html.

[3] Site officiel waytolearnx. https ://waytolearnx.com/2020/06/tutoriels-java.html.