


Sommaire

1. INTRODUCTION.....	2
1.1. LES PROBLEMES RENCONTRES	3
1.2. LES SOLUTIONS	3
2. LE CODE JAVA.....	4
2.1. COPIE INTEGRALE DU CODE COMMENTE QUI FONCTIONNE.....	4
3. RESULTAT UML.....	12
4. RESULTAT JAVA	15
5. CONCLUSION	17


	Mini-Projet JAVA 2D	Version 1.0
	09/06/2020	

1. INTRODUCTION

Durant cette dernière séance de TP, nous (Rémy SENAY & Majdi BEN SALEM) avons eu pour but principal la création d'une interface graphique dynamique.

Cette dernière séance de TP était la plus complète et la riche en enseignement. Nous redécouvrons toutes les notions que nous avons appris en Java mais sous une autre forme, celle d'un mini-projet.

Le TP commence d'abord par les diagrammes UML. Ceux-ci nous permettent d'avoir une vision globale du projet, des classes et des objets utilisés. Ensuite pour la partie programmation, un cahier des charges a été fixé, mais par la suite nous sommes libres d'utiliser la manière que l'on veut pour arriver au résultat.

	Mini-Projet JAVA 2D	Version 1.0
	09/06/2020	

1.1. Les problèmes rencontrés

Partie 1 :

- Afficher les résultats sous forme de JTable qui seraient dynamique
- Convertir une ArrayList en Tableau 2 dimensions
- Travailler entre plusieurs classes :
 - o Savoir où et comment instancier les objets
 - o Les relations entre les différents objets
 - o Le positionnement de l'instanciation
- Création de plusieurs Panneau en utilisant uniquement un constructeur sans argument

Partie 2 :

- Positionnement de l'instanciation
- La différence entre l'utilisation des Setters et d'un constructeur avec comme argument toute les caractéristiques des Setters

1.2. Les solutions trouvées

Partie 1 :

- Utilisation d'un JPanel au lieu d'un Jtable avec GridLayout afin d'avoir un positionnement similaire à une JTable
- Nous avons donc pas convertit l'ArrayList en Tableau 2 dimensions.
- Nous avons travaillé avec 4 classes différentes au début (sans compter les A/B/IProduit..) :
 Nous avons 1 classe Listener, 1 classe Main, 1 classe 1ere fenêtre et 1 classe 2^e fenêtre. Cela devenait trop compliquer pour faire les liaisons entre les différents objets. Nous avons donc tout repris depuis le début et créer 1 seul classe principal (fille de JFrame et implémentant le listener) comportant le constructeur sans arguments qui construira notre fenetre de base.
- Ainsi les 4 classes sont devenues les 4 méthodes de la classe principales. Cela est devenu vraiment beaucoup plus simple. Toute les variable encapsulé private et déclaré dans la classe.
- Nous avons utilisé le constructeur sans argument, grâce à l'héritage de JFrame, pour créer la fenêtre de base. Puis nous avons déclarés un autre JFrame associé à son JPanel en private, qui correspondra à la fenêtre « résultat ». Puis nous affichons cette fenêtre lorsque le bouton « Finish » est pressé.
- Nous souhaitions utiliser le constructeur pour créer toute nos fenetre. Cependant cela ne fonctionnait pas car l'une fonctionnait au détriment de l'autre.
- De plus nous souhaitions partager les mêmes objets (JButtons, JLabel..) entre les 2 fenêtres mais cela n'est pas possible. Il aurait fallu recrée d'autres objets.

Partie 2 :

- Pour plus de sécurité nous avons utilisé la méthode vu en cours pour nos objets :
`Class objet = new Class() ;`
- Ensuite nous travaillons avec ces mêmes objets uniquement dans la classe principal.

2. LE CODE JAVA

2.1. Copie intégrale du code commenté qui fonctionne

PARTIE 1)

```
package metier;
//Class ProduitA
public class ProduitA implements IProduit{

    private String nom;
    private String ville;
    private String qualite;
    private int prixUnitaire;
    private int quantite;

    public int calculPrix(int quantite, int prix) {

        return(quantite*prix);
    }

    public int calculPrix(int quantite, int prix, int txReduction) {return(0);}

    public ProduitA(String nom, String ville, String qualite, int prixUnitaire, int quantite) {
        super();
        this.nom = nom;
        this.ville = ville;
        this.qualite = qualite;
        this.prixUnitaire = prixUnitaire;
        this.quantite = quantite;
    }

    public String getNom() {
        return nom;
    }

    public String getVille() {
        return ville;
    }

    public String getQualite() {
        return qualite;
    }

    public int getPrixUnitaire() {
        return prixUnitaire;
    }

    public int getQuantite() {
        return quantite;
    }
}
```

```
package metier;
//Class ProduitB

public class ProduitB implements IProduit{

    private String nom;
    private String ville;
    private int prixUnitaire;
    private int quantite;
    private int txReduction;

    public int calculPrix(int quantite, int prix, int txReduction) {

        return(int)(quantite*prixUnitaire*(1-((float)txReduction/100)));
    }

    public int calculPrix(int quantite, int prix) {return(0);}
    public ProduitB(String nom, String ville, int txReduction, int prixUnitaire, int quantite) {
        super();
        this.nom = nom;
        this.ville = ville;
        this.prixUnitaire = prixUnitaire;
        this.quantite = quantite;
        this.txReduction = txReduction;
    }

    public String getNom() {
        return nom;
    }
    public String getVille() {
        return ville;
    }
    public int getTxReduction() {
        return txReduction;
    }
    public int getPrixUnitaire() {
        return prixUnitaire;
    }
    public int getQuantite() {
        return quantite;
    }
}
```

//Interface IProduit

```
package metier;
public interface IProduit {
    public int calculPrix(int quantite, int prix);
    public int calculPrix(int quantite, int prix, int txReduction);
}
```

//Class Principal

```
package metier;
import javax.swing.*;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

public class MoteurGraphique extends JFrame implements ActionListener {
    /* La classe Moteur Graphique va contenir tout notre programme hormis les Produits
    * Nous heritons de la classe mère JFrame pour concevoir notre fenetre principal
    * Nous héritons aussi des fonctionnalités de la classe ActionListener
    * Donc travail sur 1 seul classe principal -> toute nos variables sont declarer en private
    * On a enlevé tout les setter des classes Produits car on ne les utilisaient pas
    */

    private boolean flag = true;
    // Flag permettant de savoir si on ajoute un produitA ou B sur la 1ere fenetre

    private static final long serialVersionUID = 1L;
    // Permet de conserver son programme meme si modification

    private JTextField text1 = new JTextField("");
    private JTextField text2 = new JTextField("");
    private JTextField text3 = new JTextField("");
    private JTextField text4 = new JTextField("");
    private JTextField text5 = new JTextField("");
    // Les JTextFields sont les champs de texte permettant d'ecrire

    private JLabel label1 = new JLabel("Nom");
    private JLabel label2 = new JLabel("Ville");
    private JLabel label3 = new JLabel("Qualite");
    private JLabel label4 = new JLabel("Prix Unitaire");
    private JLabel label5 = new JLabel("Quantite");
    // Nous utilisons les labels comme des images contenant des mots pouvant etre afficher

    private JButton bouton1 = new JButton("Ajouter");
    // Ajoute le produit rentrer a la liste des resultats

    private JButton bouton2 = new JButton("Finish");
    // Affiche le resultat

    private JButton bouton3 = new JButton("Changer");
    // Change l'affichage pour rentrer un Produit B

    private JButton bouton4 = new JButton("Quitter");
    // Quitte le programme

    private ArrayList<ProduitA> listA = new ArrayList<ProduitA>();
    private ArrayList<ProduitB> listB = new ArrayList<ProduitB>();
    // Nos 2 tableaux dynamiques contenant les produits rentrés

    private JPanel panneau = new JPanel();
    private JFrame fen = new JFrame();
    // Fenêtre et panneau que nous utiliserons pour le résultat
    // Car nous n'avons pas réussi a construire 2 fenêtres différente a partir d'1 seul constructeur

    public MoteurGraphique() {

        this.setTitle("Fenetre Produit A"); // Titre de la fenêtre
        this.setSize(300,350); // Taille de la fenêtre
        this.setLocationRelativeTo(null); // Centre la fenêtre sur l'écran
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // ferme l'appli quand on appuie sur la croix

        //this.getContentPane() permet de faire référence au panneau par défaut attaché à notre fenêtre constructeur
        this.getContentPane().setLayout(new GridLayout(7,2));
        // Fenêtre repartie en 7 lignes et 2 colonnes (1 pour le label d'info et l'autre le champ texte

        this.getContentPane().add(label1); this.getContentPane().add(text1);
        this.getContentPane().add(label2); this.getContentPane().add(text2);
        this.getContentPane().add(label3); this.getContentPane().add(text3);
```

```

this.getContentPane().add(label4); this.getContentPane().add(text4);
this.getContentPane().add(label5); this.getContentPane().add(text5);
this.getContentPane().add(bouton1);this.getContentPane().add(bouton2);
this.getContentPane().add(bouton3);this.getContentPane().add(bouton4);
// Ajout des différents objets selon l'ordre du GridLayout

bouton1.addActionListener(this);
bouton2.addActionListener(this);
bouton3.addActionListener(this);
bouton4.addActionListener(this);
// Ajout des listener sur les boutons

this.setVisible(true);
//Rend la fenetre visible
//Nous n'utilisons pas pack() car sinon la fenetre est petite par default malgre le dimmensionnement
}

public void actionPerformed(ActionEvent evenement) {
    // Notre methode listener

    JButton boutonClique = (JButton) evenement.getSource();
    String texteEcrit = boutonClique.getText();
    // Recuperation de l'info du bouton cliqué

    if(texteEcrit.equals("Ajouter")) {
        // On ajoute le Produit correspondant dans son tableau dynamique
        // dont les valeurs sont prélevé dans les textFields

        if(flag == true) {

            listA.add(new
ProduitA(text1.getText(),text2.getText(),text3.getText(),Integer.parseInt(text4.getText()),Integer.parseInt(text5.getText())));
        }

        else {

            listB.add(new
ProduitB(text1.getText(),text2.getText(),Integer.parseInt(text3.getText()),Integer.parseInt(text4.getText()),Integer.parseInt(text5.getText())));
        }

        text1.setText(""); // Vide les TextField
        text2.setText("");
        text3.setText("");
        text4.setText("");
        text5.setText("");
    }

    else if(texteEcrit.equals("Finish")) {
        // On lance la 2e interface graphique comportant les résultats
        resultat();
    }

    else if(texteEcrit.equals("Changer")) {
        // On change le type de produit à rentré

        if(flag == true) { // ProduitA to ProduitB
            this.setTitle("Fenetre Produit B");
            this.label3.setText("TxReduction");
            flag = false;
        }

        else { //ProduitB to ProduitA
            this.setTitle("Fenetre Produit A");
            this.label3.setText("Qualite");
            flag = true;
        }
    }

    else if(texteEcrit.equals("Quitter")) {
        System.exit(1);
    }
}

```

```

public static void main(String[] args) {
    // Constructeur sans parametre instanciant notre JFrame de depart
    new MoteurGraphique();
}

public void resultat() {

    // Configuration initial de la nouvelle fenetre
    fen.setTitle("Resultat");
    fen.setSize(400,500);
    fen.setLocationRelativeTo(null);
    fen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Activation du panneau sur la fenetre
    fen.setContentPane(panneau);

    panneau.setLayout(new GridLayout(listA.size()+listB.size()+1,7));
    panneau.add(new JLabel("Nom"));
    panneau.add(new JLabel("Ville"));
    panneau.add(new JLabel("Qualite"));
    panneau.add(new JLabel("TxReduction"));
    panneau.add(new JLabel("Prix_Unitaire"));
    panneau.add(new JLabel("Quantite"));
    panneau.add(new JLabel("PrixTotale"));

    int i=0;

    for(i=0;i<listA.size();i++) {
        // On ajoute sur le panneau des labels qui correspondent à des cases qui contiennent des informations
        // 6 labels pour 6 parametres par ligne
        // On utilise les Getters et String.valueOf pour un affichage en String

        panneau.add(new JLabel(listA.get(i).getNom()));
        panneau.add(new JLabel(listA.get(i).getVille()));
        panneau.add(new JLabel(listA.get(i).getQualite()));
        panneau.add(new JLabel(" "));
        panneau.add(new JLabel(String.valueOf(listA.get(i).getPrixUnitaire())));
        panneau.add(new JLabel(String.valueOf(listA.get(i).getQuantite())));
        panneau.add(new JLabel(String.valueOf(listA.get(i).calculPrix(listA.get(i).getQuantite(), listA.get(i).getPrixUnitaire()))));
    }

    for(i=0;i<listB.size();i++) {

        panneau.add(new JLabel(listB.get(i).getNom()));
        panneau.add(new JLabel(listB.get(i).getVille()));
        panneau.add(new JLabel(" "));
        panneau.add(new JLabel(String.valueOf(listB.get(i).getTxReduction())));
        panneau.add(new JLabel(String.valueOf(listB.get(i).getPrixUnitaire())));
        panneau.add(new JLabel(String.valueOf(listB.get(i).getQuantite())));
        panneau.add(new JLabel(String.valueOf(listB.get(i).calculPrix(listB.get(i).getQuantite(),
listB.get(i).getPrixUnitaire(),listB.get(i).getTxReduction()))));
    }

    fen.setVisible(true);
    this.setVisible(false);
    fen.pack();
}
}

```


PARTIE 2)

Package Metier :

Le package métier est le même qu'à la Partie 1 Java, hormis la Classe Client que nous avons créer :

// Class Client

```
package metier;
public class Client {
    private String nom;
    private String prenom;
    private String ville;
    private int age;
    public Client() {
    }
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getPrenom() {
        return prenom;
    }
    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }
    public String getVille() {
        return ville;
    }
    public void setVille(String ville) {
        this.ville = ville;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

Package Dao :

// Class ClientDao

```
package dao;
import java.util.ArrayList;
import java.util.Scanner;
import metier.Client;

public class ClientDao {

    private Scanner scan = new Scanner(System.in);
    private ArrayList<Client> tab = new ArrayList<Client>();

    public ArrayList<Client> createClient() {
        /* Nous sommes restés confus sur cette methode. En effet elle s'appelle createClient et non
        createClients ou createTabClient Logiquement elle devraient renvoyé un client . Cependant la consigne demande de
        retourner un tableau. Cette fonction retourne donc le tableau de client que nous avons crée*/
        System.out.println("Rentrez le Nombre de client");
        int i=0, indice;
        indice = scan.nextInt();
        for(i=0;i<indice;i++) {

            Client cli = new Client();
            // On crée une nouvelle instance a chaque rebouclage car si on instancie dans la classe
            // chaque valeur rentrer vas écraser toute les anciennes

            System.out.println("Rentrez le Nom du "+(i+1)+"e client");
            cli.setNom(scan.next());
            System.out.println("Rentrez le Prénom");
            cli.setPrenom(scan.next());
            System.out.println("Rentrez l'age");
            cli.setAge(scan.nextInt());
            System.out.println("Rentrez la ville");
            cli.setVille(scan.next());

            tab.add(cli);
        }

        return(tab);
    }

    public Client getClientParMC(String mc, ArrayList<Client> list) {
        // Fonction cherchant le MC dans la liste
        // Renvoi Null si rien trouver, sinon renvoie le client
        for(Client cli : list) {
            if(cli.getNom().equals(mc)) {
                return(cli);
            }
        }
        return(null);
    }

    public ArrayList<Client >deleteClientParMC(String mc, ArrayList<Client> list) {
        // Fonction supprimant le MC de la list puis la renvoie
        for(Client cli : list) {
            if(cli.getNom().equals(mc)) {
                list.remove(cli);
            }
        }
        return(list);
    }
}
```

Package Test :

// Class StudentDao

```
package test;
import java.util.ArrayList;
import java.util.Scanner;
import dao.ClientDao;
import metier.Client;

public class StudentDao {

    private static ArrayList<Client> list = new ArrayList<Client>();
    private static ClientDao dao = new ClientDao();
    private static String motcle;
    static Scanner scan = new Scanner(System.in);

    public static void main(String[] args) {

        // On utilise la methode Dao createClient pour créer notre liste de client
        list = dao.createClient();

        System.out.println("Voici la liste des clients");

        // Boucle each parcourant le tableau de client
        for(Client cl : list) {
            System.out.println(cl.getNom()+" "+cl.getPrenom()+" "+cl.getAge()+"ans de "+cl.getVille());
        }

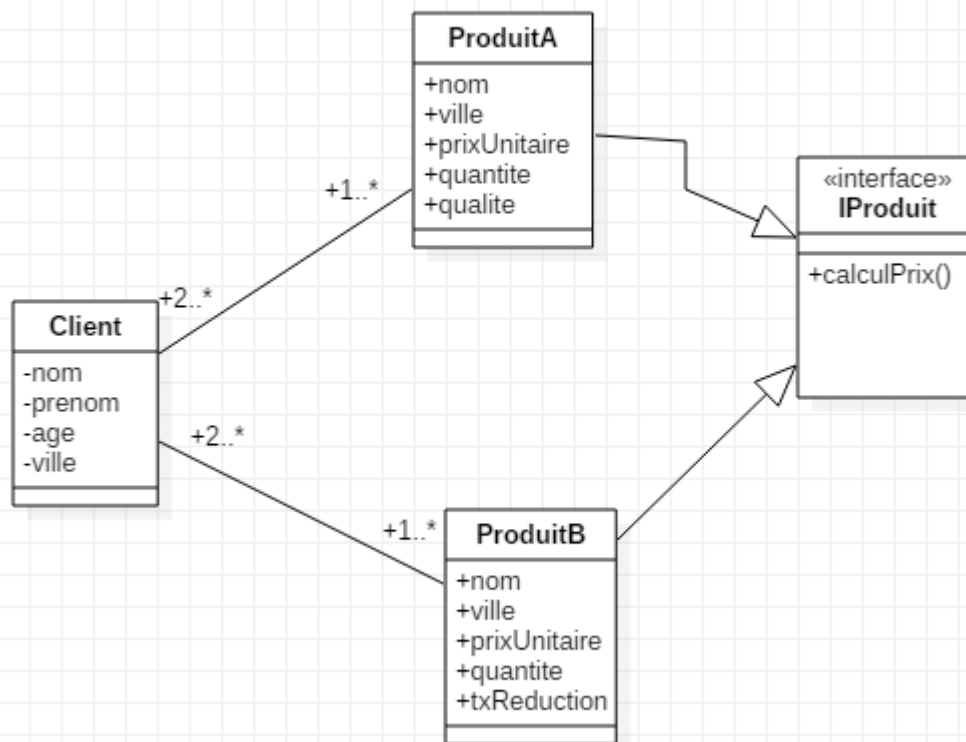
        System.out.println("Rentrez le mot clé");
        motcle = scan.nextLine();

        // On utilise la methode getClient qui renvoie soit le client trouvé soit NULL si rien trouver
        if(dao.getClientParMC(motcle, list)==null) {
            System.out.println("Pas de correspondance trouver");
        }

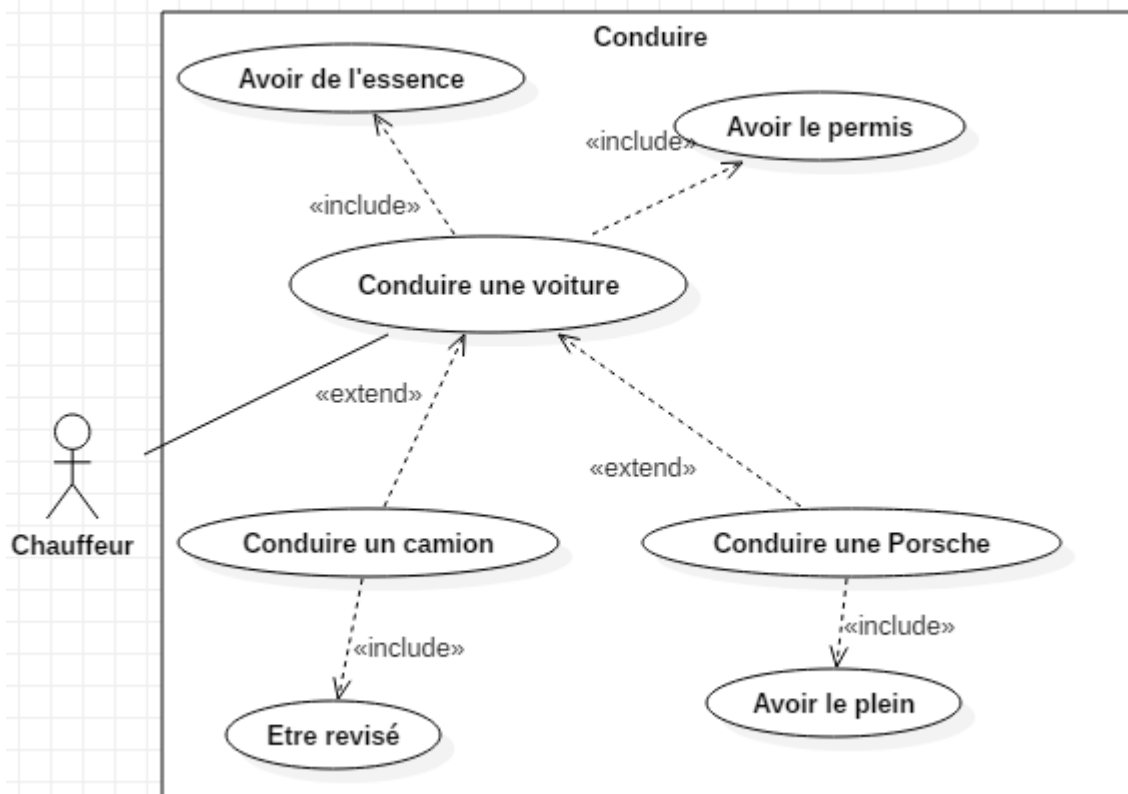
        else {
            System.out.println("Nous avons trouver un client avec le meme nom \n Nous allons donc
supprimer ce client");
            list = dao.deleteClientParMC(motcle, list); // Methode supprimant le client MC de la list
            System.out.println("Voici la nouvelle liste des clients");
            for(Client cl : list) {
                System.out.println(cl.getNom()+" "+cl.getPrenom()+" "+cl.getAge()+"ans de
"+cl.getVille());
            }
        }
    }
}
```

3. RESULTAT UML

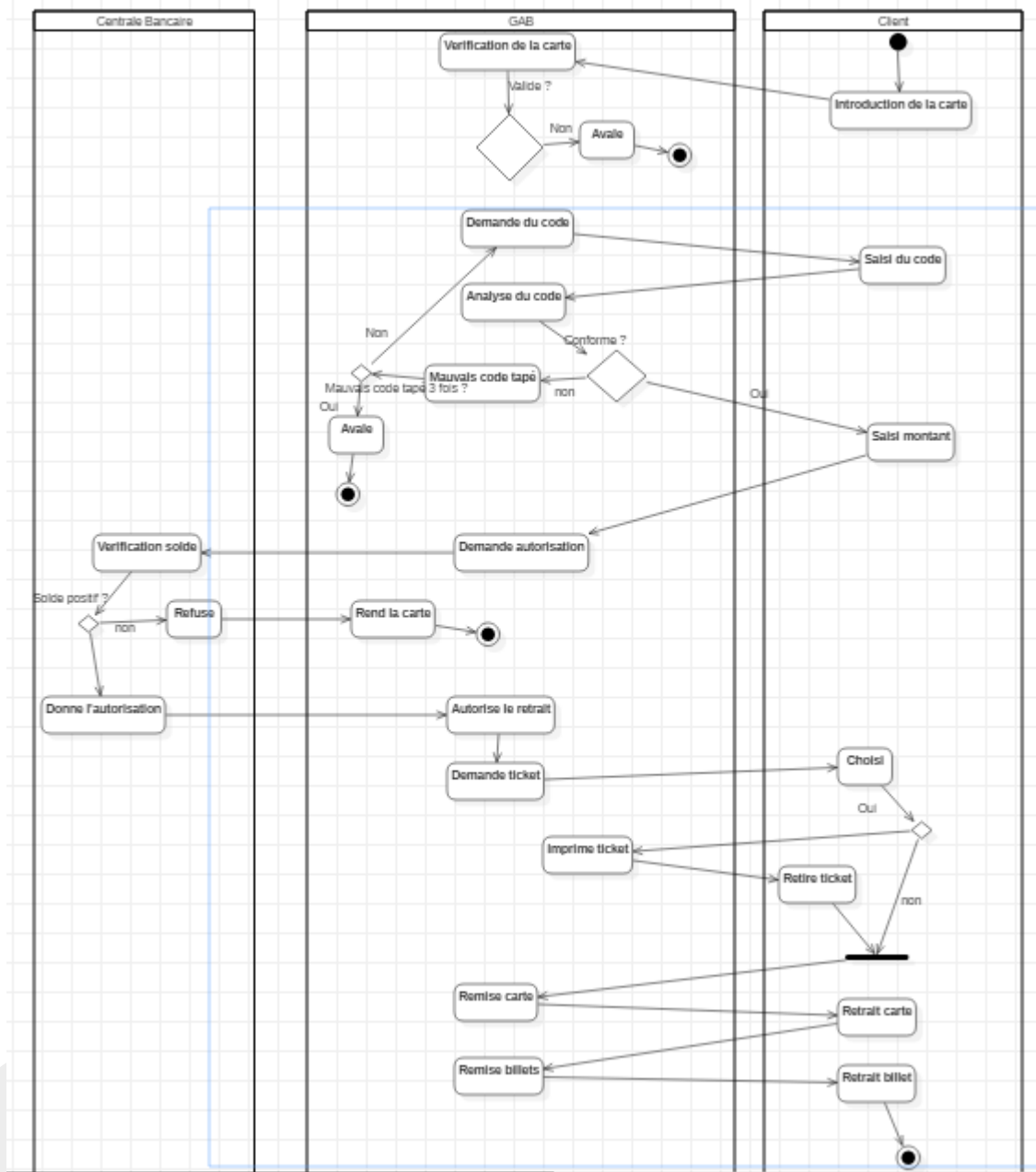
Partie 1 :



Partie 2 :



Partie 3 :



4. RESULTAT JAVA

Partie 1 :

Fenetre Produit B

Nom	Cafe
Ville	Perou
TxReduction	50
PrixUnitaire	2
Quantite	1
Ajouter	
Finish	
Changer	
Quitter	

Fenetre Produit A

Nom	Moutarde
Ville	Dijon
Qualite	AOC
PrixUnitaire	2
Quantite	1
Ajouter	
Finish	
Changer	
Quitter	

Resultat

Nom	Ville	Qualite	TxReduction	Prix_Unitaire	Quantite	PrixTotale
Moutarde	Dijon	AOC		2	1	2
Cafe	Perou		50	2	1	1
Orange	Espagne		20	2	10	16

Partie 2 :



Console

<terminated> StudentDao [Java Application] C:\Program Files\Java\jre1.8.0_45\

Rentrez le Nombre de client

3

Rentrez le Nom du 1e client

Kimpembe

Rentrez le Prénom

Presnel

Rentrez l'age

26

Rentrez la ville

Paris

Rentrez le Nom du 2e client

Dubois

Rentrez le Prénom

Olivier

Rentrez l'age

25

Rentrez la ville

Poudlard

Rentrez le Nom du 3e client

Ballo-Toure

Rentrez le Prénom

Fode

Rentrez l'age

21

Rentrez la ville

Monaco

Voici la liste des clients

Kimpembe Presnel 26ans de Paris

Dubois Olivier 25ans de Poudlard

Ballo-Toure Fode 21ans de Monaco

Rentrez le mot clé

Dubois


Nous avons trouver un client avec le meme nom

Nous allons donc supprimer ce client

Voici la nouvelle liste des clients

Kimpembe Presnel 26ans de Paris

Ballo-Toure Fode 21ans de Monaco

	Mini-Projet JAVA 2D	Version 1.0
	09/06/2020	

5. CONCLUSION

Cette dernière séance de TP nous a permis de mettre en application direct les notions apprises en cours. Tout d'abord grâce aux concepts de classe, méthode et attributs mais aussi en termes d'héritage et d'instanciation. Grâce à cette application, nous avons approfondis nos compétences en Java et avons acquis des notions en matière de Programmation Orientée Objet.

En effet ce projet a été le plus riche en enseignements et en termes d'organisations. Contrairement aux autres TP, une réflexion a dû être mis en place (par l'UML) pour avoir une vision global du projet avant d'entamer la programmation. De plus le travail en binôme était un réel atout en terme de concentration et de créativité pour notre programme.

Enfin ce projet nous a permis d'approfondir nos connaissances en développement software. C'est notamment lors d'un projet actuel en entreprise que je vais pouvoir appliquer ces connaissances dans le but de développer un IHM.