Licence 3 - Informatique

Génie logiciel

TP7 Le design pattern visitor

- Mohammed BENAOU

Responsable :
Pr. Armelle Prigent

17 Mars 2017

Contents

1 Introduction

L'objectif de ce TP est de mieux illustrer la mise en œuvre des modèles de conception, Les patterns de construction déterminent comment faire l'instanciation et la configuration des classes et des objets. Ces patterns permettent également d'encapsuler les classes concrètes et de rendre indépendant la façon dont les objets sont créés.

2 Exercice 1

il suffit donc de créer un nouveau Visiteur avant de surcharger les méthodes permettant de visiter les différents objets de la hierarchie.

2.1 Partie 1

dans un premier temps il faut implementer le code correspondant au diagramme de classe Alors on a :
-

2.1.1 Interface Figure

```
package graphique;

import com.sun.org.apache.bcel.internal.classfile.Visitor;

/**
 *
 * @author Mohammed
 */
interface Figure {
    public void accept (VisitorTest v);
}
```

-

2.1.2 L'Interface VisitorTest

```
package graphique;

/**
 *
```

```
 * @author Mohammed
 */
public interface VisitorTest { // l'interface Visitor
     void visit(Circle c);
     void visit(Rectangle r);
     void visit(Composite c);

}
```

-

2.1.3 LA Classe Circle

```
package graphique;

import com.sun.org.apache.bcel.internal.classfile.Visitor;
/**
 *
 * @author Mohammed
 */
public class Circle implements Figure {
    public double x,y,r;
    public Circle (double r,double x, double y){
        this.x=x;
        this.y=y;
        this.r=r;
    }
    public Circle() {
    }
    public void accept (VisitorTest monVisiteur){
        monVisiteur.visit(this);
         }
    public String Circlee(){
        return "Circle";
    }
    public double area(){
     return 3.14*r*r;
    }
    public void prettyPrint(){
        System.out.println("la position verticale "+y);
        System.out.println("la position horizontale "+x);
        System.out.println("la valeur du rayon est "+r);
    }
}
```

-

2.1.4 La Classe Rectangle

```
package graphique;

import com.sun.org.apache.bcel.internal.classfile.Visitor;

/**
 *
 * @author Mohammed
 */
public class Rectangle implements Figure {
    public double x,y,width,height;

    public Rectangle(double x, double y, double width,double height){
        this.height=height;
        this.width=width;
        this.x=x;
        this.y=y;
    }
    public Rectangle() {
    }
    public String Rectanglee(){
        return "Rectangle";
    }
    public void accept (VisitorTest monVisiteur){
        monVisiteur.visit(this);
         }
    public double area(){
        return width*height;
    }
    public void prettyPrint(){
        System.out.println("La position verticale "+y);
        System.out.println("La position horizontale "+x);
        System.out.println("La largeur est "+width);
        System.out.println("La hauteur est "+height);
    }
}
```

-

2.1.5 La Classe Composition

```
package graphique;
```

```
import java.util.ArrayList;

/**
 *
 * @author Mohammed
 */
public class Composite implements Figure {
   public  ArrayList <Figure>ls =new ArrayList<>();
    public void accept (VisitorTest monVisiteur){
        monVisiteur.visit(this);
    }
    public Composite(){

    }
    public void area(){

    }
    public String Compositee(){
        return "Composite";
    }
    public void addChild(Figure F){
        ls.add(F);
    }

}
```

-

2.1.6 La Classe Square

```
package graphique;

/**
 *
 * @author Mohammed
 */
public class Square extends Rectangle {

    public Square(double x, double y, double width, double height) {
        super(x, y, width, height);
    }

}
```

-

2.1.7 La Classe visitorDraw

```
package graphique;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Mohammed
 */
public class visitorDraw implements VisitorTest {

    @Override
    public void visit(Circle c) {
        System.out.println("Le dessin de la " + c.Circlee());
        drawCircle(c);
    }
    @Override
    public void visit(Rectangle r) {
        System.out.println("Le dessin du " + r.Rectanglee());
        drawRectangle(r);
    }
    @Override
    public void visit(Composite c) {
        System.out.println("Le dessin de la " + c.Compositee());
    }
    public void drawRectangle(Rectangle rec) {
        BufferedWriter bufWriter = null;
        FileWriter fileWriter = null;
        try {
            try {
                fileWriter = new FileWriter("Rictangle.html", true);
            } catch (IOException ex) {
                Logger.getLogger(visitorDraw.class.getName()).log(Level.SEVERE, null, ex);
            }
            bufWriter = new BufferedWriter(fileWriter);
            String chaineCaracteres = "<!DOCTYPE HTML>\n"
                    + "<html>\n"
                    + "  <head>\n"
                    + "    <style>\n"
                    + "      body {\n"
                    + "        margin: 0px;\n"
```

```
                + "        padding: 0px;\n"
                + "      }\n"
                + "    </style>\n"
                + "  </head>\n"
                + "  <body>\n"
                + "    <canvas id=\"myCanvas\" width=\"578\" height=\"200\"></canvas>\n'
                + "    <script>\n"
                + "      var canvas = document.getElementById('myCanvas');\n"
                + "      var context = canvas.getContext('2d');\n"
                + "\n"
                + "      context.beginPath();\n"
                + "      context.rect(" + rec.x + "," + rec.y + "," + rec.width + "," +
                + "      context.fillStyle = 'yellow';\n"
                + "      context.fill();\n"
                + "      context.lineWidth = 7;\n"
                + "      context.strokeStyle = 'black';\n"
                + "      context.stroke();\n"
                + "    </script>\n"
                + "  </body>\n"
                + "</html>      ";
            bufWriter.write(chaineCaracteres + "\n");
            bufWriter.close();
            fileWriter.close();
        } catch (IOException ex) {
            System.out.println("Problème lors de la sauvegarde");
        }
    }

    public void drawCircle(Circle cir) {
        BufferedWriter bufWriter = null;
        FileWriter fileWriter = null;
        try {
            try {
                fileWriter = new FileWriter("Circle.html", true);
            } catch (IOException ex) {
                Logger.getLogger(visitorDraw.class.getName()).log(Level.SEVERE, null, ex);
            }
            bufWriter = new BufferedWriter(fileWriter);
            String chaineCaracteres = "<!DOCTYPE HTML>\n"
                    + "<html>\n"
                    + "  <head>\n"
                    + "    <style>\n"
                    + "      body {\n"
                    + "        margin: 0px;\n"
                    + "        padding: 0px;\n"
                    + "      }\n"
```

```
                + "    </style>\n"
                + "  </head>\n"
                + "  <body>\n"
                + "    <canvas id=\"myCanvas\" width=\"578\" height=\"200\"></canvas>\n'
                + "    <script>\n"
                + "      var canvas = document.getElementById('myCanvas');\n"
                + "      var context = canvas.getContext('2d');\n"
                + "      var centerX = " + cir.x + ";\n"
                + "      var centerY = " + cir.y + ";\n"
                + "      var radius  = "  + cir.r + ";\n"
                + "\n"
                + "      context.beginPath();\n"
                + "      context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);\n
                + "      context.fillStyle = 'green';\n"
                + "      context.fill();\n"
                + "      context.lineWidth = 5;\n"
                + "      context.strokeStyle = '#003300';\n"
                + "      context.stroke();\n"
                + "    </script>\n"
                + "  </body>\n"
                + "</html>";
            bufWriter.write(chaineCaracteres + "\n");
            bufWriter.close();
            fileWriter.close();
        } catch (IOException ex) {
            System.out.println("Problème lors de la sauvegarde");
        }
    }

    public void drawComposite() {
        BufferedWriter bufWriter = null;
        FileWriter fileWriter = null;
        try {
            try {
                fileWriter = new FileWriter("Composite.html", true);
            } catch (IOException ex) {
                Logger.getLogger(visitorDraw.class.getName()).log(Level.SEVERE, null, ex);
            }
            bufWriter = new BufferedWriter(fileWriter);
            String chaineCaracteres = "<!DOCTYPE HTML>\n"
                    + "<html>\n"
                    + "  <head>\n"
                    + "    <style>\n"
                    + "      body {\n"
                    + "        margin: 0px;\n"
                    + "        padding: 0px;\n"
```

```
+ "      } \n"
+ "    </style>\n"
+ "  </head>\n"
+ "  <body>\n"
+ "    <canvas id=\"myCanvas\" width=\"578\" height=\"430\"></canvas>\n'
+ "    <canvas id=\"tempCanvas\" width=\"578\" height=\"430\" style=\"di
+ "    <script>\n"
+ "      var canvas = document.getElementById('myCanvas');\n"
+ "      var context = canvas.getContext('2d');\n"
+ "      var tempCanvas = document.getElementById('tempCanvas');\n"
+ "      var tempContext = tempCanvas.getContext('2d');\n"
+ "\n"
+ "      var squareWidth = 55;\n"
+ "      var circleRadius = 35;\n"
+ "      var shapeOffset = 50;\n"
+ "      var operationOffset = 150;\n"
+ "      var arr = [];\n"
+ "\n"
+ "      arr.push('source-atop');\n"
+ "      arr.push('source-in');\n"
+ "      arr.push('source-out');\n"
+ "      arr.push('source-over');\n"
+ "      arr.push('destination-atop');\n"
+ "      arr.push('destination-in');\n"
+ "      arr.push('destination-out');\n"
+ "      arr.push('destination-over');\n"
+ "      arr.push('lighter');\n"
+ "      arr.push('darker');\n"
+ "      arr.push('xor');\n"
+ "      arr.push('copy');\n"
+ "\n"
+ "      // translate context to add 10px padding\n"
+ "      context.translate(10, 10);      // draw each of the operations\
+ "      for(var n = 0; n < arr.length; n++) {\n"
+ "        var thisOperation = arr[n];\n"
+ "\n"
+ "        tempContext.save();\n"
+ "        \n"
+ "        // clear temp context\n"
+ "        tempContext.clearRect(0, 0, canvas.width, canvas.height);\n"
+ "\n"
+ "        // draw rectangle (destination)\n"
+ "        tempContext.beginPath();\n"
+ "        tempContext.rect(0, 0, squareWidth, squareWidth);\n"
+ "        tempContext.fillStyle = 'blue';\n"
+ "        tempContext.fill();\n"
```

```
                    + "\n"
                    + "        // set global composite\n"
                    + "        tempContext.globalCompositeOperation = thisOperation;\n"
                    + "\n"
                    + "        // draw circle (source)\n"
                    + "        tempContext.beginPath();\n"
                    + "        tempContext.arc(shapeOffset, shapeOffset, circleRadius, 0, 2
                    + "        tempContext.fillStyle = 'red';\n"
                    + "        tempContext.fill();\n"
                    + "        tempContext.restore();\n"
                    + "\n"
                    + "        // draw text\n"
                    + "        tempContext.font = '10pt Verdana';\n"
                    + "        tempContext.fillStyle = 'black';\n"
                    + "        tempContext.fillText(thisOperation, 0, squareWidth + 45);\n"
                    + "\n"
                    + "        // translate visible context so operation is drawn in the rig
                    + "        if(n > 0) {\n"
                    + "          if(n % 4 === 0) {\n"
                    + "            context.translate(operationOffset * -3, operationOffset);
                    + "          }\n"
                    + "          else {\n"
                    + "            context.translate(operationOffset, 0);\n"
                    + "          }        // copy drawing from tempCanvas onto visible canva
                    + "        context.drawImage(tempCanvas, 0, 0);\n"
                    + "      }\n"
                    + "    </script>\n"
                    + "  </body>\n"
                    + "</html>";
            bufWriter.write(chaineCaracteres + "\n");
            bufWriter.close();
            fileWriter.close();
        } catch (IOException ex) {
            System.out.println("Problème lors de la sauvegarde");
        }
    }
}
```

-

2.1.8 La Classe visitorprety

:

```
package graphique;
```

```
/**
 *
 * @author Mohammed
 */
public class visitorprety implements VisitorTest {

    @Override
    public void visit(Circle c) {
        System.out.println(c.Circlee()+" est crée");
    }
    @Override
    public void visit(Rectangle r) {
        System.out.println(r.Rectanglee()+" est crée");
    }
    @Override
    public void visit(Composite c) {
        System.out.println(c.Compositee()+" est crée");
    }
}
```

2.2 Partie 2

En suite pour calculer les surfaces et dessiner les graphes voilà ce qu'on a fait :
Apres avoir execute le main du programme suivant :

```
package graphique;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 *
 * @author Mohammed
 */
public class Graphique {
public static Figure[] list={ new Circle(),new Rectangle(),new Composite()};
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        visitorprety vp =new visitorprety();

        Rectangle R=new Rectangle(59, 64, 27, 94);
```

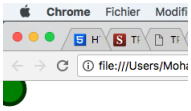
```
Circle C=new Circle(19,8,10);
Composite Cp=new Composite();
Cp.addChild(C);
Cp.addChild(R);
visitorDraw draw = new visitorDraw();
System.out.println("----------------------------------------");
Scanner sc = new Scanner(System.in);
Scanner choix = new Scanner(System.in);
List<String>msg= new ArrayList();
String saisie = "";

String m = "";

do{
System.out.println("-  1.Dessiner une Circle                 -");
System.out.println("-  2.Calucul la surface de la circle     -");
System.out.println("-  3.Dessiner un Rectangele              -");
System.out.println("-  4.Calcule la surface  du Rectangle    -");
System.out.println("-  5.Dessiner une composite              -");
System.out.println("-  6.l'affichage en pretty print         -");
System.out.println("-  7.Quitter                             -");
System.out.println("-  Validez votre choix :                  ");
System.out.println("----------------------------------------");
m = choix.nextLine();
if (m.equals("1")){
   C.accept(draw) ;
}

else if (m.equals("2")){
    double r=C.area();
    System.out.println("la valeur de area est "+r);
}
else if (m.equals("3")){
    R.accept(draw);
}
else if (m.equals("4")){
    double r=R.area();
    System.out.println("la valeur de area est "+r);
}
else if (m.equals("5")){
    draw.drawComposite();
}
else if (m.equals("6")){
    R.accept(vp);
    C.accept(vp);
    Cp.accept(vp);
```

```
        }
        }while(!m.equals("7"));

    }
}
```

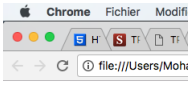
on aura ce menu comme resultat

```
run:
---------------------------------------
-  1.Dessiner une Circle                -
-  2.Calucul la surface de la circle    -
-  3.Dessiner un Rectangele             -
-  4.Calcule la surface  du Rectangle   -
-  5.Dessiner une composite             -
-  6.l'affichage en pretty print        -
-  7.Quitter                            -
-  Validez votre choix :
---------------------------------------
```

choix numéro 1 et 3 :
Circle



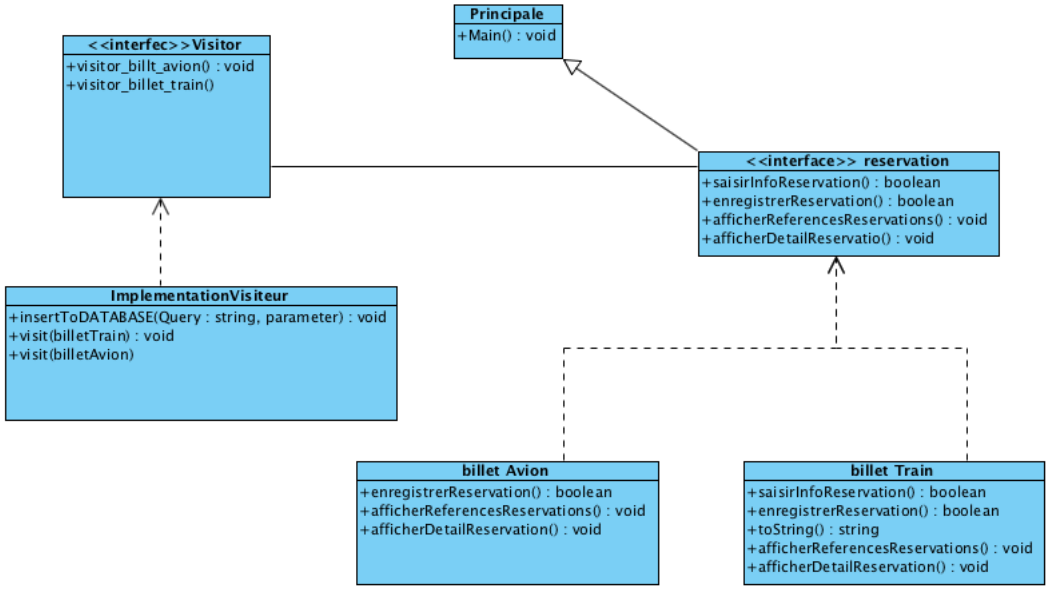
Rectangle



3 Exercice 2

3.1 Partie 1

Après une grande analyse du code donné,on peux appliquer le disign patterns alors on a proposé un diagramme de classe suivant :



3.2 Partie 2

maintenat il faut bien sur implementer le code assicié à ce diagramme
on a les classes : -

3.2.1 La Classe billetTrain

: package Exercie2; /** * * @author Mohammed */ import java.io.BufferedWriter; import java.io.FileWriter; import java.io.IOException; import java.util.*; import java.util.logging.Logger; public class billetTrain implements Reservation private int refTrain; private int refClient; private String gareDepart; private String gareArrivee; public int getRefTrain() return refTrain; public int getRefClient() return refClient; public String getGareDepart() return gareDepart; public String getGareArrivee() return gareArrivee; public billetTrain()

public billetTrain(int refTrain, int refClient, String gareDepart, String gareArrivee) this.refTrain = refTrain; this.refClient = refClient; this.gareDepart = gareDepart; this.gareArrivee = gareArrivee; @Override public boolean saisirInfoReservation() // TODO Auto-generated method stub try Scanner sc = new Scanner(System.in); System.out.println("Quelle est la référence du client :"); refClient = new Integer(sc.nextLine()).intValue(); System.out.println("Quelle est la référence du train :"); refTrain = new Integer(sc.nextLine()).intValue(); System.out.println("Quelle est la gare de départ :"); gareDepart = sc.nextLine(); System.out.println("Quelle est la gare d'arrivée :"); gareArrivee = sc.nextLine(); catch (Exception e) return false; return true;

@Override public boolean enregistrerReservation() // TODO Auto-generated method stub BufferedWriter bufWriter = null; FileWriter fileWriter = null; try fileWriter = new FileWriter("resa.txt", true); bufWriter = new BufferedWriter(fileWriter); bufWriter.write(this.toString() + ""); bufWriter.close(); fileWriter.close(); return true;

catch (IOException ex) System.out.println("Problème lors de la sauvegarde"); return false;

public String toString() return this.refClient + "-" + this.refTrain;

@Override public void afficherReferencesReservations() System.out.println(this);

@Override public void afficherDetailReservation() System.out.println("La reservation concerne le client" + this.refClient + " pour le train " + this.refTrain); System.out.println("départ de " + this.gareDepart + " vers " + this.gareArrivee);

@Override public void accepte(Visiteur v) v.visit(this);

-

3.2.2 La Classe billetAvion

:

```
package Exercice2;
/**
 *
 * @author Mohammed
 */
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;

public class billetAvion implements Reservation {

        private String noVol;
        private int refClient;
        private String classe;
private String depart;
```

```
private String arrivee;

        public String GetInformation()
        {
            return "'"+noVol+"',"+refClient+",'"+classe+"','"+depart+"','"+arrivee+"'";
        }
    public String getNoVol() {
        return noVol;
    }

    public void setNoVol(String noVol) {
        this.noVol = noVol;
    }

    public int getRefClient() {
        return refClient;
    }

    public void setRefClient(int refClient) {
        this.refClient = refClient;
    }

    public String getClasse() {
        return classe;
    }

    public void setClasse(String classe) {
        this.classe = classe;
    }

    public String getDepart() {
        return depart;
    }

    public void setDepart(String depart) {
        this.depart = depart;
    }

    public String getArrivee() {
        return arrivee;
    }

    public void setArrivee(String arrivee) {
        this.arrivee = arrivee;
    }
```

```
        public billetAvion(){}

@Override
public boolean saisirInfoReservation() {
// TODO Auto-generated method stub
try{
Scanner sc = new Scanner(System.in);
System.out.println("Quelle est la référence du client :");
refClient = new Integer(sc.nextLine()).intValue();
System.out.println("Quelle est le no vol (ref de type VN08) :");
noVol = sc.nextLine();
System.out.println("Quelle est l'aeroport de départ :");
depart = sc.nextLine();
System.out.println("Quelle est l'aeroport d'arrivée :");
arrivee = sc.nextLine();
System.out.println("Quelle est la classe de la place (buisness/premium/economy) :");
classe = sc.nextLine();
} catch (Exception e )
{
return false;
}
return true;
}

@Override
public boolean enregistrerReservation() {
// TODO Auto-generated method stub
BufferedWriter bufWriter = null;
        FileWriter fileWriter = null;
        try {
            fileWriter = new FileWriter("resa.txt", true);
            bufWriter = new BufferedWriter(fileWriter);
            bufWriter.write(this.toString()+"\n");
            bufWriter.close();
            fileWriter.close();
            return true;

        } catch (IOException ex) {
            System.out.println("Problème lors de la sauvegarde");
            return false;
        }

}

public String toString(){
return this.refClient+"-"+this.noVol;
```

```
}

@Override
public void afficherReferencesReservations() {
System.out.println(this);
}

@Override
public void afficherDetailReservation() {
System.out.println("La reservation concerne le client" +this.refClient+" pour le vol "+this.
System.out.println("départ de " +this.depart+" vers "+this.arrivee);
System.out.println("en classe " +this.classe);

}

    @Override
    public void accepte(Visiteur v) {
        v.visit(this);
    }

}
```

-

3.2.3 L'interface Visitor

: /* * To change this license header, choose License Headers in Project Properties. * To change this template file, choose Tools — Templates * and open the template in the editor. */ package Exercice2;
/** * * @author Mohammed */ public interface Visiteur public void visit(billetAvion b); public void visit(billetTrain b); public void visit(hotel h);
-

3.2.4 L'interface Reservation

:

```
package Exercice2;

import Exercice.Visiteur;
/**
 *
 * @author Mohammed
 */
public interface Reservation {
// Au travers d'un scanner, demande à l'utilisateur les infos
```

```
public boolean saisirInfoReservation();
// sauvegarde la réservation dans le fichier des réservations : resa.txt
public boolean enregistrerReservation();
// Affiche sous la forme "noclient-reference"
public void afficherReferencesReservations();
// Affiche l'ensemble des détails de la réservation en fonction de l'implementation
public void afficherDetailReservation();
// Methode accepte visiteur
public void accepte(Visiteur  v);
}
```

-

3.2.5 La classe VisitorImpl

:

```
    package Exercice2;

/**
 *
 * @author Mohammed
 */
public class VisiteurImpl implements Visiteur{

    @Override
    public void visit(billetAvion b) {
        String Query="insert into billetavion values("+b.GetInformation()+")";
        System.out.println(Query);
            insertToDATABASE(Query);
    }

    @Override
    public void visit(billetTrain b) {
        String Query="insert into billettrain values("+b.getRefClient()+","+b.getRefTrain()+
        System.out.println(Query);
        insertToDATABASE(Query);
    }

    @Override
    public void visit(hotel h) {
        String Query ="insert into hotel values("+h.GetStingInformation()+")";
        System.out.println(Query);
        insertToDATABASE(Query);
    }
```

```
    public void insertToDATABASE(String Query){
       gestionBDD gs=new gestionBDD();
       gs.openConnection();
       gs.executeRequete(Query);

    }
}
```

-

3.2.6 La classe principale

:

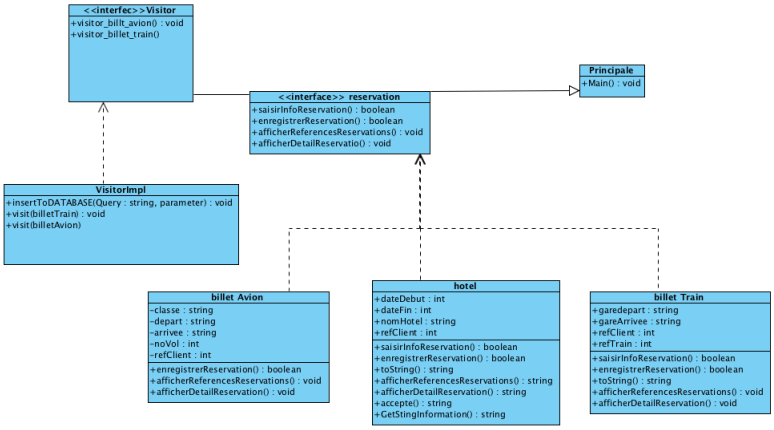
```
    package Exercice2;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

/**
 *
 * @author Mohammed
 */
public class principale {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here


        ImplementationVisiteur FistFisitor = new ImplementationVisiteur();

        gestionBDD g = new gestionBDD();
        g.openConnection();
        Reservation R2 = new hotel();
        R2.saisirInfoReservation();
        R2.afficherDetailReservation();
        R2.afficherReferencesReservations();
        R2.enregistrerReservation();

        R2.accepte(FistFisitor);
    }
```

```
    public void menu(){

    }
}
```

Après l'ajout qu'on ajout la classe Hotel on aura ce diagramme suivaint : -

3.2.7 La classe hotel

:



2.png

puis on va implementer le code de la classe ajouter hotel : -

3.2.8 La classe hotel

:

```
    package Exercice2;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

/**
 *
 * @author Mohammed
 */
public class hotel implements Reservation {
```

```
private int refClient;
private String nomHotel;
private int dateDebut;
private int dateFin;
@Override
public boolean saisirInfoReservation() {
    // TODO Auto-generated method stub
    try {
        Scanner sc = new Scanner(System.in);
        System.out.println("Quelle est la référence du client :");
        refClient = new Integer(sc.nextLine()).intValue();
        System.out.println("Quelle est le nom  de l'hotel :");
        nomHotel = sc.nextLine();
        System.out.println("Quelle est la date de debut :");
        dateDebut = new Integer(sc.nextLine()).intValue();
        System.out.println("Quelle est la date de fin :");
        dateFin = new Integer(sc.nextLine()).intValue();
    } catch (Exception e) {
        return false;
    }
    return true;
}

@Override
public boolean enregistrerReservation() {
    BufferedWriter bufWriter = null;
    FileWriter fileWriter = null;
    try {
        fileWriter = new FileWriter("resa.txt", true);
        bufWriter = new BufferedWriter(fileWriter);
        bufWriter.write(this.toString() + "\n");
        bufWriter.close();
        fileWriter.close();
        return true;

    } catch (IOException ex) {
        System.out.println("Problème lors de la sauvegarde");
        return false;
    }

}

@Override
public String toString() {
    return refClient + " " + nomHotel; //To change body of generated methods, choose Too
}
```

```
    @Override
    public void afficherReferencesReservations() {
        System.out.println(this);
    }

    @Override
    public void afficherDetailReservation() {
        System.out.println("La reservation concerne le client" + this.refClient + " pour l'h
        System.out.println("Début le " + this.dateDebut + "  vers " + this.dateFin);
    }

    @Override
    public void accepte(Visiteur v) {
        //   throw new UnsupportedOperationException("Not supported yet."); //To change body
        v.visit(this);
    }

    public String GetStingInformation() {
        return refClient + ",'" + nomHotel + "'," + dateDebut + "," + dateFin;
    }

}
```

3.3 Partie 3

cette partie est consacré pour la connection avec la base de données on ajout un nouveau Visiteur permettant de sauvegarder dans une base de données les réservations -

3.3.1 La classe gestionBDD

:

```
    package Exercice2;
import java.sql.*;
import java.io.*;
/**
 *
 * @author Mohammed
 */
public class gestionBDD {
    String login;
    String mdp;
```

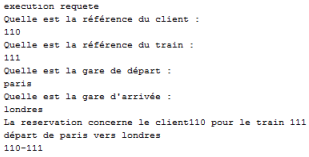
```
String base;
Connection con;
public gestionBDD(){
    login=new String("root");
    mdp=new String("simohammed");
    base=new String("RESERVATION");
}
public void openConnection(){
    try{
        System.out.println("execution requete");

        String ip="localhost";
        String port="3306";
        String protocole="jdbc:mysql:";
        String nomBase=base;
        String conString=protocole+"//"+ip+":"+port+"/"+nomBase;

        System.out.println(conString);


        String nomConnexion=login;
        String motDePasse=mdp;

        con=DriverManager.getConnection(conString,nomConnexion,motDePasse);




    }catch(Exception e){
        System.err.println(e);
    }
}
public void closeConnexion(){
    try {
            con.close();
    } catch (Exception e) {
    }
}
public void executeRequete(String requete)
{
    try {
            System.out.println("Executionn requete");
            Statement smt=con.createStatement();
            int rs=smt.executeUpdate(requete);

    } catch (Exception e) {
```

```
        }
    }
}
```

et voilà le sript :
create database RESERVATION;
use RESERVATION

    create table billetAvion (
noVol varchar(50),
refclient int,
classe varchar(50),
depart varchar(50),
arrive varchar(50)
)
create table billetTrain (
refTrain int,
refClient int,
gareDepart varchar(50),
gareArrivee varchar(50)
)
create table hotel (
refClient int,
nomHotel varchar(50),
dateDebut int,
dateFin int
)

3.3.2 Verification



vous trouvé ci-joint dans le répertoire tout le code de chauqe classe de TP

4 Conclusion

Dans la fin, il fallait penser aux objectifs et aux methodes qu'on a suivi pour atteindre le but de ce travail. Ce projet nous a permis de découvrir plus profondément plusieurs aspects du design pattern et son utilité . Ansi il nous a permis de comprendre plus en profondeur le développement logiciel, de la modélisation et la conception en passant par la réflexion logique et algorithmique.