dao.InterventionDaoJpa

```
1
2  package dao;
3  import javax.persistence.EntityManager;
4  import metier.modele.Intervention;
5  /**
6   *
7   * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
8   * Classe qui gère la persistance des interventions
9   *
10  */
11 public class InterventionDaoJpa{
12
13     //persite l'intervention placée en paramètre
14     public static void creerIntervention(Intervention i){
15         EntityManager em=JpaUtil.obtenirEntityManager();
16         try {
17             em.persist(i);
18         } catch(Exception e) {
19         }
20     }
21
22     //mise à jour de l'intervention
23      //Pour le cas où on change son statut
24     public static void modifierIntervention(Intervention i){
25         EntityManager em=JpaUtil.obtenirEntityManager();
26         try {
27             em.merge(i);
28         } catch(Exception e) {
29         }
30     }
31
32 }
```

dao.JpaUtil

```java

package dao;

/**
 *
 *@author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 */
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.RollbackException;

/**
 * Cette classe fournit des méthodes statiques utiles pour accéder aux
 * fonctionnalités de JPA (Entity Manager, Entity Transaction). Le nom de
 * l'unité de persistance (PERSISTENCE_UNIT_NAME) doit être conforme à la
 * configuration indiquée dans le fichier persistence.xml du projet.
 *
 * @author DASI Team
 */
public class JpaUtil {

    // ********************************************************************************
    // * TODO: IMPORTANT -- Adapter le nom de l'Unité de Persistance (cf. persistence.xml) *
    // ********************************************************************************
    /**
     * Nom de l'unité de persistance utilisée par la Factory de Entity Manager.
     * <br><strong>Vérifier le nom de l'unité de persistance
     * (cf. persistence.xml)</strong>
     */
    public static final String PERSISTENCE_UNIT_NAME = "TPDASIPU";
    /**
     * Factory de Entity Manager liée à l'unité de persistance.
     * <br/><strong>Vérifier le nom de l'unité de persistance indiquée dans
     * l'attribut statique PERSISTENCE_UNIT_NAME
     * (cf. persistence.xml)</strong>
     */
    private static EntityManagerFactory entityManagerFactory = null;
    /**
     * Gère les instances courantes de Entity Manager liées aux Threads.
     * L'utilisation de ThreadLocal garantie une unique instance courante par
     * Thread.
     */
    private static final ThreadLocal<EntityManager> threadLocalEntityManager = new ThreadLocal<
        EntityManager>() {

        @Override
        protected EntityManager initialValue() {
            return null;
        }
    };

    // Méthode pour avoir des messages de Log dans le bon ordre (pause)
    private static void pause(long milliseconds) {
        try {
            Thread.sleep(milliseconds);
        } catch (InterruptedException ex) {
            ex.hashCode();
        }
    }

    // Méthode pour avoir des messages de Log dans le bon ordre (log)
    private static void log(String message) {
       /* System.out.flush();
        pause(5);
        System.err.println("[JpaUtil:Log] " + message);
        System.err.flush();
        pause(5);*/
    }

    /**
     * Initialise la Factory de Entity Manager.
     * <br><strong>À utiliser uniquement au début de la méthode main() [projet
     * Java Application] ou dans la méthode init() de la Servlet Contrôleur
     * (ActionServlet) [projet Web Application].</strong>
```

```
75       */
76      public static synchronized void init() {
77          log("Initialisation de la factory de contexte de persistance");
78          if (entityManagerFactory != null) {
79              entityManagerFactory.close();
80          }
81          entityManagerFactory = Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
82      }
83
84      /**
85       * Libère la Factory de Entity Manager.
86       * <br><strong>À utiliser uniquement à la fin de la méthode main() [projet
87       * Java Application] ou dans la méthode destroy() de la Servlet Contrôleur
88       * (ActionServlet) [projet Web Application].</strong>
89       */
90      public static synchronized void destroy() {
91          log("Libération de la factory de contexte de persistance");
92          if (entityManagerFactory != null) {
93              entityManagerFactory.close();
94              entityManagerFactory = null;
95          }
96      }
97
98      /**
99       * Créée l'instance courante de Entity Manager (liée à ce Thread).
100      * <br><strong>À utiliser uniquement au niveau Service.</strong>
101      */
102     public static void creerEntityManager() {
103         log("Création du contexte de persistance");
104         threadLocalEntityManager.set(entityManagerFactory.createEntityManager());
105     }
106
107     /**
108      * Ferme l'instance courante de Entity Manager (liée à ce Thread).
109      * <br><strong>À utiliser uniquement au niveau Service.</strong>
110      */
111     public static void fermerEntityManager() {
112         log("Fermeture du contexte de persistance");
113         EntityManager em = threadLocalEntityManager.get();
114         em.close();
115         threadLocalEntityManager.set(null);
116     }
117
118     /**
119      * Démarre une transaction sur l'instance courante de Entity Manager.
120      * <br><strong>À utiliser uniquement au niveau Service.</strong>
121      */
122     public static void ouvrirTransaction() {
123         log("Ouverture de la transaction (begin)");
124         try {
125             EntityManager em = threadLocalEntityManager.get();
126             em.getTransaction().begin();
127         } catch (Exception ex) {
128             log("Erreur lors de l'ouverture de la transaction");
129             throw ex;
130         }
131     }
132
133     /**
134      * Valide la transaction courante sur l'instance courante de Entity Manager.
135      * <br><strong>À utiliser uniquement au niveau Service.</strong>
136      *
137      * @exception RollbackException lorsque le <em>commit</em> n'a pas réussi.
138      */
139     public static void validerTransaction() throws RollbackException {
140         log("Validation de la transaction (commit)");
141         try {
142             EntityManager em = threadLocalEntityManager.get();
143             em.getTransaction().commit();
144         } catch (Exception ex) {
145             log("Erreur lors de la validation (commit) de la transaction");
146             throw ex;
147         }
148     }
149
150     /**
151      * Annule la transaction courante sur l'instance courante de Entity Manager.
```

```
152      * Si la transaction courante n'est pas démarrée, cette méthode n'effectue
153      * aucune opération.
154      * <br><strong>À utiliser uniquement au niveau Service.</strong>
155      */
156     public static void annulerTransaction() {
157         try {
158             log("Annulation de la transaction (rollback)");
159 
160             EntityManager em = threadLocalEntityManager.get();
161             if (em.getTransaction().isActive()) {
162                 log("Annulation effective de la transaction (rollback d'une transaction active)");
163                 em.getTransaction().rollback();
164             }
165 
166         } catch (Exception ex) {
167             log("Erreur lors de l'annulation (rollback) de la transaction");
168             throw ex;
169         }
170     }
171 
172     /**
173      * Retourne l'instance courante de Entity Manager.
174      * <br><strong>À utiliser uniquement au niveau DAO.</strong>
175      *
176      * @return instance de Entity Manager
177      */
178     protected static EntityManager obtenirEntityManager() {
179         log("Obtention du contexte de persistance");
180         return threadLocalEntityManager.get();
181     }
182 }
```

dao.PersonneDaoJpa

```
1
2  package dao;
3  import java.util.List;
4  import javax.persistence.EntityManager;
5  import javax.persistence.Query;
6  import metier.modele.Employe;
7  import metier.modele.Personne;
8
9  /**
10  *
11  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
12  * Classe qui gère la persistance des Personnes
13  */
14 public class PersonneDaoJpa{
15
16     public static void creerPersonne(Personne p){
17         EntityManager em=JpaUtil.obtenirEntityManager();
18         try {
19             em.persist(p);
20         } catch(Exception e) {
21         }
22     }
23
24     //mise à jour de la Personne
25      //Pour le cas où on change sa disponibilité
26     public static void modifierPersonne(Personne p){
27         EntityManager em=JpaUtil.obtenirEntityManager();
28         try {
29             em.merge(p);
30         } catch(Exception e) {
31         }
32     }
33
34     //Trouve la peronne qui possède le mail indiqué
35     public static Personne recupererPersonne(String mail){
36         String jpql= "Select␣p␣from␣Personne␣p␣where␣p.mail=:email";
37         Query requete=JpaUtil.obtenirEntityManager().createQuery(jpql);
38         requete.setParameter("email",mail);
39         Personne result=null;
40         try{
41             result=(Personne) requete.getSingleResult();
42         } catch (Exception e){
43         }
44         return result;
45     }
46
47     //Trouve tous les employés disponibles et qui travaillent à l'heure indiquée
48     public static List<Employe> trouverListeEmployeDispo(int heure){
49         EntityManager em=JpaUtil.obtenirEntityManager();
50         String jpql= "Select␣e␣from␣Employe␣e␣where␣e.dispo=1␣and␣e.horaireEntree<=:heure␣and␣e.
               horaireSortie>:heure";
51         Query requete=em.createQuery(jpql);
52         requete.setParameter("heure",heure);
53         List<Employe> result=null;
54         try{
55             result=(List<Employe>) requete.getResultList();
56         } catch (Exception e){
57         }
58         return result;
59     }
60
61 }
```

util.Saisie

```java

package util;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.List;

/**
 *
 * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 */
public class Saisie {

    public static String lireChaine(String invite) {
        String chaineLue = null;
        System.out.print(invite);
        try {
            InputStreamReader isr = new InputStreamReader(System.in);
            BufferedReader br = new BufferedReader(isr);
            chaineLue = br.readLine();
        } catch (IOException ex) {
            ex.printStackTrace(System.err);
        }
        return chaineLue;

    }

    public static Integer lireInteger(String invite) {
        Integer valeurLue = null;
        while (valeurLue == null) {
            try {
                valeurLue = Integer.parseInt(lireChaine(invite));
            } catch (NumberFormatException ex) {
                System.out.println("/!\\ Erreur de saisie - Nombre entier attendu /!\\");
            }
        }
        return valeurLue;
    }

    public static Integer lireInteger(String invite, List<Integer> valeursPossibles) {
        Integer valeurLue = null;
        while (valeurLue == null) {
            try {
                valeurLue = Integer.parseInt(lireChaine(invite));
            } catch (NumberFormatException ex) {
                System.out.println("/!\\ Erreur de saisie - Nombre entier attendu /!\\");
            }
            if (!(valeursPossibles.contains(valeurLue))) {
                System.out.println("/!\\ Erreur de saisie - Valeur non-autorisée /!\\");
                valeurLue = null;
            }
        }
        return valeurLue;
    }

    public static void pause() {
        lireChaine("--PAUSE--");
    }

}
```

util.Message

```java

package util;

import java.io.PrintStream;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 *
 * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 */
public class Message {

    private final static PrintStream OUT = System.out;
    private final static SimpleDateFormat TIMESTAMP_FORMAT = new SimpleDateFormat("yyyy-MM-dd~HH:mm:ss");
    private final static SimpleDateFormat HORODATE_FORMAT = new SimpleDateFormat("dd/MM/yyyy à HH:mm:ss")
        ;

    private static void debut() {
        Date maintenant = new Date();
        OUT.println();
        OUT.println();
        OUT.println("---<([ MESSAGE @ " + TIMESTAMP_FORMAT.format(maintenant) + " ])>---");
        OUT.println();
    }

    private static void fin() {
        OUT.println();
        OUT.println("---<([ FIN DU MESSAGE ])>---");
        OUT.println();
        OUT.println();
    }

    public static void envoyerMail(String mailExpediteur, String mailDestinataire, String objet, String
        corps) {

        Date maintenant = new Date();
        Message.debut();
        OUT.println("~~~ E-mail envoyé le " + HORODATE_FORMAT.format(maintenant) + " ~~~");
        OUT.println("Expéditeur : " + mailExpediteur);
        OUT.println("Pour  : " + mailDestinataire);
        OUT.println("Sujet : " + objet);
        OUT.println("Corps : " +corps);
        Message.fin();
    }

    public static void envoyerNotification(String nom, String prenom,String telephoneDestinataire, String
         message) {

        Date maintenant = new Date();
        Message.debut();
        OUT.println("~~~ Notification envoyée le " + HORODATE_FORMAT.format(maintenant) + " ~~~");
        OUT.println("Pour  : " + prenom+" "+nom);
        OUT.println("Tel  : " + telephoneDestinataire);
        OUT.println("Message : "+message);
        Message.fin();
    }

}
```

util.DebugLogger

```java

package util;

/**
 *
 * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 * FACULTATIF
 * Gestion des affichages de Debug dans la console
 */
public class DebugLogger {

    // Méthode pour avoir des messages de Log dans le bon ordre (pause)
    public static void pause(long milliseconds) {
        try {
            Thread.sleep(milliseconds);
        } catch (InterruptedException ex) {
            ex.hashCode();
        }
    }

    // Méthode pour avoir des messages de Log dans le bon ordre (log)
    public static void log(String message) {
        System.out.flush();
        pause(5);
        System.err.println("[DebugLogger] " + message);
        System.err.flush();
        pause(5);
    }

    // Méthode pour avoir des messages de Log dans le bon ordre (log avec exception)
    public static void log(String message, Exception ex) {
        System.out.flush();
        pause(5);
        System.err.println("[DebugLogger] " + message);
        System.err.println("[**EXCEPTION**] " + ex.getMessage());
        System.err.print("[**EXCEPTION**] >>> ");
        ex.printStackTrace(System.err);
        System.err.flush();
        pause(5);
    }

}
```

util.GeoTest

```java
/*
 * 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 util;

import com.google.maps.DirectionsApi;
import com.google.maps.DirectionsApiRequest;
import com.google.maps.GeoApiContext;
import com.google.maps.GeocodingApi;
import com.google.maps.model.DirectionsResult;
import com.google.maps.model.DirectionsRoute;
import com.google.maps.model.GeocodingResult;
import com.google.maps.model.LatLng;
import com.google.maps.model.TravelMode;

/**
 *
 * @author colap
 */
public class GeoTest {

    final static String MA_CLE_GOOGLE_API = "AIzaSyDcVVJjfmxsNdbdUYeg9MjQoJJ6THPuap4";//
        AIzaSyALiuq9WOv2l6aw4361X8bnyGPv4B4vA58";

    final static GeoApiContext MON_CONTEXTE_GEOAPI = new GeoApiContext.Builder().apiKey(MA_CLE_GOOGLE_API
        ).build();

    public static LatLng getLatLng(String adresse) {
        try {
            GeocodingResult[] results = GeocodingApi.geocode(MON_CONTEXTE_GEOAPI, adresse).await();

            return results[0].geometry.location;

        } catch (Exception ex) {
            return null;
        }
    }

    public static double toRad(double angleInDegree) {
        return angleInDegree * Math.PI / 180.0;
    }

    public static double getFlightDistanceInKm(LatLng origin, LatLng destination) {

        // From: http://www.movable-type.co.uk/scripts/latlong.html
        double R = 6371.0; // Average radius of Earth (km)
        double dLat = toRad(destination.lat - origin.lat);
        double dLon = toRad(destination.lng - origin.lng);
        double lat1 = toRad(origin.lat);
        double lat2 = toRad(destination.lat);

        double a = Math.sin(dLat / 2.0) * Math.sin(dLat / 2.0)
                + Math.sin(dLon / 2.0) * Math.sin(dLon / 2.0) * Math.cos(lat1) * Math.cos(lat2);
        double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1.0 - a));
        double d = R * c;

        return Math.round(d * 1000.0) / 1000.0;
    }

    public static Double getTripDurationByBicycleInMinute(LatLng origin, LatLng destination, LatLng...
        steps) {
        return getTripDurationOrDistance(TravelMode.BICYCLING, true, origin, destination, steps);
    }

    public static Double getTripDistanceByCarInKm(LatLng origin, LatLng destination, LatLng... steps) {
        return getTripDurationOrDistance(TravelMode.DRIVING, false, origin, destination, steps);
    }

    public static Double getTripDurationOrDistance(TravelMode mode, boolean duration, LatLng origin,
        LatLng destination, LatLng... steps) {

        DirectionsApiRequest request = DirectionsApi.getDirections(MON_CONTEXTE_GEOAPI, origin.toString()
            , destination.toString());
```

```
71          request.mode(mode);
72          request.region("fr");
73
74          if (steps.length > 0) {
75
76              String[] stringSteps = new String[steps.length];
77              for (int i = 0; i < steps.length; i++) {
78                  stringSteps[i] = steps[i].toString();
79              }
80
81              request.waypoints(stringSteps);
82          }
83
84          double cumulDistance = 0.0;
85          double cumulDuration = 0.0;
86
87          try {
88              DirectionsResult result = request.await();
89              DirectionsRoute[] directions = result.routes;
90
91              for (int legIndex = 0; legIndex < directions[0].legs.length; legIndex++) {
92
93                  cumulDistance += directions[0].legs[legIndex].distance.inMeters / 1000.0;
94                  cumulDuration += Math.ceil(directions[0].legs[legIndex].duration.inSeconds / 60.0);
95              }
96
97          } catch (Exception ex) {
98              return null;
99          }
100
101         if (duration) {
102             return cumulDuration;
103         } else {
104             return cumulDistance;
105         }
106     }
107
108 }
```

metier.modele.Personne

```java

package metier.modele;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;

/**
 *
 *@author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 * Description d'une Personne
 */
@Entity
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
public abstract class Personne implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    private String nom;
    private String prenom;
    private boolean civilite;
    private String motDePasse;

    private String adressePostale;

    private String tel;
    private String mail;

    //calculer à partir de l'adresse
    private double latitude;
    private double longitude;


    public Personne() {
    }

    public Personne(boolean civilite, String nom, String prenom, String motDePasse, String adressePostale
        , String tel, String mail) {
        this.civilite = civilite;
        this.nom = nom;
        this.prenom = prenom;
        this.motDePasse = motDePasse;
        this.adressePostale = adressePostale;
        this.tel = tel;
        this.mail = mail;
    }

    public void setCivilite(boolean civilite) {
        this.civilite = civilite;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public void setMotDePasse(String motDePasse) {
        this.motDePasse = motDePasse;
    }

    public void setAdressePostale(String adressePostale) {
        this.adressePostale = adressePostale;
    }

    public void setTel(String tel) {
        this.tel = tel;
```

```
75      }
76
77      public void setMail(String mail) {
78          this.mail = mail;
79      }
80
81      public void setLatitude(Float latitude) {
82          this.latitude = latitude;
83      }
84
85      public void setLongitude(Float longitude) {
86          this.longitude = longitude;
87      }
88
89      public Integer getId() {
90          return id;
91      }
92
93      public boolean isCivilite() {
94          return civilite;
95      }
96
97      public String getNom() {
98          return this.nom;
99      }
100
101     public String getPrenom() {
102         return prenom;
103     }
104
105     public String getMotDePasse() {
106         return motDePasse;
107     }
108
109     public String getAdressePostale() {
110         return adressePostale;
111     }
112
113     public String getTel() {
114         return tel;
115     }
116
117     public String getMail() {
118         return mail;
119     }
120
121     public double getLatitude() {
122         return latitude;
123     }
124
125     public double getLongitude() {
126         return longitude;
127     }
128
129     public void setId(Integer id) {
130         this.id = id;
131     }
132
133     public void setLatitude(double latitude) {
134         this.latitude = latitude;
135     }
136
137     public void setLongitude(double longitude) {
138         this.longitude = longitude;
139     }
140
141 }
```

metier.modele.Intervention

```java

package metier.modele;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.ManyToOne;

/**
 *
 *@author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 * Description d'une Intervention
 */
@Entity
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
public abstract class Intervention implements Serializable {
    @ManyToOne
    private Client unClient;

    @ManyToOne
    private Employe unEmploye;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    private int type;
    private String description;

    private int status;
    private int heureDeFin;
    private String commentaire;

    @Temporal(TemporalType.DATE)
     private Date horodate;

    public Intervention() {
    }

    public Intervention(int type, String description) {
        this.type = type;
        this.description = description;
    }


    public Integer getId() {
        return id;
    }

    public int getStatus() {
        return status;
    }

    public int getType() {
        return type;
    }

    public String getDescription() {
        return description;
    }

    public int getHeureDeFin() {
        return heureDeFin;
    }

    public String getCommentaire() {
        return commentaire;
    }
```

```
76
77      public Date getHorodate() {
78          return horodate;
79      }
80
81      public void setStatus(int status) {
82          this.status = status;
83      }
84
85      public void setType(int type) {
86          this.type = type;
87      }
88
89      public void setDescription(String description) {
90          this.description = description;
91      }
92
93      public void setHeureDeFin(int heureDeFin) {
94          this.heureDeFin = heureDeFin;
95      }
96
97      public void setCommentaire(String commentaire) {
98          this.commentaire = commentaire;
99      }
100
101     public void setHorodate(Date horodate) {
102         this.horodate = horodate;
103     }
104
105     public Client getUnClient() {
106         return unClient;
107     }
108
109     public void setUnClient(Client unClient) {
110         this.unClient = unClient;
111     }
112
113     public Employe getUnEmploye() {
114         return unEmploye;
115     }
116
117     public void setUnEmploye(Employe unEmploye) {
118         this.unEmploye = unEmploye;
119     }
120
121 }
```

metier.modele.Employe

```
1
2  package metier.modele;
3
4  import java.util.ArrayList;
5  import java.util.List;
6  import javax.persistence.Entity;
7  import javax.persistence.OneToMany;
8
9  /**
10  *
11  * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
12  * Description d'un Employe
13  */
14 @Entity
15 public class Employe extends Personne {
16     @OneToMany(mappedBy="unEmploye")
17     private List<Intervention> listInterventions;
18
19     boolean dispo;
20     int horaireEntree;
21     int horaireSortie;
22
23     public Employe() {
24     }
25
26     public Employe(boolean civilite, String nom, String prenom, String motDePasse, String adressePostale,
           String tel, String mail,boolean dispo,int horaireEntree,int horaireSortie) {
27         super(civilite, nom, prenom, motDePasse, adressePostale, tel, mail);
28         this.dispo=dispo;
29         this.horaireEntree=horaireEntree;
30         this.horaireSortie=horaireSortie;
31         this.listInterventions=new ArrayList<Intervention>();
32     }
33
34     public void ajouterIntervention(Intervention i){
35        listInterventions.add(i);
36        i.setUnEmploye(this);
37     }
38
39     public boolean isDispo() {
40         return dispo;
41     }
42
43     public int getHoraireEntree() {
44         return horaireEntree;
45     }
46
47     public int getHoraireSortie() {
48         return horaireSortie;
49     }
50
51     public List<Intervention> getListInterventions() {
52         return listInterventions;
53     }
54     public void setDispo(boolean dispo) {
55         this.dispo = dispo;
56     }
57
58     public void setHoraireEntree(int horaireEntree) {
59         this.horaireEntree = horaireEntree;
60     }
61
62     public void setHoraireSortie(int horaireSortie) {
63         this.horaireSortie = horaireSortie;
64     }
65
66     public void setListInterventions(List<Intervention> interventions) {
67         this.listInterventions = interventions;
68     }
69
70 }
```

metier.modele.InterventionIncident

```java

package metier.modele;

import javax.persistence.Entity;

/**
 *
 *@author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 * Description d'une InterventionIncident
 */
@Entity
public class InterventionIncident extends Intervention{

    public InterventionIncident() {
    }

    public InterventionIncident(String description) {
        super(3, description);
    }
}
```

metier.modele.InterventionAnimal

```java

package metier.modele;

import javax.persistence.Entity;

/**
 *
 * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 * Description d'une InterventionAnimal
 */
@Entity
public class InterventionAnimal extends Intervention {
    private String animal;

    public InterventionAnimal() {
    }

    public InterventionAnimal(String animal) {
        this.animal = animal;
    }

    public InterventionAnimal(String animal, String description) {
        super(1, description);//initialisation du type à 1
        this.animal = animal;
    }

    public String getAnimal() {
        return animal;
    }

    public void setAnimal(String animal) {
        this.animal = animal;
    }
}
```

metier.modele.Client

```java

package metier.modele;

/**
 *
 * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 * Description d'un Client
 */
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.persistence.Entity;
import javax.persistence.OneToMany;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
public class Client extends Personne {
    @OneToMany(mappedBy="unClient")
    private List<Intervention> listInterventions;

    @Temporal(TemporalType.DATE)
    private Date dateDeNaissance;

    public Client() {
        this.listInterventions=new ArrayList<Intervention>();
    }

    public Client(boolean civilite, String nom, String prenom, String motDePasse, String adressePostale,
        String tel, String mail, Date dateDeNaissance) {
        super(civilite, nom, prenom, motDePasse, adressePostale, tel, mail);
        this.dateDeNaissance = dateDeNaissance;
        this.listInterventions=new ArrayList<Intervention>();
    }

    public void ajouterIntervention(Intervention i){
       listInterventions.add(i);
       i.setUnClient(this);
    }

    public Date getDateDeNaissance() {
        return dateDeNaissance;
    }

    public void setDateDeNaissance(Date dateDeNaissance) {
        this.dateDeNaissance = dateDeNaissance;
    }

    public List<Intervention> getListInterventions() {
        return listInterventions;
    }

    public void setListInterventions(List<Intervention> listInterventions) {
        this.listInterventions = listInterventions;
    }

}
```

metier.modele.InterventionLivraison

```java

package metier.modele;

/**
 *
 * @author Chanèle Jourdan, Jorge Terreu, Corentin laharotte
 * Description d'une InterventionLivraison
 */

import javax.persistence.Entity;

@Entity
public class InterventionLivraison extends Intervention {
    private String objet;
    private String entreprise;

    public InterventionLivraison() {
    }

    public InterventionLivraison(String objet, String entreprise, String description) {
        super(2, description);//initialisation du type à 2
        this.objet = objet;
        this.entreprise = entreprise;
    }

    public String getObjet() {
        return objet;
    }

    public String getEntreprise() {
        return entreprise;
    }

    public void setObjet(String objet) {
        this.objet = objet;
    }

    public void setEntreprise(String entreprise) {
        this.entreprise = entreprise;
    }

}
```

metier.service.Service

```java
/* @author jorge terreu, corentin laharotte, chanèle jourdan
   Classe qui implémente l'ensemble des services de notre application
*/

package metier.service;

import com.google.maps.model.LatLng;
import dao.InterventionDaoJpa;
import dao.JpaUtil;
import dao.PersonneDaoJpa;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import metier.modele.Client;
import metier.modele.Employe;
import metier.modele.Intervention;
import metier.modele.InterventionAnimal;
import metier.modele.InterventionIncident;
import metier.modele.InterventionLivraison;
import metier.modele.Personne;
import java.util.Date;
import util.GeoTest;
import util.Message;

public class Service{

        //Connection du client ou de l'employé à l'application
        public static Personne seConnecter(String mail, String motdepasse){
                JpaUtil.creerEntityManager();
                JpaUtil.ouvrirTransaction();
                Personne a=PersonneDaoJpa.recupererPersonne(mail);
                if (a!=null){
                        if (!(a.getMotDePasse().equals(motdepasse)))
                            a=null;
                }
                JpaUtil.validerTransaction();
                JpaUtil.fermerEntityManager();
                return a;
        }

        //Inscription d'un client. Vérifications effectuées : tous les champs ont été remplis, le mail
            contient un @, le client a plus de 16 ans, le tel contient 10 caractères
        public static Personne seInscrire(boolean civilite, String nom, String prenom, String motDePasse,
             String adressePostale, String tel, String mail, Date dateDeNaissance){
                JpaUtil.creerEntityManager();
                JpaUtil.ouvrirTransaction();
                Personne x = PersonneDaoJpa.recupererPersonne(mail);
                if (x==null){
                        Date ahorita = new Date();
                        //seize ans en millisecondes
                        double seizeAns= 16.0*365.0*24.0*3600.0*1000.0;
                        //différence de temps en millisecondes entre aujourd'hui et dateDeNaissance
                        double differenceTemps=ahorita.getTime() - dateDeNaissance.getTime();

                        if(8 < motDePasse.length() && motDePasse.length()< 16 && nom.length()>0 && prenom
                            .length()>0 && mail.length()>2 && mail.contains("@") &&
                        adressePostale.length()>0 && tel.length()==10 && (differenceTemps > seizeAns)) {

                                //Création et persistence d'un nouveau client
                                x = new Client (civilite, nom, prenom, motDePasse, adressePostale, tel,
                                    mail, dateDeNaissance);
                                calculerLatLng(x);
                                PersonneDaoJpa.creerPersonne(x);

                                //Envoi de mail de confirmation d'inscription
                                String message="Bonjour␣"+prenom+",␣nous␣vous␣confirmons␣votre␣
                                    inscription␣au␣service␣PROACT'IF.";
                                Message.envoyerMail("contact@proact.if", mail, "Bienvenue␣chez␣PROACT'IF"
                                    , message);
                        }
                }else{
                //Envoie de mail pour informer de l'échec de l'inscription
                String message="Bonjour␣"+prenom+",␣␣votre␣inscription␣a␣échoué,␣veuillez␣recommencer␣
                    ultérieurement.";
                Message.envoyerMail("contact@proact.if", mail, "Bienvenue␣chez␣PROACT'IF", message);
```

```
69              x=null;
70              }
71              JpaUtil.validerTransaction();
72              JpaUtil.fermerEntityManager();
73              return x;
74          }
75
76          //Demande d'une intervention de type Animal. Vérification : tous les champs sont remplis
77          public static Employe demanderIntervention(Client c,int type, String description){
78              JpaUtil.creerEntityManager();
79              JpaUtil.ouvrirTransaction();
80              Employe employeTrouve=trouverEmploye(c);
81              if (employeTrouve!=null && type>= 0 && type <=3 && description.length()>0){
82
83                      //Création et persistence de l'intervention
84                      InterventionIncident a = new InterventionIncident(description);
85                      a.setHorodate(new  Date());
86                      c.ajouterIntervention(a);
87                      employeTrouve.ajouterIntervention(a);
88                      employeTrouve.setDispo(false);
89                      InterventionDaoJpa.creerIntervention(a);
90
91                      //Envoi d'une notification à l'employé trouvé
92                      String message = "Intervention␣Animal␣demandee␣pour␣" +c.getPrenom()+ "␣" +c.
                            getNom()+ ",␣" +c.getAdressePostale()+ ".␣Commentaire␣:␣" +description;
93                      Message.envoyerNotification(employeTrouve.getNom(), employeTrouve.getPrenom(),
                            employeTrouve.getTel(), message);
94              }else{
95                  employeTrouve=null;
96              }
97              JpaUtil.validerTransaction();
98              JpaUtil.fermerEntityManager();
99              return employeTrouve;
100
101
102         }
103
104         //Demande d'une intervention de type Livraison. Vérification : tous les champs sont remplis
105         public static Employe demanderIntervention(Client c,int type, String description, String objet,
                String entreprise){
106             JpaUtil.creerEntityManager();
107             JpaUtil.ouvrirTransaction();
108             Employe employeTrouve=trouverEmploye(c);
109             if (employeTrouve!=null && type>= 0 && type <=3 && description.length()>0 && objet.length
                    ()>0 && entreprise.length()>0){
110
111                     //Creation et persistence de l'intervention
112                     InterventionLivraison a = new InterventionLivraison(objet, entreprise,
                            description);
113                     a.setHorodate(new  Date());
114                     c.ajouterIntervention(a);
115                     employeTrouve.ajouterIntervention(a);
116                     employeTrouve.setDispo(false);
117                     InterventionDaoJpa.creerIntervention(a);
118
119                     //Envoi d'une notification à l'employé trouvé
120                     String message = "Intervention␣Livraison␣demandee␣pour␣" +c.getPrenom()+ "␣" +c.
                            getNom()+ ",␣" +c.getAdressePostale()+ ".␣Commentaire␣:␣" +description;
121                     Message.envoyerNotification(employeTrouve.getNom(), employeTrouve.getPrenom(),
                            employeTrouve.getTel(), message);
122
123             }else{
124                 employeTrouve=null;
125             }
126             JpaUtil.validerTransaction();
127             JpaUtil.fermerEntityManager();
128             return employeTrouve;
129         }
130
131         //Demande d'une intervention de type Incident. Vérification : tous les champs sont remplis
132         public static Employe demanderIntervention(Client c,int type, String description, String animal){
133             JpaUtil.creerEntityManager();
134             JpaUtil.ouvrirTransaction();
135             Employe employeTrouve=trouverEmploye(c);
136             if (employeTrouve!=null && type>= 0 && type <=3 && description.length()>0 && animal.
```

```
                length()>0 ){
137
                    //Creation et persistence de l'intervention
138
                    InterventionAnimal a = new InterventionAnimal(animal, description);
139
                    a.setHorodate(new  Date());
140
                    c.ajouterIntervention(a);
141
                    employeTrouve.ajouterIntervention(a);
142
                    employeTrouve.setDispo(false);
143
                    InterventionDaoJpa.creerIntervention(a);
144
145
                    //Envoi d'une notification à l'employé trouvé
146
                    String message = "Intervention Incident demandee pour " +c.getPrenom()+ " " +c.
147
                        getNom()+ ", " +c.getAdressePostale()+ ". Commentaire : " +description;
148
                    Message.envoyerNotification(employeTrouve.getNom(), employeTrouve.getPrenom(),
                        employeTrouve.getTel(), message);
149
            }else{
150
                employeTrouve=null;
151
            }
152
            JpaUtil.validerTransaction();
153
            JpaUtil.fermerEntityManager();
154
            return employeTrouve;
155
        }
156
157
        //Recherche de l'employé le plus proche du client pour faire une intervention.
158
        //Vérification : il est disponible, l'heure de la demande d'intervention est dans ses horaires de
             travail
159
        public static Employe trouverEmploye(Client c){
160
            int heure=heureActuelleToInt();
161
162
            //Récupération de la liste des employés disponibles
163
            List <Employe> list=PersonneDaoJpa.trouverListeEmployeDispo(heure);
164
165
            Employe employeSelect=null;
166
            if(list!=null){
167
                double min=1000000000;
168
                LatLng clientGPS=new LatLng(c.getLatitude(),c.getLongitude());
169
                for (Employe e : list){
170
                            LatLng employeGPS=new LatLng(e.getLatitude(),e.getLongitude());
171
                            double tempsTrajet=GeoTest.getTripDurationByBicycleInMinute(clientGPS,
                                employeGPS);
172
                            if(tempsTrajet<min){
173
                                min=tempsTrajet;
174
                                employeSelect=e;
175
                            }
176
                    }
177
            }
178
            return employeSelect;
179
        }
180
181
        //Récupération de l'historique d'un client pour le consulter
182
        public static List<Intervention> recupererHistorique(Client c){
183
            JpaUtil.creerEntityManager();
184
            JpaUtil.ouvrirTransaction();
185
            PersonneDaoJpa.modifierPersonne(c);
186
            JpaUtil.validerTransaction();
187
            JpaUtil.fermerEntityManager();
188
            return c.getListInterventions();
189
        }
190
191
        //Récupération des interventions d'un employé pour les consulter (tableau de bord)
192
        public static List<Intervention> recupererTableauDeBord(Employe e){
193
            List <Intervention>l=e.getListInterventions();
194
            List <Intervention> list=new ArrayList<Intervention>();
195
            Date d=new Date();
196
            for (Intervention i: l){
197
                        if(d.compareTo(i.getHorodate())!=0)
198
                            list.add(i);
199
            }
200
            return list;
201
        }
202
203
        //Clotûre d'une intervention par un employé. Vérification : tous les champs sont remplis
204
        public static boolean cloturerIntervention(Employe emp, int status, int heureDefin, String
            commentaire){
205
                JpaUtil.creerEntityManager();
206
                JpaUtil.ouvrirTransaction();
```

```
207                 Intervention i = rechercherInterventionEnCours(emp);
208                 boolean cloture=false;
209                 if (status >= 0 && status <=3 && commentaire.length()>0 && heureDefin >= 0 && heureDefin
                        <= 24 && i != null){
210 
211                         //Modification de l'intervention
212                         i.setStatus(status);
213                         i.setHeureDeFin(heureDefin);
214                         i.setCommentaire(commentaire);
215                         InterventionDaoJpa.modifierIntervention(i);
216 
217                         //libération de l'employe
218                         emp.setDispo(true);
219                         PersonneDaoJpa.modifierPersonne(emp);
220 
221                         //Envoi d'une notification au client
222                         String message = "Votre demande d'intervention a ete cloturee. Commentaire de l'
                                employe : " +commentaire;
223                         Message.envoyerNotification(i.getUnClient().getNom(), i.getUnClient().getPrenom()
                                , i.getUnClient().getTel(), message);
224 
225                          cloture=true;
226                 }
227                 JpaUtil.validerTransaction();
228                 JpaUtil.fermerEntityManager();
229                 return cloture;
230         }
231 
232         //Recherche de l'intervention en cours pour un employé donnée (s'il en a une)
233         public static Intervention rechercherInterventionEnCours(Employe emp){
234                 List<Intervention> list=emp.getListInterventions();
235                 for (Intervention i : list){
236                     if(i.getStatus()==0)
237                         return i;
238                 }
239                 return null;
240         }
241 
242         //Calcul des coordonnées GPS d'un employé ou d'un client
243         public static void calculerLatLng(Personne p){
244             LatLng GPS=GeoTest.getLatLng(p.getAdressePostale());
245             p.setLongitude(GPS.lng);
246             p.setLatitude(GPS.lat);
247         }
248 
249         //SERVICE EN PLUS DE CEUX SPECIFIES DANS LE COMPTE RENDU
250         //Récupération de l'heure actuelle (pour savoir à quel moment est faite une demande d'
                intervention)
251          public static int heureActuelleToInt(){
252             SimpleDateFormat format = new SimpleDateFormat ("HH:mm");
253             Date heureActuelle = new Date();
254             String heureString = format.format(heureActuelle);
255             String nbHeureString = heureString.split(":")[0];
256             int heure=Integer.parseInt(nbHeureString);
257             return heure;
258         }
259 }
```

vue.Main

```java
/* @author jorge terreu, corentin laharotte, chanèle jourdan
   Classe qui représente l'interface avec l'utilisateur
*/

package vue;

import dao.JpaUtil;
import dao.PersonneDaoJpa;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.util.List;
import metier.modele.Client;
import metier.modele.Employe;
import metier.modele.Intervention;
import metier.modele.Personne;
import metier.service.Service;
import util.Saisie;


public class Main {

   public final static String NOM_PERSISTENCE = "TPDASIPU";

   public static void main(String[] args) {
        Personne p=null;

        JpaUtil.init();

        //méthode à appeler une seule fois (pour remplir la base avec les employés)
       insererTousLesEmployes();

        lancerMenuPrincipal(p);
        JpaUtil.destroy();
    }

    //Création de la liste des employés de l'entreprise
    public static List<Employe> listDesEmployes(){
        List<Employe>l=new ArrayList<Employe>();
        l.add(new Employe(true,"BORROTI MATIAS DANTAS","Raphaël","motdepassepouremploye","8 Rue Arago, Villeurbanne","328178508","rborrotimatiasdantas4171@free.fr",true,10,20));
        l.add(new Employe(false,"OLMEADA MARAIS","Nor","motdepassepouremploye","5 Rue Léon Fabre, Villeurbanne","0418932546","nolmeadamarais1551@gmail.com",true,8,18));
        l.add(new Employe(false,"RAYES GEMEZ","Olena","motdepassepouremploye","12 Rue de la Prevoyance, Villeurbanne","0532731620","orayesgemez5313@outlook.com",true,6,16));
        l.add(new Employe(false,"SING","Ainhoa","motdepassepouremploye","4 Rue Phelypeaux, Villeurbanne","0705224200","asing8183@free.fr",true,5,15));
        l.add( new Employe(true,"ABDIULLINA","David Alexander","motdepassepouremploye","8 Rue Wilhelmine, Villeurbanne","0590232772","david-alexander.abdiullina@laposte.net",true,12,22));
        l.add(new Employe(true,"WOAGNER","Moez","motdepassepouremploye","6 Rue Camille Koechlin, Villeurbanne","0832205629","moez.woagner@laposte.net",true,7,17));
        l.add(new Employe(true,"HONRY","Matteo","motdepassepouremploye","9 Impasse Guillet, Villeurbanne","0482381862","matteo.honry@yahoo.com",true,5,15));
        l.add( new Employe(true,"CECCANI","Kevin","motdepassepouremploye","20 Rue Decomberousse, Villeurbanne","0664426037","kevin.ceccani@hotmail.com",true,8,18));
        l.add(new Employe(false,"VOYRET","Alice","motdepassepouremploye","1 Rue d'Alsace, Villeurbanne","0486856520","alice.voyret@hotmail.com",true,6,16));
        l.add(new Employe(true,"RINERD","Julien","motdepassepouremploye","4 Rue de la Jeunesse, Villeurbanne","0727252485","jrinerd5241@yahoo.com",true,9,19));
        return l;
    }

    //Permet d'insérer l'ensemble des employés dans la base de données
    public static void insererTousLesEmployes(){
        JpaUtil.creerEntityManager();
        JpaUtil.ouvrirTransaction();
        List<Employe>list=listDesEmployes();
        for (Employe e : list){
            Service.calculerLatLng(e);
            PersonneDaoJpa.creerPersonne(e);
        }
        JpaUtil.validerTransaction();
        JpaUtil.fermerEntityManager();
    }
```

```
66
67      //Permet d'afficher une liste d'intervention
68      //Sert pour l'affichage de l'historique d'un client et du tableau de bord d'un employé
69      public static void afficherListeInterventions (List<Intervention> listint){
70          if (!listint.isEmpty()){
71          System.out.println("|  Type  |  Description  |  Statut  |");
72                  for (Intervention i : listint){
73                       int type=i.getType();
74                       int status=i.getStatus();
75                       String typeString="";
76                       String statusString="";
77                      switch (type) {
78                      case 1:
79                          typeString="Animal";
80                          break;
81                      case 2:
82                          typeString="Livraison";
83                          break;
84                      case 3:
85                          typeString="Incident";
86                          break;
87                      default:
88                          break;
89                      }
90                      switch (status) {
91                      case 0:
92                          statusString="En Cours";
93                          break;
94                      case 1:
95                          statusString="Succès";
96                          break;
97                      case 2:
98                          statusString="Echec";
99                          break;
100                     default:
101                         break;
102                     }
103                     System.out.println ("| "+typeString+" | "+i.getDescription()+" | "+statusString+" | "
                            );
104                 }
105         }else{
106             System.out.println ("Pas d'intervention");
107         }
108     }
109
110
111 //Affichage des différents Menu (principal, client, employé)
112
113 public static void affichageMenuPrincipal(){
114     System.out.println("--------------------");
115     System.out.println("Bienvenue sur ProAct'IF");
116     System.out.println("--------------------");
117     System.out.println("Choisir une option:");
118     System.out.println("1.Se connecter");
119     System.out.println("2.S'inscrire");
120     System.out.println("3.Quitter");
121     System.out.println();
122  }
123
124  public static void affichageMenuClient(){
125     System.out.println();
126     System.out.println("Choisir une option:");
127     System.out.println("1.Demander une intervention");
128     System.out.println("2.Consulter Historique");
129     System.out.println("3.Retour");
130     System.out.println();
131  }
132
133   public static void affichageMenuEmploye(){
134     System.out.println();
135     System.out.println("Choisir une option:");
136     System.out.println("1.Cloturer l'intervention en cours");
137     System.out.println("2.Consulter Tableau de bord");
138     System.out.println("3.Retour");
139     System.out.println();
140  }
141
```

```
142
143     //Lancement des différents menus (principal, client, employé)
144     //Ils font automatiquement appel aux différents affichage et service selon la demande de l'
            utilisateur
145
146     public static void lancerMenuPrincipal(Personne p){
147         affichageMenuPrincipal();
148          int choix = Saisie.lireInteger("Choix:␣", Arrays.asList(1,2,3));
149          switch(choix){
150              case 1:
151              {
152                  String mail = Saisie.lireChaine("Mail␣:␣");
153                  String motDePasse = Saisie.lireChaine("Mot␣de␣passe␣:␣");
154                  p=Service.seConnecter(mail, motDePasse);
155                  if(p==null){
156                         System.out.println("Echec␣de␣la␣connexion␣\n\n");
157                         lancerMenuPrincipal(p);
158                 }
159                  if (p instanceof Employe){
160                         lancerMenuEmploye((Employe)p);
161                  } else if ( p instanceof Client){
162                         lancerMenuClient((Client)p);
163                  }
164                  break;
165              }
166              case 2:
167                  try{
168                  int madame = Saisie.lireInteger("Civilité␣(0=Mme,␣1=Mr)␣:␣",Arrays.asList(0,1));
169                  boolean civilite=false;
170                     if (madame==1){
171                         civilite=true;
172                     }
173                      String nom = Saisie.lireChaine("Nom␣:␣");
174                      String prenom = Saisie.lireChaine("Prenom␣:␣");
175                      String mdp = Saisie.lireChaine("Mot␣de␣passe␣:␣");
176                      String addpost = Saisie.lireChaine("Adresse␣postale␣:␣");
177                      String tel =Saisie.lireChaine("Tel␣:");
178                      String mail = Saisie.lireChaine("Mail␣:␣");
179
180                      SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
181                      String dateNaiss = Saisie.lireChaine("Date␣de␣naissance␣(dd/MM/yyyy)␣:␣");
182                      Date date = sdf.parse(dateNaiss);
183                      p=Service.seInscrire(civilite, nom, prenom, mdp, addpost, tel, mail, date);
184
185                      if(p==null){
186                             System.out.println("Erreur␣:␣L'inscription␣n'a␣pas␣pu␣être␣réalisée␣(champ␣
                                    vide␣ou␣mail␣déjà␣utilisé)␣\n\n");
187                             lancerMenuPrincipal(p);
188                      } else {
189                              System.out.println("Inscription␣réussie␣\n\n");
190                      }
191
192                      if (p instanceof Employe){
193                             lancerMenuEmploye((Employe)p);
194                      } else if ( p instanceof Client){
195                             lancerMenuClient((Client)p);
196                      }
197
198                   }catch(Exception e){
199                      System.out.println("Erreur␣:␣L'inscription␣n'a␣pas␣pu␣être␣réalisée␣(erreur␣sur␣la␣
                            date)␣\n\n");
200                      lancerMenuPrincipal(p);
201                  }
202
203                  break;
204              case 3:
205                     System.out.println("A␣bientôt");
206                     System.exit(0);
207          }
208      }
209
210     public static void lancerMenuEmploye(Employe e){
211              affichageMenuEmploye();
212              int choix = Saisie.lireInteger("Choix:␣", Arrays.asList(1,2,3));
213              switch(choix){
214                  case 1:
215                      if(Service.rechercherInterventionEnCours(e)!=null){
```

```
216                         int status = Saisie.lireInteger("Statut (1=Succès, 2=Echec) : ",Arrays.asList
                                (1,2));
217                         int heureDeFin = Saisie.lireInteger("Heure de fin : ");
218                         String commentaire = Saisie.lireChaine("Commentaire : ");
219                         boolean cloture=Service.cloturerIntervention(e,status,heureDeFin,commentaire);
220                         if(!cloture){
221                                 System.out.println("Erreur : Cloture de l'intervention échouée \n\n");
222                         } else {
223                                 System.out.println("Cloture de l'intervention réussie \n\n");
224                         }
225                     } else {
226                         System.out.println("Vous navez pas dintervention à clôturer \n\n");
227                     }
228                     lancerMenuEmploye(e);
229                     break;
230                 case 2:
231                     afficherListeInterventions(Service.recupererTableauDeBord(e));
232                     lancerMenuEmploye(e);
233                     break;
234                 case 3:
235                     lancerMenuPrincipal(e);
236                     break;
237             }
238         }
239 
240     public static void lancerMenuClient(Client c){
241             affichageMenuClient();
242             int choix = Saisie.lireInteger("Choix: ", Arrays.asList(1,2,3));
243             switch(choix){
244                 case 1:
245                     Employe tmp=null;
246                     int type = Saisie.lireInteger("Type (1=Animal, 2=Livraison, 3=Incident) : ",Arrays.
                            asList(1,2,3));
247                     switch (type) {
248                     case 1:
249                             String animal = Saisie.lireChaine("Animal : ");
250                             String description = Saisie.lireChaine("Description: ");
251                             tmp=Service.demanderIntervention(c,type,description,animal);
252                             break;
253                     case 2:
254                             String objet = Saisie.lireChaine("Objet : ");
255                             String entreprise = Saisie.lireChaine("Entreprise : ");
256                             String description1 = Saisie.lireChaine("Description: ");
257                             tmp=Service.demanderIntervention(c,type,description1,objet, entreprise);
258                             break;
259                     case 3:
260                             String description2= Saisie.lireChaine("Description: ");
261                             tmp=Service.demanderIntervention(c,type,description2);
262                             break;
263                     default:
264                         break;
265                     }
266 
267                     if(tmp==null){
268                         System.out.println("Erreur : la demande d'intervention n'est pas possible (champ
                                vide ou pas d'employe disponible)");
269                     } else {
270                         System.out.println("Demande dintervention validée");
271                     }
272                     lancerMenuClient(c);
273                     break;
274 
275                 case 2:
276                      afficherListeInterventions(Service.recupererHistorique(c));
277                      lancerMenuClient(c);
278                      break;
279 
280                 case 3:
281                     lancerMenuPrincipal(c);
282                     break;
283 
284                 default:
285                      break;
286             }
287       }
288 }
```