```java
package metier.modele;

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

@Entity
public class Activite implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String denomination;
    private Boolean parEquipe;
    private Integer nbParticipants;

    public Activite() {
    }

    public Activite(String denomination, Boolean parEquipe, Integer nbParticipants) {
        this.denomination = denomination;
        this.parEquipe = parEquipe;
        this.nbParticipants = nbParticipants;
    }

    public Long getId() {
        return id;
    }

    public String getDenomination() {
        return denomination;
    }

    public Boolean isParEquipe() {
        return parEquipe;
    }

    public Integer getNbParticipants() {
        return nbParticipants;
    }

    public void setDenomination(String denomination) {
        this.denomination = denomination;
    }

    public void setParEquipe(Boolean parEquipe) {
        this.parEquipe = parEquipe;
    }

    public void setNbParticipants(Integer nbParticipants) {
        this.nbParticipants = nbParticipants;
    }

    @Override
    public String toString() {
        return "Activite{" + "id=" + id + ", denomination=" + denomination +
                ", parEquipe=" + parEquipe + ", nbParticipants=" +
                nbParticipants + '}';
    }

}
```

```java
package dao;

import metier.modele.Activite;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import java.util.List;

public class ActiviteDao {

    public void create(Activite activite) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            em.persist(activite);
        }
        catch(Exception e) {
            throw e;
        }
    }

    public Activite update(Activite activite) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            activite = em.merge(activite);
        }
        catch(Exception e){
            throw e;
        }
        return activite;
    }

    public Activite findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Activite activite = null;
        try {
            activite = em.find(Activite.class, id);
        }
        catch(Exception e) {
            throw e;
        }
        return activite;
    }

    public Activite findByName(String name) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Activite activite = null;
        try {
            activite = em.find(Activite.class, name);
        }
        catch(Exception e) {
            throw e;
        }
        return activite;
    }

    public List<Activite> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Activite> activites = null;
        try {
            Query q = em.createQuery("SELECT a FROM Activite a");
            activites = (List<Activite>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return activites;
    }
}
```

```java
package metier.modele;

import com.google.maps.model.LatLng;

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


@Entity
public class Adherent implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String nom;
    private String prenom;
    private String mail;
    private String adresse;
    private Double longitude;
    private Double latitude;

    public Adherent() {
    }

    public Adherent(String nom, String prenom, String adresse, String mail) {
        this.nom = nom;
        this.prenom = prenom;
        this.mail = mail;
        this.adresse = adresse;
    }

    public Long getId() {
        return id;
    }

    public String getNom() {
        return nom;
    }

    public String getPrenom() {
        return prenom;
    }

    public String getMail() {
        return mail;
    }

    public String getAdresse() {
        return adresse;
    }

    public Double getLongitude() {
        return longitude;
    }

    public Double getLatitude() {
        return latitude;
    }

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

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

    public void setMail(String mail) {
        this.mail = mail;
    }
```

```
    public void setAdresse(String adresse) {
        this.adresse = adresse;
    }

    public void setCoordonnees(LatLng latLng) {
        this.longitude = latLng.lng;
        this.latitude = latLng.lat;
    }

    @Override
    public String toString() {
        return "Adherent{" + "id=" + id + ", nom=" + nom + ", prenom=" + prenom
                + ", mail=" + mail + ", adresse=" + adresse + ", longitude=" +
                longitude + ", latitude=" + latitude + '}';
    }
}
```

```java
package dao;

import metier.modele.Adherent;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import java.util.List;

public class AdherentDao {

    public void create(Adherent adherent) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            em.persist(adherent);
        }
        catch(Exception e) {
            throw e;
        }
    }

    public Adherent update(Adherent adherent) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            adherent = em.merge(adherent);
        }
        catch(Exception e){
            throw e;
        }
        return adherent;
    }

    public Adherent findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Adherent adherent = null;
        try {
            adherent = em.find(Adherent.class, id);
        }
        catch(Exception e) {
            throw e;
        }
        return adherent;
    }

    /**
     * @param mail adresse mail de l'adherent
     * @return un adherent si existant sinon null
     * @throws Throwable
     */
    public Adherent findByMail(String mail) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Adherent adherent = null;
        try {
            Query q = em.createQuery("SELECT a FROM Adherent a WHERE a.mail='"+mail+"'");
            if(q.getResultList().size() > 0) {
                adherent = ((List<Adherent>) q.getResultList()).get(0);
            }
        }
        catch(Exception e) {
            throw e;
        }
        return adherent;
    }

    public List<Adherent> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Adherent> adherents = null;
        try {
            Query q = em.createQuery("SELECT a FROM Adherent a");
            adherents = (List<Adherent>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
```

```
        }

        return adherents;
    }
}
```

```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.ManyToOne;
import javax.persistence.Temporal;

@Entity
public class Demande implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Temporal(javax.persistence.TemporalType.DATE)
    private Date dateEvenement;
    @Temporal(javax.persistence.TemporalType.DATE)
    private Date dateDemande;
    private Boolean traite;
    @ManyToOne
    private Adherent demandeur;
    @ManyToOne
    private Activite activite;

    public Demande() {
    }

    public Demande(Date dateEvenement, Date dateDemande, Adherent demandeur, Activite activite) {

        this.dateEvenement = dateEvenement;
        this.dateDemande = dateDemande;
        this.traite = false;
        this.demandeur = demandeur;
        this.activite = activite;
    }

    public Long getId() {
        return id;
    }

    public Date getDateEvenement() {
        return dateEvenement;
    }

    public Date getDateDemande() {
        return dateDemande;
    }

    public Boolean getTraite() {
        return traite;
    }

    public Adherent getDemandeur() {
        return demandeur;
    }

    public Activite getActivite() {
        return activite;
    }

    public void setDateEvenement(Date dateEvenement) {
        this.dateEvenement = dateEvenement;
    }

    public void setDateDemande(Date dateDemande) {
        this.dateDemande = dateDemande;
    }

    public void setTraite(Boolean traite) {
        this.traite = traite;
```

```
    }

    public void setDemandeur(Adherent demandeur) {
        this.demandeur = demandeur;
    }

    public void setActivite(Activite activite) {
        this.activite = activite;
    }

    @Override
    public String toString() {
        return "Demande{" + "id=" + id + ", Demandeur=" + demandeur +
                ", Activite=" + activite.getDenomination() + ", DateDemande=" +
                dateDemande + ", DateEvenement=" + dateEvenement + '}';
    }

}
```

```java
package dao;

import metier.modele.Activite;
import metier.modele.Adherent;
import metier.modele.Demande;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import java.util.Date;
import java.util.LinkedList;
import java.util.List;

public class DemandeDao {

    public void create(Demande demande) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            em.persist(demande);
        }
        catch(Exception e) {
            throw e;
        }
    }

    public Demande update(Demande demande) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            demande = em.merge(demande);
        }
        catch(Exception e){
            throw e;
        }
        return demande;
    }

    public Demande findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Demande demande = null;
        try {
            demande = em.find(Demande.class, id);
        }
        catch(Exception e) {
            throw e;
        }
        return demande;
    }

    public List<Demande> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Demande> demande = null;
        try {
            Query q = em.createQuery("SELECT a FROM Demande a");
            demande = (List<Demande>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return demande;
    }

    public List<Demande> findNotValidedByActiviteeAndByDate(Activite activite, Date demandeDate) throws
Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Demande> demande = null;
        try {
            Query q = em.createQuery("SELECT a FROM Demande a WHERE a.activite.id="+activite.getId());
            demande = (List<Demande>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }
```

```
        System.out.println("demandes trouvees : "+demande.size());
        List<Demande> ret = new LinkedList<>();
        for (Demande d:demande) {
            if(/*d.getActivite().equals(activite) &&*/ d.getTraite().equals(false) &&
d.getDateEvenement().getDate()==demandeDate.getDate() &&
                    d.getDateEvenement().getMonth()==demandeDate.getMonth() && d.getDateEvenement
().getYear()==demandeDate.getYear())
            {
                ret.add(d);
            }
            else{
                System.out.println("traitee: "+d.getTraite()+" date: "+d.getDateEvenement()+" !=
"+demandeDate);
            }
        }
        System.out.println("demande selectionnee: "+ret.size());
        return ret;
    }

    public List<Demande> findByAdherentOrderByDate(Adherent adherent) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Demande> demande = null;
        try {
            Query q = em.createQuery("SELECT a FROM Demande a WHERE a.demandeur.id = "+adherent.getId()
+" ORDER BY a.dateDemande ASC");
            demande = (List<Demande>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }
        /*List<Demande> ret = new LinkedList<>();
        for (Demande d:demande) {
            if(d.getDemandeur().equals(adherent))
            {
                ret.add(d);
            }
        }

        return ret;*/
        return demande;
    }

    public List<Demande> findByAdherentOrderByDateDesc(Adherent adherent) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Demande> demande = null;
        try {
            Query q = em.createQuery("SELECT a FROM Demande a WHERE a.demandeur.id = "+adherent.getId()
+" ORDER BY a.dateEvenement DESC");
            demande = (List<Demande>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }
        /*List<Demande> ret = new LinkedList<>();
        for (Demande d:demande) {
            if(d.getDemandeur().equals(adherent))
            {
                ret.add(d);
            }
        }

        return ret;*/
        return demande;
    }

    public List<Demande> findAllOrderByActivite() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Demande> ret = null;
        try {
            Query q = em.createQuery("SELECT a FROM Demande a ORDER BY a.activite.denomination ASC");
            ret = (List<Demande>) q.getResultList();
        }
```

```
        catch(Exception e) {
            throw e;
        }
        return ret;
    }

    public List<Demande> findAllOrderByActiviteDesc() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Demande> ret = null;
        try {
            Query q = em.createQuery("SELECT a FROM Demande a ORDER BY a.activite.denomination DESC");
            ret = (List<Demande>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }
        return ret;
    }
}
```

```java
package metier.modele;

import java.util.Date;
import java.util.List;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.ManyToMany;

/**
 *
 * @author pllefebvre
 */
@Entity
public class Evenement1equipe extends Evenement {

    @ManyToMany
    @ElementCollection
    private List<Adherent> listeAdherents;

    public Evenement1equipe() {
        super();
    }

    public Evenement1equipe(Date date, Activite activite, List<Adherent> listeAdherents) {
        super(date, activite);
        this.listeAdherents = listeAdherents;
    }

    public List<Adherent> getListeAdherents() {
        return listeAdherents;
    }

    public void setListeAdherents(List<Adherent> listeAdherents) {
        this.listeAdherents = listeAdherents;
    }
}
```

```java
package dao;

import metier.modele.Evenement;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import java.util.List;

public class EvenementDao {

    public void create(Evenement evenement) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            em.persist(evenement);
        }
        catch(Exception e) {
            throw e;
        }
    }

    public Evenement update(Evenement evenement) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            evenement = em.merge(evenement);
        }
        catch(Exception e){
            throw e;
        }
        return evenement;
    }

    public Evenement findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Evenement evemenent = null;
        try {
            evemenent = em.find(Evenement.class, id);
        }
        catch(Exception e) {
            throw e;
        }
        return evemenent;
    }

    public List<Evenement> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Evenement> evenement = null;
        try {
            Query q = em.createQuery("SELECT a FROM Evenement a");
            evenement = (List<Evenement>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return evenement;
    }

    public List<Evenement> findSansLieu() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Evenement> evenement = null;
        try {
            Query q = em.createQuery("SELECT a FROM Evenement a WHERE a.lieu IS NULL");
            evenement = (List<Evenement>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return evenement;
    }

    /**
```

```java
     * Methode renvoyant la liste des evenements affectes a un lieu donne, utlisee
     * pour determiner si un lieu a deja ete affecte a un evenement
     * @param idLieu l'id du lieu
     * @return la lsite d'evenements en question
     */

    public List<Evenement> findByIdLieu(Long idLieu) {

        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Evenement> evenement = null;
        try {
            Query q = em.createQuery("SELECT a FROM Evenement a WHERE a.lieu.id = " + idLieu + "");
            evenement = (List<Evenement>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return evenement;
    }
}
```

```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.ManyToOne;

import javax.persistence.Temporal;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Evenement implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Temporal(javax.persistence.TemporalType.DATE)
    protected Date dateEvenement;
    @ManyToOne
    protected Activite activite;
    @ManyToOne
    protected Lieu lieu;

    public Evenement() {
    }

    public Evenement(Date date, Activite activite) {

        this.dateEvenement = date;
        this.activite = activite;
    }

    public Long getId() {
        return id;
    }

    public Date getDateEvenement() {
        return dateEvenement;
    }

    public Lieu getLieu() {
        return lieu;
    }

    public Activite getActivite() {
        return activite;
    }

    public void setDateEvenement(Date dateEvenement) {
        this.dateEvenement = dateEvenement;
    }

    public void setActivite(Activite activite) {
        this.activite = activite;
    }

    public void setLieu(Lieu lieu) {
        this.lieu = lieu;
    }

     @Override
    public String toString() {
        return "Evenement{" + "id=" + id +  ", Activite=" +
                activite.getDenomination() + ", DateEvenement=" + dateEvenement
                + '}';
    }
}
```

```java
package metier.modele;

import java.util.Date;
import java.util.List;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.ManyToMany;

/**
 *
 * @author pllefebvre
 */
@Entity
public class Evenement1equipe extends Evenement {

    @ManyToMany
    @ElementCollection
    private List<Adherent> listeAdherents;

    public Evenement1equipe() {
        super();
    }

    public Evenement1equipe(Date date, Activite activite, List<Adherent> listeAdherents) {
        super(date, activite);
        this.listeAdherents = listeAdherents;
    }

    public List<Adherent> getListeAdherents() {
        return listeAdherents;
    }

    public void setListeAdherents(List<Adherent> listeAdherents) {
        this.listeAdherents = listeAdherents;
    }
}
```

```java
package metier.modele;

import java.util.Date;
import java.util.List;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

/**
 *
 * @author pllefebvre
 */
@Entity
public class Evenement2equipes extends Evenement {

    @ManyToMany
    @JoinTable(name = "ListeEquipeA")
    @ElementCollection
    private List<Adherent> equipeA;
    @ManyToMany
    @JoinTable(name = "ListeEquipeB")
    @ElementCollection
    private List<Adherent> equipeB;

    public Evenement2equipes() {
        super();
    }

    public Evenement2equipes(Date date, Activite activite, List<Adherent> listeEquipeA, List<Adherent>
listeEquipeB) {
        super(date, activite);
        this.equipeA = listeEquipeA;
        this.equipeB = listeEquipeB;
    }

    public List<Adherent> getListeEquipeA() {
        return equipeA;
    }

    public List<Adherent> getListeEquipeB() {
        return equipeB;
    }

    public void setListeEquipeA(List<Adherent> listeEquipeA) {
        this.equipeA = listeEquipeA;
    }

    public void setListeEquipeB(List<Adherent> listeEquipeB) {
        this.equipeB = listeEquipeB;
    }
}
```

```java
package dao;

import metier.modele.Evenement;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import java.util.List;

public class EvenementDao {

    public void create(Evenement evenement) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            em.persist(evenement);
        }
        catch(Exception e) {
            throw e;
        }
    }

    public Evenement update(Evenement evenement) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            evenement = em.merge(evenement);
        }
        catch(Exception e){
            throw e;
        }
        return evenement;
    }

    public Evenement findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Evenement evemenent = null;
        try {
            evemenent = em.find(Evenement.class, id);
        }
        catch(Exception e) {
            throw e;
        }
        return evemenent;
    }

    public List<Evenement> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Evenement> evenement = null;
        try {
            Query q = em.createQuery("SELECT a FROM Evenement a");
            evenement = (List<Evenement>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return evenement;
    }

    public List<Evenement> findSansLieu() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Evenement> evenement = null;
        try {
            Query q = em.createQuery("SELECT a FROM Evenement a WHERE a.lieu IS NULL");
            evenement = (List<Evenement>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return evenement;
    }

    /**
```

```
     * Methode renvoyant la liste des evenements affectes a un lieu donne, utlisee
     * pour determiner si un lieu a deja ete affecte a un evenement
     * @param idLieu l'id du lieu
     * @return la lsite d'evenements en question
     */

    public List<Evenement> findByIdLieu(Long idLieu) {

        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Evenement> evenement = null;
        try {
            Query q = em.createQuery("SELECT a FROM Evenement a WHERE a.lieu.id = " + idLieu + "");
            evenement = (List<Evenement>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return evenement;
    }
}
```

```java
package dao;



import java.util.logging.Level;
import java.util.logging.Logger;
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 = "fr.insalyon.dasi_TP-COLLECTIF-B3129_jar_1.0-
SNAPSHOTPU";
    /**
     * 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 = Persistence.createEntityManagerFactory
(PERSISTENCE_UNIT_NAME);
    /**
     * 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;
        }
    };

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

    private static void log(String message) {
//        pause(5);
//        System.err.println(message);
//        pause(5);
    }


    /**
     * Créée l'instance courante de Entity Manager (liée à ce Thread).
     * <br/><strong>À utiliser uniquement au niveau Service.</strong>
```

```
     */
    public static void creerEntityManager() {
        log("création du contexte de persistance");
        threadLocalEntityManager.set(entityManagerFactory.createEntityManager());
    }

    /**
     * Ferme l'instance courante de Entity Manager (liée à ce Thread).
     * <br/><strong>À utiliser uniquement au niveau Service.</strong>
     */
    public static void fermerEntityManager() {
        log("fermeture du contexte de persistance");
        EntityManager em = threadLocalEntityManager.get();
        em.close();
        threadLocalEntityManager.set(null);
    }

    /**
     * Démarre une transaction sur l'instance courante de Entity Manager.
     * <br/><strong>À utiliser uniquement au niveau Service.</strong>
     */
    public static void ouvrirTransaction() {
        log("debut transaction");
        try {
            EntityManager em = threadLocalEntityManager.get();
            em.getTransaction().begin();
        } catch (Exception ex) {
            Logger.getLogger(JpaUtil.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * Valide la transaction courante sur l'instance courante de Entity Manager.
     * <br/><strong>À utiliser uniquement au niveau Service.</strong>
     *
     * @exception RollbackException lorsque le <em>commit</em> n'a pas réussi.
     */
    public static void validerTransaction() throws RollbackException {
        log("commit transaction");
        try {
            EntityManager em = threadLocalEntityManager.get();
            em.getTransaction().commit();
        } catch (Exception ex) {
            Logger.getLogger(JpaUtil.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * Annule la transaction courante sur l'instance courante de Entity Manager.
     * Si la transaction courante n'est pas démarrée, cette méthode n'effectue
     * aucune opération.
     * <br/><strong>À utiliser uniquement au niveau Service.</strong>
     */
    public static void annulerTransaction() {
        try {
            log("rollback transaction");

            EntityManager em = threadLocalEntityManager.get();
            if (em.getTransaction().isActive()) {
                log("rollback transaction effectue");
                em.getTransaction().rollback();
            }

        } catch (Exception ex) {
            Logger.getLogger(JpaUtil.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * Retourne l'instance courante de Entity Manager.
     * <br/><strong>À utiliser uniquement au niveau DAO.</strong>
     *
```

```
     * @return instance de Entity Manager
     */
    protected static EntityManager obtenirEntityManager() {
        log("obtention du contexte de persistance");
        return threadLocalEntityManager.get();
    }
}
```

```java
package metier.modele;

import com.google.maps.model.LatLng;

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

@Entity
public class Lieu implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String denomination;
    private String description;
    private String adresse;
    private Double longitude;
    private Double latitude;

    public Lieu() {
    }
    public Lieu(String denomination, String description, String adresse) {
        this.denomination = denomination;
        this.description = description;
        this.adresse = adresse;
    }


    public Long getId() {
        return id;
    }

    public String getDenomination() {
        return denomination;
    }

    public String getDescription() {
        return description;
    }

    public String getAdresse() {
        return adresse;
    }

    public Double getLongitude() {
        return longitude;
    }

    public Double getLatitude() {
        return latitude;
    }

    public void setCoordonnees(LatLng latLng) {
        this.longitude = latLng.lng;
        this.latitude = latLng.lat;
    }

    public void setDenomination(String denomination) {
        this.denomination = denomination;
    }


    public void setDescription(String description) {
        this.description = description;
    }

    public void setAdresse(String adresse) {
        this.adresse = adresse;
    }
```

```
    public void setLongitude(Double longitude) {
        this.longitude = longitude;
    }

    public void setLatitude(Double latitude) {
        this.latitude = latitude;
    }

    @Override
    public String toString() {
        return "Lieu{" + "id=" + id + ", denomination=" + denomination + ", description=" +
description + ", adresse=" + adresse + ", longitude=" + longitude + ", latitude=" + latitude + '}';
    }


}
```

```java
package dao;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import metier.modele.Lieu;

public class LieuDao {

    public void create(Lieu lieu) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            em.persist(lieu);
        }
        catch(Exception e) {
            throw e;
        }
    }

    public Lieu update(Lieu lieu) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        try {
            lieu = em.merge(lieu);
        }
        catch(Exception e){
            throw e;
        }
        return lieu;
    }

    public Lieu findById(long id) throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        Lieu lieu = null;
        try {
            lieu = em.find(Lieu.class, id);
        }
        catch(Exception e) {
            throw e;
        }
        return lieu;
    }

    public List<Lieu> findAll() throws Throwable {
        EntityManager em = JpaUtil.obtenirEntityManager();
        List<Lieu> lieux = null;
        try {
            Query q = em.createQuery("SELECT l FROM Lieu l");
            lieux = (List<Lieu>) q.getResultList();
        }
        catch(Exception e) {
            throw e;
        }

        return lieux;
    }
}
```

```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 metier.service;

import dao.*;
import metier.modele.*;

import java.util.Date;
import java.util.LinkedList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author pllefebvre
 */

public class ServiceMetier {

    private final ActiviteDao activiteDao = new ActiviteDao();
    private final AdherentDao adherentDao = new AdherentDao();
    private final LieuDao lieuDao = new LieuDao();
    private final DemandeDao demandeDao = new DemandeDao();
    private final EvenementDao evenementDao = new EvenementDao();

    /**
     * Methode permettant d'ajouter un adherent a la base, seulement si celui-ci
     * identife par son adresse mail n'existe pas deja dans la base de donne, gere
     * aussi l'envoi des mails d'infirmation ou de confirmation d'inscription
     * @param adherent l'adherent a ajouter
     * @return 1 si l'adherent a ete correctement ajoute 0 sinon
     */
    public int creerAdherent(Adherent adherent) {

        ServiceTechnique.majCoordonnees(adherent);

        try {
            JpaUtil.creerEntityManager();

            try {
                JpaUtil.ouvrirTransaction();

                if(adherentDao.findByMail(adherent.getMail()) == null) {

                    adherentDao.create(adherent);
                    JpaUtil.validerTransaction();
                    ServiceTechnique.mailConfirmationInscriptionAdherent(adherent);
                    ServiceTechnique.mailConfirmationInscriptionResponsable(adherent);
                    return 1;

                } else {

                    ServiceTechnique.mailInfirmationInscriptionAdherent(adherent);
                    ServiceTechnique.mailInfirmationInscriptionResponsable(adherent);
                    JpaUtil.annulerTransaction();
                }

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);

            }

            JpaUtil.fermerEntityManager();

        } catch (Exception e) {

            System.err.println("entiyManager creation error");
```

```
            e.printStackTrace();
        }

        return 0;
    }

    /**
     * Methode permettant d'ajouter une activite a la base
     * @param activite l'activite a ajouter
     */
    public void creerActivite(Activite activite) {

        try {
            JpaUtil.creerEntityManager();

            try {
                JpaUtil.ouvrirTransaction();
                activiteDao.create(activite);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }

            JpaUtil.fermerEntityManager();

        } catch (Exception e) {

            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
    }

    /**
     * Methode permettant d'ajouter un lieu a la base
     * @param lieu l'adherent a ajouter
     */
    public void creerLieu(Lieu lieu) {

        ServiceTechnique.majCoordonnees(lieu);

        try {
            JpaUtil.creerEntityManager();

            try {
                JpaUtil.ouvrirTransaction();
                lieuDao.create(lieu);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }

            JpaUtil.fermerEntityManager();

        } catch (Exception e) {

            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
    }

    /**
     * Ajoute a la base de donnee une demande
     * @param demande la demande ajoute a la base de donnee
     */
    public void creerDemande(Demande demande) {

        try {
            JpaUtil.creerEntityManager();
```

```
            try {
                JpaUtil.ouvrirTransaction();
                demandeDao.create(demande);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }

            JpaUtil.fermerEntityManager();
            //essaye de creer un evenement:
            rechercherEtCreerEvenement(demande.getActivite(),demande.getDateEvenement());

            // lecture au clavier pour suspendre temporairement l'execution
            System.out.println("Appyuer sur un chiffre puis ENTER pour poursuivre\n"
                    + "la tentative de creation d'un evenement");
            ServiceTest.nextInt();
            // essaye de creer un evenement:
            rechercherEtCreerEvenement(demande.getActivite(),demande.getDateEvenement());

        } catch (Exception e) {

            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
    }

    /**
     * Ajoute a la base de donnee un evenement
     * @param evenement la demande ajoute a la base de donnee
     */
    public void creerEvenement(Evenement evenement) {

        try {
            JpaUtil.creerEntityManager();

            try {
                JpaUtil.ouvrirTransaction();
                evenementDao.create(evenement);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }

            JpaUtil.fermerEntityManager();

        } catch (Exception e) {

            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
    }


    /**
     * Renvoie l'evenemnt corespondant a l'ID fournie
     * @param id le numero de l'evenement demande
     * @return l'evenemnt corespondant a l'ID fournie
     */
    public Evenement trouverEvenement(long id){
        Evenement ret=null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = evenementDao.findById(id);
                JpaUtil.validerTransaction();
```

```
            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Renvoie le lieu corespondant a l'ID fournie
     * @param id le numero du lieu demande
     * @return le lieu corespondant a l'ID fournie
     */
    public Lieu trouverLieu(long id){
        Lieu ret=null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = lieuDao.findById(id);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Renvoie l'adherent corespondant au mail fournie
     * @param mail mail de l'adherent demande
     * @return l'adherent corespondant au mail fournie
     */
    public Adherent trouverAdherent(String mail){
        Adherent ret=null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = adherentDao.findByMail(mail);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Renvoie l'adherent corespondant a l'ID fournie
     * @param id le numero de l'adherent demande
     * @return l'adherent corespondant a l'ID fournie
     */
    public Adherent trouverAdherent(long id){
        Adherent ret=null;
        try {
```

```
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = adherentDao.findById(id);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Renvoie l'activite corespondant au nom fournie
     * @param nom le nom de l'adherent demande
     * @return l'activite corespondant au nom fournie
     */
    public Activite trouverActivite(String nom)
    {
        Activite ret=null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = activiteDao.findByName(nom);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Renvoie la liste complete des activitees presante dans la base de donnee
     * @return la liste complete des activitees presante dans la base de donnee
     */
    public List<Activite> obtenirActivitees()
    {
        List<Activite> ret =null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = activiteDao.findAll();
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
```

```
     * Renvoie la liste des demandes de l'adherent donnee en parametre, trier par ordre de date
croisante
     * @param adherent l'adherent conserne par la domande
     * @return la liste des demandes de l'adherent donnee en parametre, trier par ordre de date
croisante
     */
    public List<Demande> obtenirDemandesTrierParDatePourAdherent(Adherent adherent)
    {
        List<Demande> ret =null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = demandeDao.findByAdherentOrderByDate(adherent);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Renvoie la liste des demandes de l'adherent donnee en parametre, trier par ordre de date
decroisante
     * @param adherent l'adherent conserne par la domande
     * @return la liste des demandes de l'adherent donnee en parametre, trier par ordre de date
decroisante
     */
    public List<Demande> afficherDemandesTrierParDateDescPourAdherent(Adherent adherent)
    {
        List<Demande> ret =null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = demandeDao.findByAdherentOrderByDateDesc(adherent);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Renvoie la liste des demandes contenue dans la base de donnee, trier par nom d'activitee
croissant
     * @return la liste des demandes contenue dans la base de donnee, trier par nom d'activitee
croissant
     */
    public List<Demande> obtenirDemandesTrierParActivitee()
    {
        List<Demande> ret =null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = demandeDao.findAllOrderByActivite();
                JpaUtil.validerTransaction();
```

```
            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Renvoie la liste des demandes contenue dans la base de donnee, trier par nom d'activitee
decroissant
     * @return la liste des demandes contenue dans la base de donnee, trier par nom d'activitee
decroissant
     */
    public List<Demande> afficherDemandesTrierParActiviteeDesc()
    {
        List<Demande> ret =null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = demandeDao.findAllOrderByActiviteDesc();
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }

    /**
     * Methode renvoye une liste de tous les evenements affectes a aucun lieu
     * @return liste de tous les evenements affectes a aucun lieu
     */
    public List<Evenement> obtenirEvenementSansLieu() {

        List<Evenement> listeEvenements = null;

        try {
            JpaUtil.creerEntityManager();

            try {
                JpaUtil.ouvrirTransaction();
                listeEvenements = evenementDao.findSansLieu();
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }

            JpaUtil.fermerEntityManager();

        } catch (Exception e) {

            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
            return listeEvenements;
    }
```

```
    /**
     * Renvoie l'evenement creer si la creation a eu lieu, null sinon
     * @param activite activitee a chercher
     * @param date date de l'activitee
     * @return l'evenement creer si la creation a eu lieu, null sinon
     */
    public Evenement rechercherEtCreerEvenement(Activite activite, Date date){
        System.out.println("tentative de creation eve. act: "+activite.getDenomination()+" date:
"+date);
        Evenement event = null;
        try {
            JpaUtil.creerEntityManager();

            try {
                JpaUtil.ouvrirTransaction();
                List<Demande> demandes = demandeDao.findNotValidedByActiviteeAndByDate(activite, date);
                if(activite.getNbParticipants() <= demandes.size() ) {
                    LinkedList<Adherent> equipeA = new LinkedList<>();
                    LinkedList<Adherent> equipeB = new LinkedList<>();
                    //repartie les demendeurs dans les equipes (si plusieurs equipes)
                    for(Demande d:demandes) {
                        Adherent a=d.getDemandeur();
                        if(equipeA.size() <= equipeB.size() || !activite.isParEquipe()) {
                            equipeA.add(a);
                        }
                        else {
                            equipeB.add(a);
                        }
                    }
                    if(activite.isParEquipe()) {
                        event = new Evenement2equipes(date,activite,equipeA,equipeB);
                    }
                    else{
                        event = new Evenement1equipe(date,activite,equipeA);
                    }
                    evenementDao.create(event);
                }
                else{
                    System.out.println("nb participant: "+activite.getNbParticipants()+" nb demande:
"+demandes.size());
                }
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }

            JpaUtil.fermerEntityManager();

        } catch (Exception e) {

            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return event;
    }

    /**
     * Affecte un lieu a un evenement et envoye le mail correspondant
     * @param lieu le lieu a affecter
     * @param evenement l'evenement en question
     */
    public void affecterLieuAEvenement(Lieu lieu, Evenement evenement) {

        evenement.setLieu(lieu);

         try {
            JpaUtil.creerEntityManager();

            try {
                JpaUtil.ouvrirTransaction();
```

```
                evenementDao.update(evenement);
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }

            JpaUtil.fermerEntityManager();

            ServiceTechnique.mailEvenement(evenement);

        } catch (Exception e) {

            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
    }

    /**
     * Renvoie la liste des Lieux contenue dans la base de donnee
     * @return la liste des Lieux contenue dans la base de donnee
     */
    public List<Lieu> obtenirLieux() {

        List<Lieu> ret =null;
        try {
            JpaUtil.creerEntityManager();
            try {
                JpaUtil.ouvrirTransaction();
                ret = lieuDao.findAll();
                JpaUtil.validerTransaction();

            } catch (Throwable ex) {
                JpaUtil.annulerTransaction();
                Logger.getLogger(ServiceMetier.class.getName()).log(Level.SEVERE, null, ex);
            }
            JpaUtil.fermerEntityManager();
        } catch (Exception e) {
            System.err.println("entiyManager creation error");
            e.printStackTrace();
        }
        return ret;
    }


}
```

```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 metier.service;

import com.google.maps.GeoApiContext;
import com.google.maps.GeocodingApi;
import com.google.maps.model.GeocodingResult;
import com.google.maps.model.LatLng;
import metier.modele.*;

import java.util.List;

/**
 *
 * @author pllefebvre
 */
public abstract class ServiceTechnique {

    final static GeoApiContext CONTEXTE_GEOAPI =
            new GeoApiContext().setApiKey("AIzaSyAhf3JleYpal9S-xouJYH8lf7Dvz5Y2Nko");

    /**
     * Methode de recuperation de la latitude et longitude
     * @param adresse a localiser
     * @return latitude et longitude
     */
    private static LatLng getLatLng(String adresse) {

        try {
            GeocodingResult[] results =
                    GeocodingApi.geocode(CONTEXTE_GEOAPI,adresse).await();
            return results[0].geometry.location;

        } catch (Exception ex) {

            ex.printStackTrace();
            return null;
        }
    }

    /**
     * Methode de mise a jour de la latitude et longitude d'un Lieu
     * @param lieu a localiser puis mettre a jour
     */
    public static void majCoordonnees(Lieu lieu) {

        lieu.setCoordonnees(getLatLng(lieu.getAdresse()));
    }

    /**
     * Methode de mise a jour de la latitude et longitude d'un Adherent
     * @param adherent a localiser puis mettre a jour
     */
    public static void majCoordonnees(Adherent adherent) {

        adherent.setCoordonnees(getLatLng(adherent.getAdresse()));
    }

    /**
     * Affiche sur la console le mail que l'adherent recoit en cas de
     * confirmation de son inscription
     * @param adherent l'adherent en question
     */
    public static void mailConfirmationInscriptionAdherent(Adherent adherent) {

        System.out.println("Expediteur : collectif@collectif.org\n"
                + "Pour : " + adherent.getMail() + ",\n"
                + "Sujet : Bienvenue chez Collect'IF\n"
                + "\n"
```

```
                + "Bonjour " + adherent.getNom() + "\n"
                + "Nous vous confirmons votre adhesion a l'assocation COLLECT'IF."
                + "Votre numero d'adherent est : 2450"
                + ".\n\n");
    }

    /**
     * Affiche sur la console le mail que le responsable recoit en cas de
     * confirmation de l'inscription d'un adherent passe en parametre
     * @param adherent l'adherent dont l'inscription a ete confirmee
     */
    public static void mailConfirmationInscriptionResponsable(Adherent adherent) {

        System.out.println("Expediteur : collectif@collectif.org\n"
                + "Pour : responsable@collectif.com\n"
                + "Sujet : confirmation d'incription de l'adherent : "
                + adherent.getMail()
                + "\n\n"
                + "Madame, Monsieur,\n"
                + "Nous confirmons l'inscription a notre service Collect'IF de \n"
                + "l'adherent : "+ adherent.getPrenom() +" "+ adherent.getNom()
                + "a l'adresse mail : " + adherent.getMail() + ".\n\n");
    }

    /**
     * Affiche sur la console le mail que l'adherent recoit en cas d'infirmation
     * de son inscription
     * @param adherent l'adherent en question
     */
    public static void mailInfirmationInscriptionAdherent(Adherent adherent) {

        System.out.println("Expediteur : collectif@collectif.org\n"
                + "Pour : " + adherent.getMail() + ",\n"
                + "Sujet : Bienvenue chez Collect'IF\n"
                + "\n"
                + "Bonjour " + adherent.getNom() + "\n"
                + "Votre adhesion a l'assocation COLLECT'IF a malencontreusement "
                + "echoue... Merci de recommencer ulterieurement"
                + ".\n\n");
    }

    /**
     * Affiche sur la console le mail que le responsable recoit en cas
     * d'infirmation de l'inscription d'un adherent passe en parametre
     * @param adherent l'adherent dont l'inscription a ete infirmee
     */
    public static void mailInfirmationInscriptionResponsable(Adherent adherent) {

        System.out.println("Expediteur : collectif@collectif.org\n"
                + "Pour : responsable@collectif.com\n"
                + "Sujet : confirmation d'incription de l'adherent : "
                + adherent.getMail()
                + "\n\n"
                + "Madame, Monsieur,\n"
                + "L'inscription a notre service Collect'IF de \n"
                + "l'adherent : "+ adherent.getPrenom() +" "+ adherent.getNom()
                + "a l'adresse mail : " + adherent.getMail() + " a echoue.\n\n");
    }
    /**
     * Affiche sur la console le mail que le responsable envoie en cas
     * de creation d'un evenement passe en parametre
     * @param evenement l'evenement venant d'etre cree
     */
    public static void mailEvenement(Evenement evenement) {

        System.out.print( "Expediteur : collectif@collectif.org\n"
                + "Pour : ");
        for(Adherent adherent : obtenirAdherentAEvenement(evenement))
            System.out.print(adherent.getMail() + ", ");
        System.out.print( "\nSujet : Nouvel Evenement Collect'IF\n"
                + "\n"
                + "Bonjour ");
```

```
        for(Adherent adherent : obtenirAdherentAEvenement(evenement))
            System.out.print(adherent.getNom() + ", ");
        System.out.println( "\nComme vous l'aviez souhaite, COLLECT'IF organise \n"
                + "un evenement de " + evenement.getActivite().getDenomination()
                + "le " + evenement.getDateEvenement() + ". Vous trouverez ci-dessous \n"
                + "les details de cet evenement. \n\n"
                + "Associativement vôtre,\n\n"
                + "Le Responsable de l'association\n\n"
                + "Evenement " + evenement.getActivite().getDenomination() + "\n"
                + "Date : " + evenement.getDateEvenement() + "\n"
                + "Lieu : " + evenement.getLieu().getDenomination() + ", "
                + evenement.getLieu().getAdresse());
    }

    /**
      * Methode renvoyant tous les adherent a un evenement donne
      * @param evenement l'enevenement en question
      * @return la liste de tous les adherents
      */
     private static List<Adherent> obtenirAdherentAEvenement(Evenement evenement) {

        if(evenement instanceof Evenement1equipe) {

            return ((Evenement1equipe) evenement).getListeAdherents();
        }

        List<Adherent> ret = ((Evenement2equipes) evenement).getListeEquipeA();
        ret.addAll(((Evenement2equipes) evenement).getListeEquipeB());
        return ret;
     }
}
```

```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 metier.service;

import java.util.Scanner;

/**
 *
 * @author pllefebvre
 */
public abstract class ServiceTest {

    private static final Scanner sc = new Scanner(System.in);

    public static boolean hasNextInt() {

        return sc.hasNextInt();
    }

    public static String next() {

        return sc.next();
    }

    public static String nextLine() {

        return sc.nextLine();
    }

    public static long nextLong() {

        return sc.nextLong();
    }

    public static int nextInt() {

        return sc.nextInt();
    }
}
```