```java
package main.java.boundary;

import main.java.controller.ControllerProposerVente;
import main.java.controller.ControllerVerifierIdentifiant;
import main.java.metier.Fournisseur;
import main.java.metier.Produit;
import main.java.metier.Vente;

import java.net.URI;
import java.util.Date;
import java.util.Scanner;

/**
 * Classe BoundaryProposerVente.
 * Interface utilisateur permettant de gérer les interactions liées à la
 * proposition de vente.
 * Cette classe facilite la communication entre le fournisseur et les
 * contrôleurs.
 */
public class BoundaryProposerVente {

    private ControllerProposerVente proposerVente = new ControllerProposerVente();
    private ControllerVerifierIdentifiant verifierIdentifiant = new ControllerVerifierIdentifiant();
    private Scanner scanner = new Scanner(System.in);

    /**
     * Permet à un fournisseur de proposer une vente.
     * Cette méthode demande à l'utilisateur les détails de la vente et les produits
     * associés.
     *
     * @param fournisseur L'objet fournisseur qui propose la vente.
     * @throws Exception Si une erreur survient lors de la création des URI ou des
     *                   données saisies.
     */
    public void proposerVente(Fournisseur fournisseur) throws Exception {
        System.out.println("Nom de la vente : ");
        String nomVente = scanner.nextLine();

        Vente vente = proposerVente.creerVente(nomVente, new Date(), new Date());

        System.out.println("Ajout de produits : ");
        while (true) {
            System.out.print("Nom du produit (stop pour terminer) : ");
            String nomProduit = scanner.nextLine();
            if (nomProduit.equalsIgnoreCase("stop"))
                break;

            System.out.print("Prix du produit : ");
            double prix = scanner.nextDouble();

            System.out.print("Quantité : ");
            int quantite = scanner.nextInt();
            scanner.nextLine();

            vente.ajouterProduit(new Produit(nomProduit, prix, new URI("http://example.com")), quantite);
        }

        fournisseur.ajouterVente(vente);
        System.out.println("Vente ajoutée avec succès !");
    }
}
```

```java
package main.java.controller;

import main.java.metier.Vente;

import java.util.Date;
```

```java
/**
 * Classe ControllerProposerVente.
 * Gère la création des ventes en contrôlant les données fournies.
 */
public class ControllerProposerVente {

    /**
     * Crée une vente avec les informations fournies.
     * Cette méthode encapsule la logique de création de la vente.
     *
     * @param nom       Le nom de la vente.
     * @param dateDebut La date de début de la vente.
     * @param dateFin   La date de fin de la vente.
     * @return Une instance de {@link Vente} avec les détails spécifiés.
     */
    public Vente creerVente(String nom, Date dateDebut, Date dateFin) {
        return new Vente(nom, dateDebut, dateFin);
    }
}
```

```java
package main.java.controller;

import main.java.metier.Fournisseur;

/**
 * Classe ControllerVerifierIdentifiant.
 * Gère la vérification des identifiants d'un fournisseur.
 */
public class ControllerVerifierIdentifiant {

    /**
     * Vérifie si les identifiants fournis sont corrects.
     * Cette méthode compare l'identifiant et le mot de passe fournis avec des
     * valeurs prédéfinies.
     *
     * @param fournisseur L'instance du fournisseur à authentifier (non utilisé dans
     *                    cette implémentation).
     * @param id          L'identifiant saisi par l'utilisateur.
     * @param mdp         Le mot de passe saisi par l'utilisateur.
     * @return true si les identifiants sont corrects, sinon false.
     */
    public boolean verifierIdentifiants(Fournisseur fournisseur, String id, String mdp) {
        return id.equals("fournisseur") && mdp.equals("password");
    }
}
```

```java
package main.java.metier;

/**
 * Énumération Activite.
 * Représente les différentes activités disponibles pour les ventes.
 */
public enum Activite {
    /**
     * Activité de surf.
     */
    SURFING,

    /**
     * Activité de snowboard.
     */
    SNOWBOARDING,

    /**
     * Activité de skateboard.
```

```
     */
    SKATEBOARDING,

    /**
     * Activité de kitesurf.
     */
    KITESURFING,

    /**
     * Activité de planche à voile (windsurf).
     */
    WINDSURFING,

    /**
     * Activité de wakeboard.
     */
    WAKEBOARDING
}
```

```
package main.java.metier;

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

/**
 * Classe Fournisseur.
 * Représente un fournisseur qui gère des ventes.
 * Cette classe contient une liste de ventes associées au fournisseur.
 */
public class Fournisseur {

    /**
     * Liste des ventes associées au fournisseur.
     */
    private List<Vente> ventes;

    /**
     * Constructeur par défaut.
     * Initialise une nouvelle instance de fournisseur avec une liste de ventes
     * vide.
     */
    public Fournisseur() {
        this.ventes = new ArrayList<>();
    }

    /**
     * Ajoute une vente à la liste des ventes du fournisseur.
     *
     * @param vente La vente à ajouter.
     */
    public void ajouterVente(Vente vente) {
        ventes.add(vente);
    }

    /**
     * Récupère la liste des ventes associées au fournisseur.
     *
     * @return Une liste de {@link Vente}.
     */
    public List<Vente> getVentes() {
        return ventes;
    }
}
```

```
package main.java.metier;
```

```java
import java.net.URI;

/**
 * Classe Produit.
 * Représente un produit vendu par un fournisseur.
 */
public class Produit {

    /**
     * Le nom du produit.
     */
    private String nom;

    /**
     * Le prix de vente du produit.
     */
    private double prixVente;

    /**
     * L'URI de la photo associée au produit.
     */
    private URI photo;

    /**
     * Constructeur de la classe Produit.
     *
     * @param nom       Le nom du produit.
     * @param prixVente Le prix de vente du produit.
     * @param photo     L'URI de la photo associée au produit.
     */
    public Produit(String nom, double prixVente, URI photo) {
        this.nom = nom;
        this.prixVente = prixVente;
        this.photo = photo;
    }

    /**
     * Récupère le nom du produit.
     *
     * @return Une chaîne représentant le nom du produit.
     */
    public String getNom() {
        return nom;
    }

    /**
     * Récupère le prix de vente du produit.
     *
     * @return Le prix de vente du produit.
     */
    public double getPrixVente() {
        return prixVente;
    }
}


package main.java.metier;

/**
 * Classe ProduitVente.
 * Représente une association entre un produit et sa quantité dans une vente.
 */
public class ProduitVente {

    /**
     * Le produit associé à cette ligne de vente.
     */
    private Produit produit;
```

```
    /**
     * La quantité du produit dans la vente.
     */
    private int quantite;

    /**
     * Constructeur de la classe ProduitVente.
     *
     * @param produit  Le produit associé.
     * @param quantite La quantité de ce produit.
     */
    public ProduitVente(Produit produit, int quantite) {
        this.produit = produit;
        this.quantite = quantite;
    }

    /**
     * Calcule le total pour cette ligne de vente.
     *
     * @return Le total (prix du produit * quantité).
     */
    public double calculerTotal() {
        return produit.getPrixVente() * quantite;
    }

    /**
     * Récupère le produit associé à cette ligne de vente.
     *
     * @return Une instance de {@link Produit}.
     */
    public Produit getProduit() {
        return produit;
    }

    /**
     * Génère une représentation textuelle de cette ligne de vente.
     *
     * @return Une chaîne décrivant le produit, la quantité et le total.
     */
    @Override
    public String toString() {
        return produit.getNom() + " - Quantité: " + quantite + " - Total: " + calculerTotal();
    }
}


package main.java.metier;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Classe Vente.
 * Représente une vente effectuée par un fournisseur, contenant plusieurs
 * produits.
 */
public class Vente {

    /**
     * Le nom de la vente.
     */
    private String nom;

    /**
     * La date de début de la vente.
     */
```

```
    private Date dateDebut;

    /**
     * La date de fin de la vente.
     */
    private Date dateFin;

    /**
     * La liste des produits inclus dans cette vente.
     */
    private List<ProduitVente> produits = new ArrayList<>();

    /**
     * Constructeur de la classe Vente.
     *
     * @param nom       Le nom de la vente.
     * @param dateDebut La date de début de la vente.
     * @param dateFin   La date de fin de la vente.
     */
    public Vente(String nom, Date dateDebut, Date dateFin) {
        this.nom = nom;
        this.dateDebut = dateDebut;
        this.dateFin = dateFin;
    }

    /**
     * Ajoute un produit à cette vente.
     *
     * @param produit  Le produit à ajouter.
     * @param quantite La quantité de ce produit.
     */
    public void ajouterProduit(Produit produit, int quantite) {
        produits.add(new ProduitVente(produit, quantite));
    }

    /**
     * Récupère la liste des produits inclus dans cette vente.
     *
     * @return Une liste de {@link ProduitVente}.
     */
    public List<ProduitVente> getProduits() {
        return produits;
    }

    /**
     * Redéfinition de la méthode toString pour afficher les détails de la vente.
     *
     * @return Une chaîne contenant le nom, les dates et les produits de la vente.
     */
    @Override
    public String toString() {
        StringBuilder details = new StringBuilder();
        details.append("Vente: ").append(nom)
                .append("\nDate début: ").append(dateDebut)
                .append("\nDate fin: ").append(dateFin)
                .append("\nProduits: \n");

        for (ProduitVente pv : produits) {
            details.append(" - ").append(pv).append("\n");
        }

        return details.toString();
    }
}
```