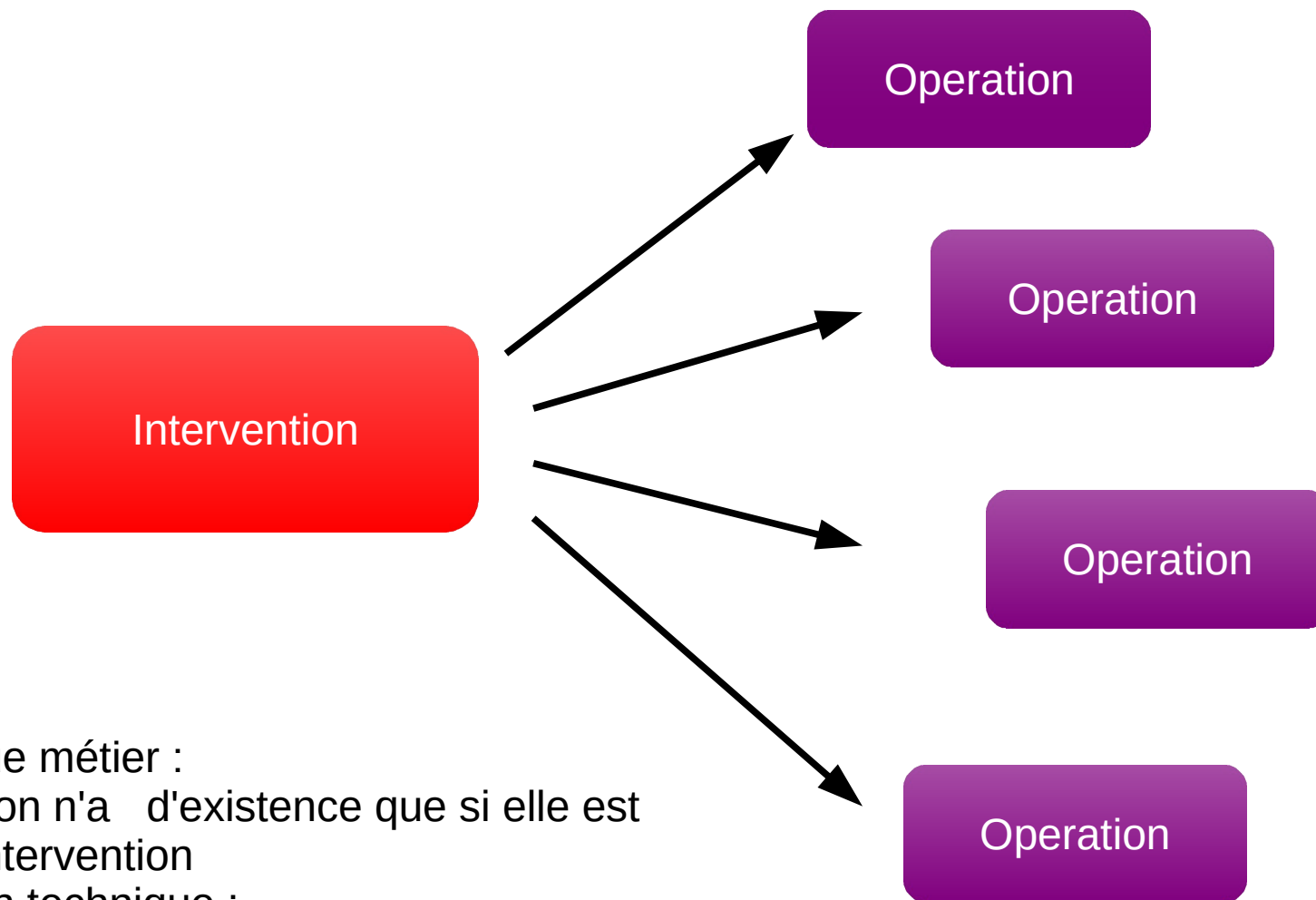




# Jpa relations

## OneToMany Unidirectionnel

# One to Many Unidirectionnel



Notre logique métier :

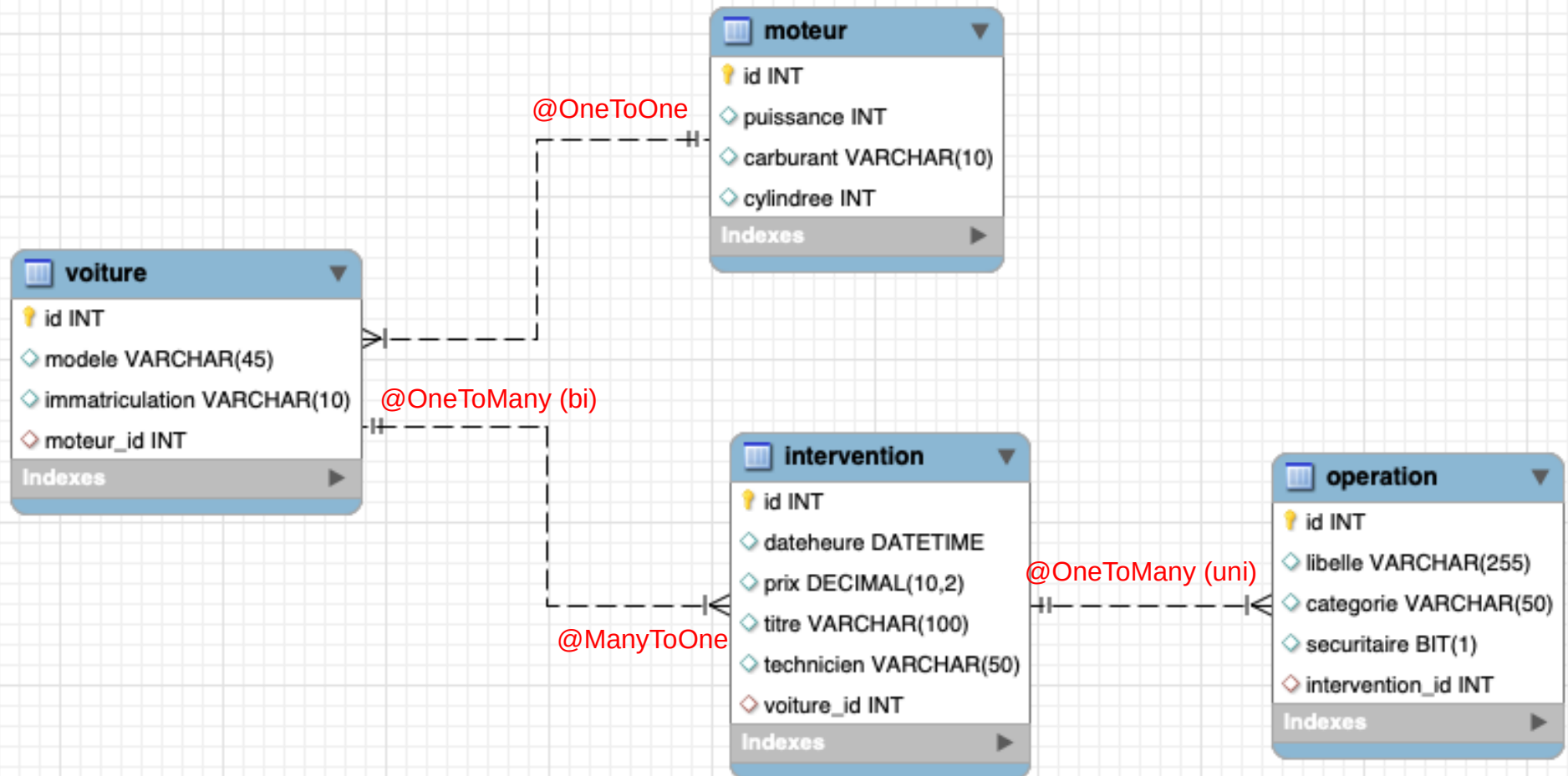
Une opération n'a d'existence que si elle est liée à une intervention

Spécification technique :

On prévoit donc de cascader la suppression d'une intervention sur les opérations liées

# Diagramme schema bdd

## OneToMany – le projet grandit





# Ce qui nous attend dans cette section

- Définir le schéma de bdd
- Créer la classe Operation
- Modifier la classe Intervention
- Manipuler ces éléments avec méthode main

# Create-db.sql

. . .

```
-- Table structure for table `operation`  
DROP TABLE IF EXISTS `operation`;  
CREATE TABLE `operation` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `libelle` varchar (255) DEFAULT NULL,  
  `categorie` varchar (50) DEFAULT NULL,  
  `securitaire` BIT(1) DEFAULT NULL,  
  `intervention_id` INT DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `FK_to_intervention_idx` (`intervention_id`),  
  CONSTRAINT `FK_to_intervention_id`  
  FOREIGN KEY (`intervention_id`) REFERENCES `intervention` (`id`)  
  ON DELETE NO ACTION ON UPDATE NO ACTION  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
```

# La classe Operation.java

```
@Entity
public class Operation {
    @Id
    @GeneratedValue
    (strategy=GenerationType.IDENTITY)
    private int id;

    @Column
    private String libelle;

    @Column
    private char categorie;

    @Column
    private boolean securitaire;

    // Getters & Setters

    //Constructeurs
```

# Modifier Intervention.java

```
@Entity
@Table(name="intervention")
public class Intervention {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column
    private int id ;

    @Column
    private LocalDateTime dateheure;

    @Column
    private String titre;

    @Column
    private Double prix;

    @Column
    private String technicien;

    @ManyToOne(cascade= {CascadeType.PERSIST, CascadeType.MERGE,
                        CascadeType.DETACH, CascadeType.REFRESH})
    @JoinColumn(name="voiture_id")
    private Voiture voiture;

    @OneToMany(fetch=FetchType.LAZY, cascade=CascadeType.ALL)
    @JoinColumn(name="intervention_id")
    private List<Operation> operations ;

    public void add(Operation o) {
        if(operations ==null) operations=new ArrayList<>();
        operations.add(o);
    }

    // getters & setters
    . . .
}
```

Cette Join colonne se réfère  
à la table des Operations

# @JoinColumn en détail

- Dans notre scenario @JoinColumn indique à hibernate ...
- dans la table de cette propriété (ici Operation) il y a (on la spécifie en fait) une variable de jointure (une FK)
- Elle s'appelle "intervention\_id" (par défaut hibernate va lui donner un autre nom, donc ici on choisi le nom de la FK)
- Utilise cette FK pour trouver les associations avec moi-même (this) l'objet intervention



# Commençons par peupler une nouvelle bdd

```
// récupérer une session & ouvrir une transaction
session = factory.getCurrentSession();
session.beginTransaction();

// créer des objets
Voiture v= new Voiture("clio", "444-ddd-44");
Moteur m = new Moteur (115, "diesel",1400);

Intervention i1= new Intervention("Petite Révision", 80.5,"A. Didonk",LocalDateTime.of(2020,Month.OCTOBER,10,10,00,00) );
Intervention i2= new Intervention("Changement des pneus avant", 20. , "M. Imome",LocalDateTime.of(2019,Month.DECEMBER,5,14,30,00) );
Intervention i3= new Intervention("Vidange", 80.5,"S. Ouraille",LocalDateTime.of(2017,Month.JUNE,21,8,45,00) );

v.setMoteur(m);

v.add(i1);
v.add(i2);
v.add(i3);

// sauvegarder ces interventions
session.save(i1);
session.save(i2);
session.save(i3);

session.save(v);

//commit transaction
session.getTransaction().commit();

System.out.println("Terminé !");
```

Le nom du nouveau schéma  
dans la documentation du  
cours sera :

bdd\_07\_one\_to\_many\_uni

Pensez à réexécuter le  
script sql de creation du  
schéma et des tables !

# Methode main() pour manipuler les opérations

```
// récupérer une session & ouvrir une transaction
session = factory.getCurrentSession();
session.beginTransaction();

//créer un objet Intervention

    Intervention i= new Intervention("Revision des 30 000km ", 350.45,
        "O. Kenobi",LocalDateTime.of(2020, Month.AUGUST,18,11,30,00));

    Operation o1= new Operation("vidange huile", 'B',false );
    Operation o2= new Operation("filtre a air", 'A',false );
    Operation o3= new Operation("plaquette avant", 'C',true );

// lier les objets
    i.add(o1);
    i.add(o2);
    i.add(o3);

//sauvegarder l'intervention et cascader les opérations
    System.out.println("Save de l'intervention");
    System.out.println(i);
    System.out.println(i.getOperations());

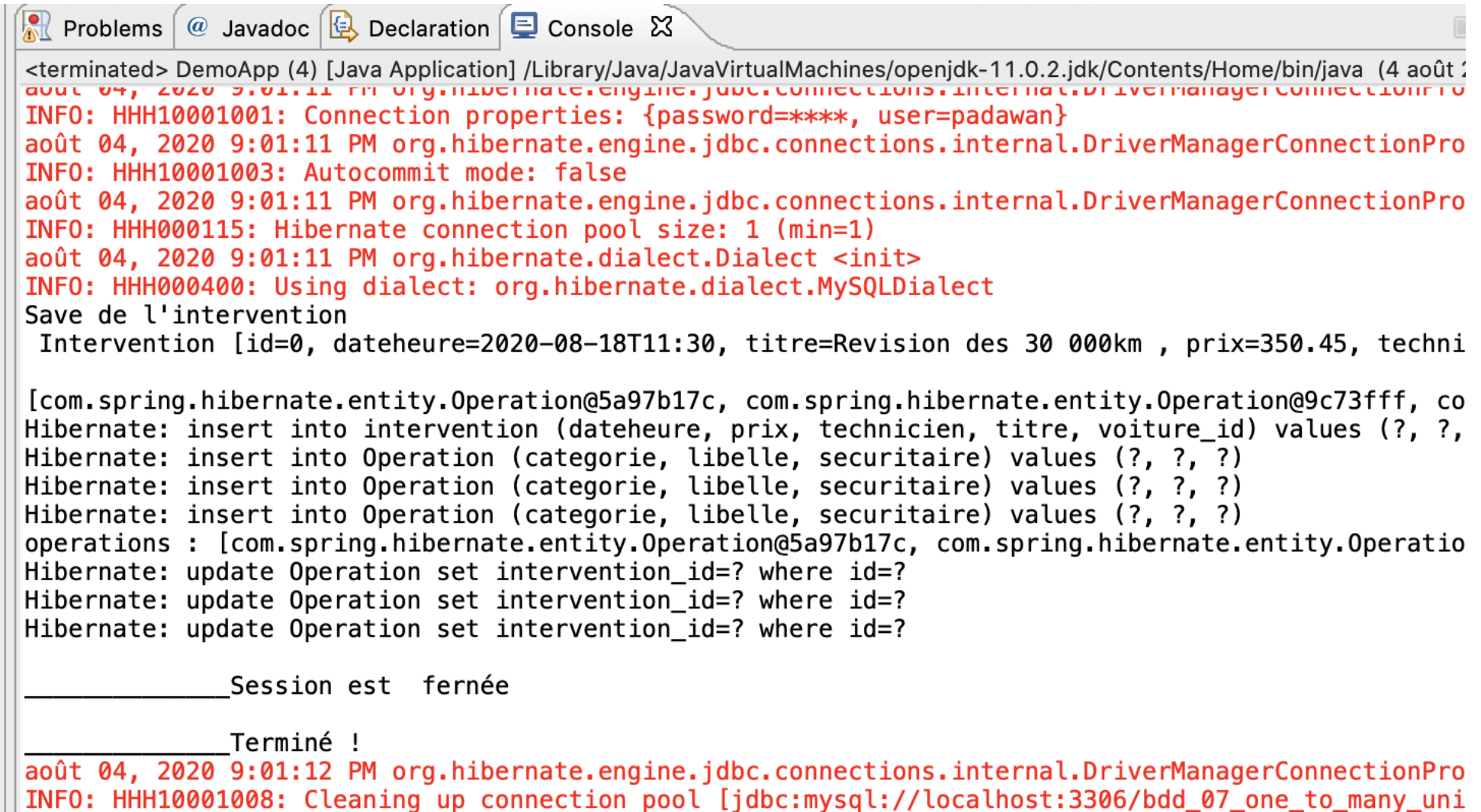
    session.save(i);

//affiche les operations associées
    System.out.println("operations : "+i.getOperations());

//commit transaction
    session.getTransaction().commit();
```

Execute

# Affichage en console



The screenshot shows an IDE window with a tab labeled 'Console'. The console output is as follows:




```
<terminated> DemoApp (4) [Java Application] /Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home/bin/java (4 août 2020 9:01:11 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionPro
INFO: HHH10001001: Connection properties: {password=****, user=padawan}
août 04, 2020 9:01:11 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionPro
INFO: HHH10001003: Autocommit mode: false
août 04, 2020 9:01:11 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionPro
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 04, 2020 9:01:11 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Save de l'intervention
Intervention [id=0, dateheure=2020-08-18T11:30, titre=Revision des 30 000km , prix=350.45, techni

[com.spring.hibernate.entity.Operation@5a97b17c, com.spring.hibernate.entity.Operation@9c73fff, co
Hibernate: insert into intervention (dateheure, prix, technicien, titre, voiture_id) values (?, ?,
Hibernate: insert into Operation (categorie, libelle, securitaire) values (?, ?, ?)
Hibernate: insert into Operation (categorie, libelle, securitaire) values (?, ?, ?)
Hibernate: insert into Operation (categorie, libelle, securitaire) values (?, ?, ?)
operations : [com.spring.hibernate.entity.Operation@5a97b17c, com.spring.hibernate.entity.Operation
Hibernate: update Operation set intervention_id=? where id=?
Hibernate: update Operation set intervention_id=? where id=?
Hibernate: update Operation set intervention_id=? where id=?


_____Session est fermée

_____Terminé !
août 04, 2020 9:01:12 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionPro
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_07_one_to_many_uni
```



**Result Grid**   Filter Rows:  Edit: 

	id	libelle	categorie	securitaire	intervention_id
▶	1	vidange huile	B	0	4
	2	filtre a air	A	0	4
	3	plaquette avant	C	1	4
	NULL	NULL	NULL	NULL	NULL

**Result Grid**   Filter Rows:  Edit:    E

	id	dateheure	prix	titre	technicien	voiture_ic
▶	1	2020-10-10 08:00:00	80.50	Petite Révision	A. Didonk	1
	2	2019-12-05 13:30:00	20.00	Changement des pneus avant	M. Imome	1
	3	2017-06-21 06:45:00	80.50	Vidange	S. Ouraille	1
	4	2020-08-18 09:30:00	350.45	Revision des 30 000km	O. Kenobi	NULL
	NULL	NULL	NULL	NULL	NULL	NULL

# Ce qu'on veut faire c'est lire des interventions et leurs opérations stockées en bdd

Adaptez la valeur de l'id selon votre cas

```
try {  
  
    // récupérer une session & ouvrir une transaction  
    session = factory.getCurrentSession();  
    session.beginTransaction();  
  
    //récupérer un objet Intervention  
    int id = 4;  
    Intervention i=session.get(Intervention.class, id);  
  
    //affiche l'intervention  
    System.out.println("l'intervention : "+i);  
  
    //affiche les opérations associées  
    System.out.println("operations : "+i.getOperations());  
  
    //commit transaction  
    session.getTransaction().commit();  
  
    // fermer la session  
    session.close();  
    System.out.println("\n_____Session est fermée \n");  
  
    System.out.println("_____Terminé !");  
  
    . . .  
}
```

Lazy Load pour les opérations

Execute

# Affichage en console

1ère requête  
transmise à la bdd

```
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 04, 2020 9:17:10 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select interventi0_.id as id1_0_0_, interventi0_.dateheure as dateheur2_0_0_, inter
l'intervention : Intervention [id=4, dateheure=2020-08-18T11:30, titre=Revision des 30 000km

Hibernate: select operations0_.intervention_id as interven5_2_0_, operations0_.id as id1_2_0_,
operations : [com.spring.hibernate.entity.Operation@415ef4d8, com.spring.hibernate.entity.Operation@415ef4d8]

_____Session est fermée

_____Terminé !
août 04, 2020 9:17:11 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnection
```

2ème requête  
transmise à la bdd

Car nous avons configuré le chargement des relations en LAZY  
Hibernate va chercher la data car on la demande explicitement avec le get sur l'objet principal

# Supprimer une Intervention et ses opérations

```
try {  
  
    // récupérer une session & ouvrir une transaction  
    session = factory.getCurrentSession();  
    session.beginTransaction();  
  
    //récupérer un objet Intervention  
    int id = 4;  
    Intervention i=session.get(Intervention.class, id);  
  
    //supprimer l'intervention  
    System.out.println("Suppression de l'intervention : "+i );  
  
    session.delete(i);  
  
    //commit transaction  
    session.getTransaction().commit();  
  
    // fermer la session  
    session.close();  
    System.out.println("\n_____Session est fermée \n");  
  
    System.out.println("_____Terminé !");  
}
```



Execute

# Resultat

Delete en cascade !

```
Problems @ Javadoc Declaration Console X
<terminated> DeleteInterventionAndOperationsDemo [Java Application] /Library/Java/JavaVirtualMachines/openj
août 04, 2020 9:27:16 PM org.hibernate.engine.jdbc.connections.internal.DriverMana
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 04, 2020 9:27:16 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select interventi0_.id as id1_0_0_, interventi0_.dateheure as dateheur2
Suppression de l'intervention : Intervention [id=4, dateheure=2020-08-18T11:30,
Hibernate: select operations0_.intervention_id as interven5_2_0_, operations0_.id
Hibernate: update Operation set intervention_id=null where intervention_id=?
Hibernate: delete from Operation where id=?
Hibernate: delete from Operation where id=?
Hibernate: delete from Operation where id=?
Hibernate: delete from intervention where id=?

_____Session est fermée

_____Terminé !
août 04, 2020 9:27:17 PM org.hibernate.engine.jdbc.connections.internal.DriverMana
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_07
```

Result Grid						
Filter Rows: Search						
id	dateheure	prix	titre	technicien	voiture_id	
1	2020-10-10 08:00:00	80.50	Petite Révision	A. Didonk	1	
2	2019-12-05 13:30:00	20.00	Changement des pneus avant	M. Imome	1	
3	2017-06-21 06:45:00	80.50	Vidange	S. Ouraille	1	
NULL	NULL	NULL	NULL	NULL	NULL	

Result Grid					
Filter Rows: Search					
id	libelle	categorie	securitaire	intervention_id	
NULL	NULL	NULL	NULL	NULL	