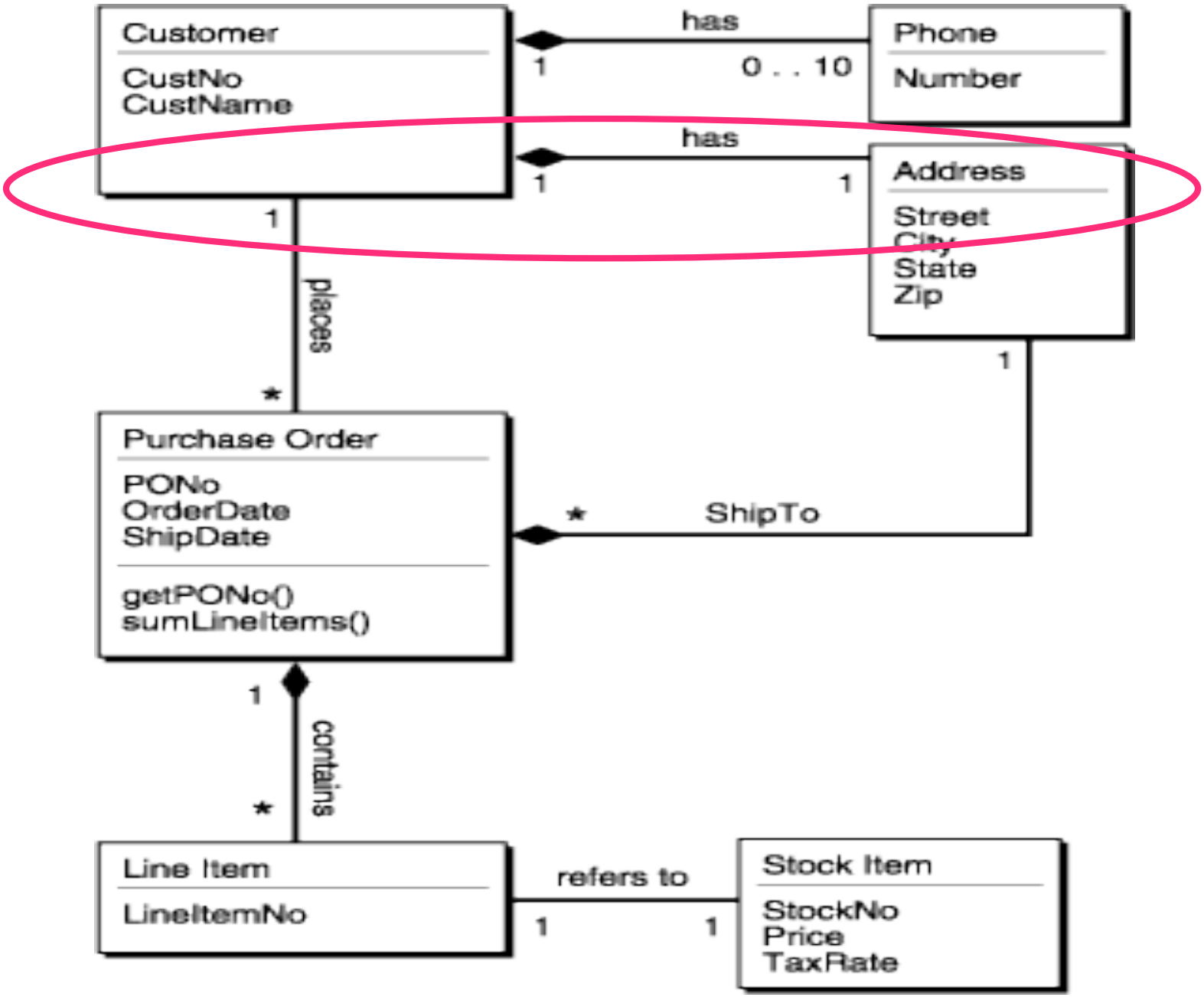
La correspondance Objet-Relationnel

Les associations

Hala Skaf-Molli
Hala.Skaf@univ-nantes.fr
http://pagesperso.lina.univ-nantes.fr/~skaf-h

Association

- Une association peut être uni ou bidirectionnelle
- Elle peut être de type 1:1, 1:N, N:1 ou M:N
- Les associations doivent être indiquées par une annotation sur la propriété correspondante, pour que Hibernate puisse les gérer correctement


Customer
CustNo
CustName
has
1
0 . . 10
Phone
Number
has
1
1
Address
Street
City
State
Zip
1
places
*
1
Purchase Order
PONo
OrderDate
ShipDate
getPONo()
sumLineItems()
*
ShipTo
1
contains
*
Line Item
LineItemNo
refers to
1
1
Stock Item
StockNo
Price
TaxRate

One-To-One

- Trois façons:
  - Embedded (component)
  - Conventionnelle (foreign key)
  - Mêmes clés primaires

  - Et par une table association (voir 1-N ou N-M)
    …

Classe Embeddable

- Les entités persistantes ne sont pas les seules classes persistantes
- Il existe aussi des classes « insérées » ou « incorporées » (embedded) dont les données n'ont pas d'identité dans la BD

```java
import javax.persistence.*;
@Embeddable
public class Address {
    public String street;
    public String city;
    public String state;
    public int zip;


    public Address () {}


    public String getStreet() {
        return street;
    }
    public void setStreet(String street) {
        this.street = street;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getState() {
        return state;
    }
```

```java
import javax.persistence.*;
@Entity
@Table(name="Client")

public class Customer {
    public int id;
    public  String CustName;
    public Address address;

    // getter
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public int getId() {
        return this.id;
    }
    // setter
    public void setId(int id){
        this.id =id;
    }
    @Embedded
    public Address getAddress() {
        return address;
    }
    public void setAddress(Address address) {
        this.address = address;
    }
    @Override
    public String toString() {
```


Table : client(id,nom,Street, City,State,Zip);

Conventionnelle OneToOne

- Annotation @ OneToOne
- Association Unidirectionnelle
- Représentée par une clé étrangère ajoutée dans la table qui correspond au côté propriétaire

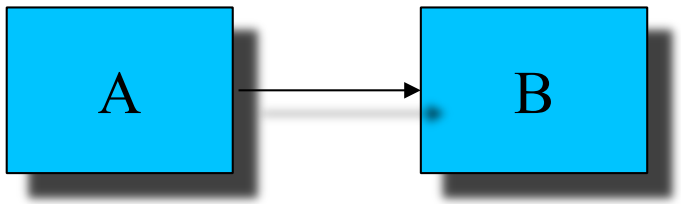
```java
import javax.persistence.*;
@Entity

public class Address {
    private String street;
    private  String city;
    private  String state;
    private int zip;
    int idAddr;


    public Address () {}

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)

    public int getIdAddr() {
            return idAddr;
        }

        public void setIdAddr(int idAddr) {
            this.idAddr = idAddr;
        }

    public String getStreet() {
        return street;
    }
    public void setStreet(String street) {
```

```java
import javax.persistence.*;
@Entity
@Table(name="Client")

public class Customer {
    public int id;
    public  String CustName;
    public Address address;

    // getter
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public int getId() {
        return this.id;
    }
    // setter
    public void setId(int id){
        this.id =id;
    }
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name="address_fk")

    public Address getAddress() {
        return address;
    }
    public void setAddress(Address address) {
        this.address = address;
    }
```

```
Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (6 févr. 08 18:22:05)
Hibernate: insert into Address (idAddr, city, state, street, zip) values (null, ?, ?, ?,
Hibernate: call identity()
Hibernate: insert into Client (id, address_fk, nom) values (null, ?, ?)
Hibernate: call identity()
Id address:1streetvillers
Hibernate: select customer0_.id as id0_1_, customer0_.address_fk as address3_0_1_, custom
Id:1 name: Hala1
Id address:1streetvillers
```

Cascading: persistance par transitivité

- Indique comment gérer la persistance avec des relations
  - Faut-il rendre persistant / supprimer / rafraichir / fusionner un objet en relation avec un autre
- Save-Update »
- REMOVE
- REFRESH
- MERGE



```java
import javax.persistence.*;
@Entity

public class Address {
    private String street;
    private  String city;
    private  String state;
    private int zip;
    int idAddr;


    public Address () {}

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)

    public int getIdAddr() {
            return idAddr;
        }

        public void setIdAddr(int idAddr) {
            this.idAddr = idAddr;
        }

    public String getStreet() {
        return street;
    }
    public void setStreet(String street) {
```

```java
import javax.persistence.*;
@Entity
@Table(name="Client")

public class Customer {
    public int id;
    public  String CustName;
    public Address address;

    // getter
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public int getId() {
        return this.id;
    }
    // setter
    public void setId(int id){
        this.id =id;
    }
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name="address_fk")

    public Address getAddress() {
        return address;
    }
    public void setAddress(Address address) {
        this.address = address;
```

```java
Session s = sf.openSession();
s.beginTransaction();
Customer c1 = new Customer();
// s1.setId(1);
c1.setCustName("Hala1");
Address a = new Address();
a.setStreet("villers");
a.setCity("Nancy");
a.setState("France");
a.setZip(54000);
c1.setAddress(a);
s.save(c1);
s.getTransaction().commit();
s.close();
```

Association OneToOne sur les clés

- 2 classes peuvent être reliées par leur identificateurs : 2 entités sont associées ssi elles ont les mêmes clés

- L'annotation **@PrimaryKeyJoinColumn**
- Attention, c'est au développeur de s'assurer que les entités associées ont bien les mêmes clés.

```java
import javax.persistence.*;
// @Embeddable
@Entity

public class Address {
    private String street;
    private  String city;
    private  String state;
    private int zip;
    int id;


    public Address () {}

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public int getId() {
            return id;
        }

        public void setId(int id) {
            this.id = id;
        }
```

```java
import javax.persistence.*;
@Entity
@Table(name="Client")

public class Customer {
    public int id;
    public  String CustName;
    public Address address;

    // getter
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public int getId() {
        return this.id;
    }
    // setter
    public void setId(int id){
        this.id =id;
    }
    @OneToOne(cascade = CascadeType.ALL)
    @PrimaryKeyJoinColumn

    public Address getAddress() {
        return address;
    }
    public void setAddress(Address address) {
        this.address = address;
```

Représentation des associations 1:N et M:N

- Elles sont représentées par des collections ou maps qui doivent être déclarées par un des types interface suivants (de java.util) :
  - Collection
  - Set
  - List
  - Map
- Les variantes génériques sont conseillées ; par exemple Collection<Customer>

Types à utiliser

- Le plus souvent **Collection** sera utilisé
- **Set** peut être utile pour éliminer les doublons
- Les types concrets, tels que **HashSet** ou **ArrayList**,

ne peuvent être utilisés que pour des entités « nouvelles » ; dès que l'entité est gérée, les types interfaces doivent être utilisés (ce qui permet au fournisseur de persistance d'utiliser son propre type concret)

- **List** peut être utilisé pour conserver un ordre

mais nécessite quelques précautions (@OrderBy)

Représentation des associations 1:N et N:1

- Représentée par une clé étrangère
  - Dans la table qui correspond au côté propriétaire (obligatoirement le côté « Many »)
- Annotations:
  - @ManyToOne, @OneToMany
- Représentée par une table association
  - @JoinTable

Example


```
public class User {
Private int id;
private String name;
}
```

```
public class BillingDetails {
Private int id;
private User user;
}
```

Association Unidirectionnelle Many-to-One

```java
public class User {
    private int id;
    private  String name;


    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="IdUser")
    public int getId() {
        return this.id;
    }
    public void setId(int id){
        this.id =id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
```

Error Log | Tasks | Problems | Console

```
erminated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (13 févr. 08 22:20:35)
.bernate: insert into User (IdUser, name) values (null, ?)
.bernate: call identity()
.bernate: insert into BillingDetails (BILLING_DETAILS_ID, ownername, user_IdUser) v
```

```java
public abstract class BillingDetails {

    public int id;
 private User user;

 @ManyToOne(cascade =CascadeType.ALL)

    public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

    @Id  @GeneratedValue
    @Column(name="BILLING_DETAILS_ID")
    public int getId() {
```

Fetch

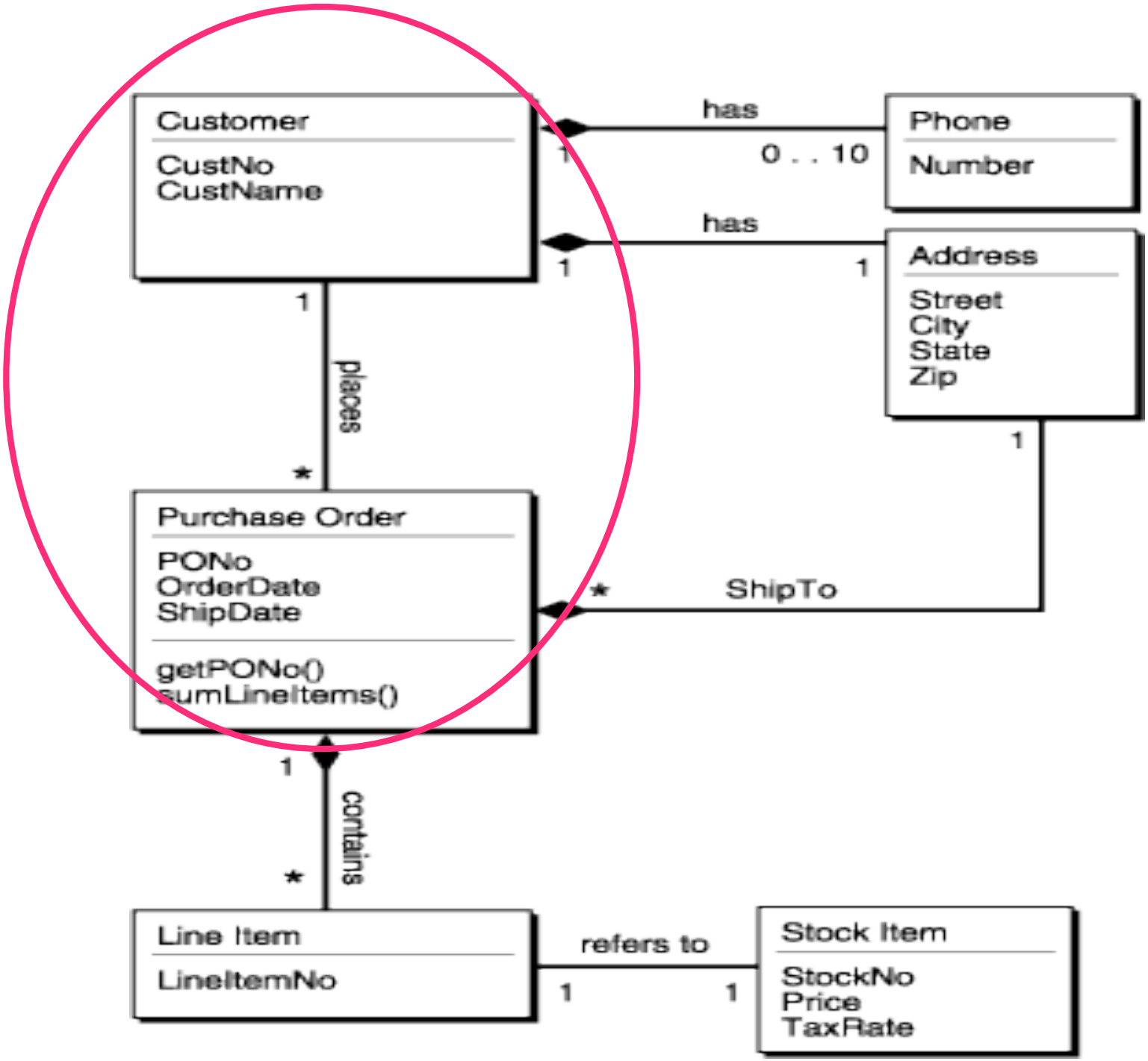
- Permet de spécifier le mode de chargement des relations
  - LAZY : les objets sont chargés à la demande
    - Exemple : j'accède aux *BillingDetails*, les users ne sont pas chargés en mémoire, sauf si on y accède
  - EAGER : la requête charge les objets référencés par la relation.
- Dépend des cas d'utilisation.
- Éviter de mettre EAGER partout

```java
public abstract class BillingDetails {

    public int id;
 private User user;

 @ManyToOne(fetch=FetchType.LAZY,cascade =CascadeType.ALL

    public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

    @Id  @GeneratedValue
    @Column(name="BILLING_DETAILS_ID")
    public int getId() {
        return id;
```

ror Log | Tasks | **Problems** | Console

inated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (13 févr.

```
ernate: select billingdet0_.BILLING_DETAILS_ID as BILLING1
ernate: select user0_.IdUser as IdUser4_0_, user0_.name as
 1 Ow:HalaVISA USER: UserCC
ernate: select user0_.IdUser as IdUser4_0_, user0_.name as
 2 Ow:HalaMASTER USER: UserCC
ernate: select user0_.IdUser as IdUser4_0_, user0_.name as
 3 Ow:HalaCASH USER: UserBA
ernate: select creditcard0_.CREDIT_CARD_ID as BILLING1_1_,
 1 Ow: HalaVISA U:UserCC
 2 Ow: HalaMASTER U:UserCC
```


Customer
CustNo
CustName
has
1
0 . . 10
Phone
Number
has
1
1
Address
Street
City
State
Zip
1
places
*
Purchase Order
PONo
OrderDate
ShipDate
getPONo()
sumLineItems()
*
ShipTo
1
1
contains
*
Line Item
LineItemNo
refers to
1
1
Stock Item
StockNo
Price
TaxRate

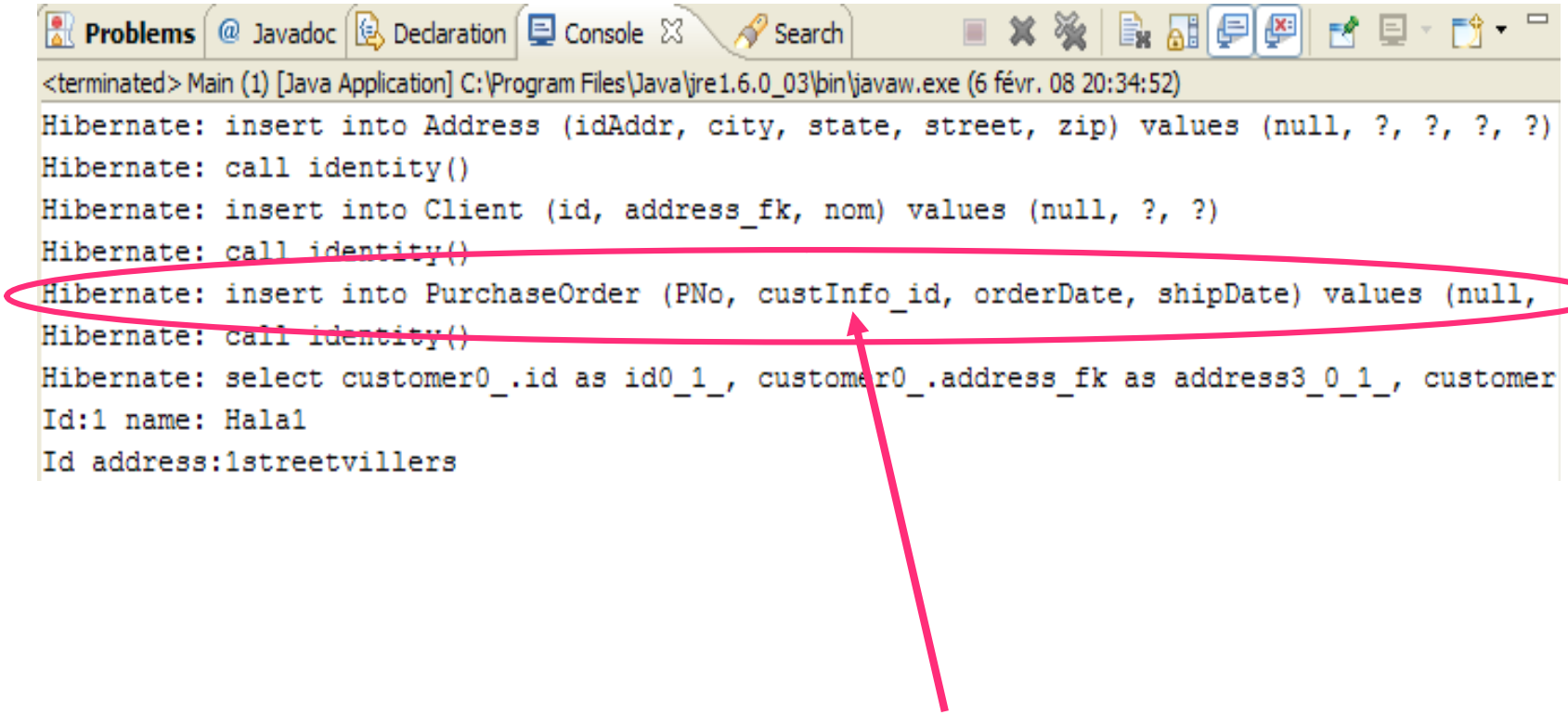
Association bidirectionnelle Many-to-One

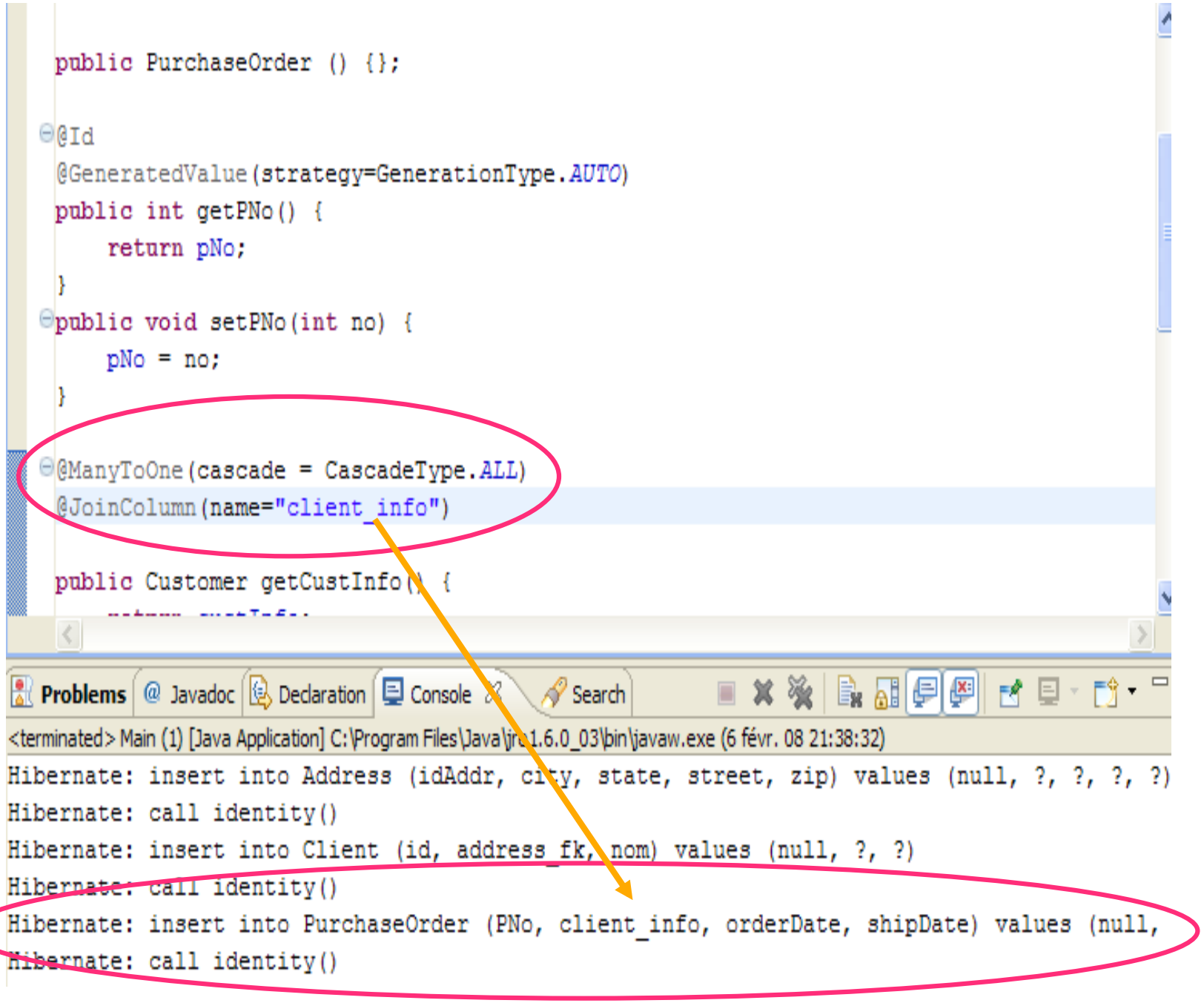
```
private int pNo;
private int orderDate;
private int shipDate;
private Customer custInfo;

public PurchaseOrder () {};

@Id
@GeneratedValue(strategy=GenerationType.AUTO)
public int getPNo() {
    return pNo;
}
public void setPNo(int no) {
    pNo = no;
}

@ManyToOne(cascade = CascadeType.ALL)
public Customer getCustInfo() {
    return custInfo;
}
public void setCustInfo(Customer custInfo) {
    this.custInfo = custInfo;
}
public int getOrderDate() {
    return orderDate;
}
public void setOrderDate(int orderDate) {
```

```java
import java.util.Collection;
import javax.persistence.*;
@Entity
@Table(name="Client")

public class Customer {
    private int id;
    private  String custName;
    private Address address;
    private Collection<PurchaseOrder> orders;

    // getter
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public int getId() {
        return this.id;
    }
    // setter
    public void setId(int id){
        this.id =id;
    }

    @OneToMany(mappedBy="custInfo")
    public Collection<PurchaseOrder> getOrders() {
        return orders;
    }
```

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    SessionFactory sf = new AnnotationConfiguration().configure().buildS

    Session s = sf.openSession();
    s.beginTransaction();

    Address a = new Address();
    a.setStreet("villers");
    a.setCity("Nancy");
    a.setState("France");
    a.setZip(54000);

    Customer c1 = new Customer();
    c1.setCustName("Hala1");
    c1.setAddress(a);
    PurchaseOrder ord = new PurchaseOrder();
    ord.setOrderDate(1);
    ord.setShipDate(2);
    ord.setCustInfo(c1);

    s.save(ord);
```

c1 est rendu persistent par transitivité


Problems
Javadoc
Declaration
Console
Search
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (6 févr. 08 20:34:52)
Hibernate: insert into Address (idAddr, city, state, street, zip) values (null, ?, ?, ?, ?)
Hibernate: call identity()
Hibernate: insert into Client (id, address_fk, nom) values (null, ?, ?)
Hibernate: call identity()
Hibernate: insert into PurchaseOrder (PNo, custInfo_id, orderDate, shipDate) values (null,
Hibernate: call identity()
Hibernate: select customer0_.id as id0_1_, customer0_.address_fk as address3_0_1_, customer
Id:1 name: Hala1
Id address:1streetvillers


public PurchaseOrder () {};
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
public int getPNo() {
return pNo;
}
public void setPNo(int no) {
pNo = no;
}
@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn(name="client_info")
public Customer getCustInfo() {
Problems
Javadoc
Declaration
Console
Search
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (6 févr. 08 21:38:32)
Hibernate: insert into Address (idAddr, city, state, street, zip) values (null, ?, ?, ?, ?)
Hibernate: call identity()
Hibernate: insert into Client (id, address_fk, nom) values (null, ?, ?)
Hibernate: call identity()
Hibernate: insert into PurchaseOrder (PNo, client_info, orderDate, shipDate) values (null,
Hibernate: call identity()

One-To-Many (reverse)

Main.java | Customer.java | hibernate.cfg.xml | Toto.java | PurchaseOrder.java

```
a.setCity("Nancy");
a.setState("France");
a.setZip(54000);
Customer c1 = new Customer();
c1.setCustName("Hala1");
c1.setAddress(a);

Collection<PurchaseOrder> LesOrders =new HashSet<PurchaseOrder>();
PurchaseOrder ord = new PurchaseOrder();
ord.setOrderDate(1);
ord.setShipDate(2);
LesOrders.add(ord);
c1.setOrders(LesOrders);
s.save(c1);

s.getTransaction().commit();
```

ça marche ?!

Problems | Javadoc | Declaration | Console | Search

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (6 févr. 08 22:01:57)
Hibernate: insert into Address (idAddr, city, state, street, zip) values (null, ?, ?, ?, ?)
Hibernate: call identity()
Hibernate: insert into Client (id, address_fk, nom) values (null, ?, ?)
Hibernate: call identity()
Hibernate: insert into PurchaseOrder (PNo, client_info, orderDate, shipDate) values (null,
Hibernate: call identity()
Hibernate: select customer0_.id as id0_1_, customer0_.address_fk as address3_0_1_, customer
Id:1 name: Hala1
```

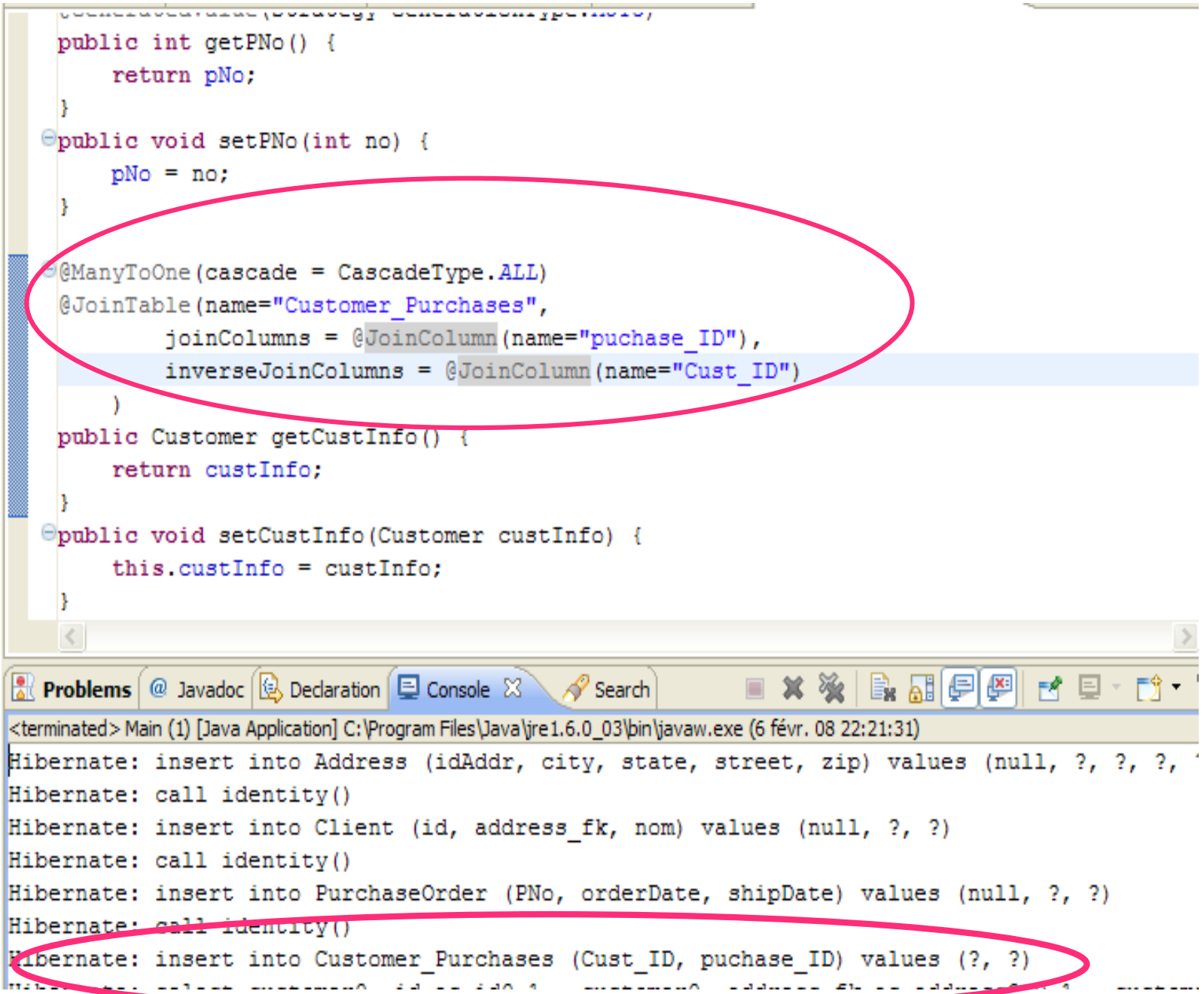
```
// getter
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
public int getId() {
    return this.id;
}
// setter
public void setId(int id){
    this.id =id;
}

@OneToMany(mappedBy="custInfo",cascade =CascadeType.ALL)
public Collection<PurchaseOrder> getOrders() {
    return orders;
}
public void setOrders(Collection<PurchaseOrder> orders) {
    this.orders = orders;
}
```

Représentation des associations 1:N et N:1

- Représentée par une clé étrangère
  - Dans la table qui correspond au côté propriétaire (obligatoirement le côté « Many »)
- Annotations:
  - @ManyToOne, @OneToMany
- Représentée par une table association
  - @JoinTable

@JoinTable

- Donne des informations sur la table association
- Attribut *name* donne le nom de la table
- Attribut *joinColumns* donne les noms des colonnes de la table qui référencent les clés primaires du côté propriétaire de l'association (Many)
- Attribut inverseJoinColumns donne les noms des colonnes de la table qui référencent les clés primaires du côté qui n'est pas propriétaire de l'association


public int getPNo() {
return pNo;
}
public void setPNo(int no) {
pNo = no;
}
@ManyToOne(cascade = CascadeType.ALL)
@JoinTable(name="Customer_Purchases",
joinColumns = @JoinColumn(name="puchase_ID"),
inverseJoinColumns = @JoinColumn(name="Cust_ID")
)
public Customer getCustInfo() {
return custInfo;
}
public void setCustInfo(Customer custInfo) {
this.custInfo = custInfo;
}
Problems
Javadoc
Declaration
Console
Search
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (6 févr. 08 22:21:31)
Hibernate: insert into Address (idAddr, city, state, street, zip) values (null, ?, ?, ?,
Hibernate: call identity()
Hibernate: insert into Client (id, address_fk, nom) values (null, ?, ?)
Hibernate: call identity()
Hibernate: insert into PurchaseOrder (PNo, orderDate, shipDate) values (null, ?, ?)
Hibernate: call identity()
Hibernate: insert into Customer_Purchases (Cust_ID, puchase_ID) values (?, ?)

One-To-Many (reverse)

Main.java | Customer.java | hibernate.cfg.xml | Toto.java | PurchaseOrder.java

```
a.setCity("Nancy");
a.setState("France");
a.setZip(54000);
Customer c1 = new Customer();
c1.setCustName("Hala1");
c1.setAddress(a);

Collection<PurchaseOrder> LesOrders =new HashSet<PurchaseOrder>();
PurchaseOrder ord = new PurchaseOrder();
ord.setOrderDate(1);
ord.setShipDate(2);
LesOrders.add(ord);
c1.setOrders(LesOrders);
s.save(c1);

s.getTransaction().commit();
```

Problems | Javadoc | Declaration | Console | Search

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (6 févr. 08 22:01:57)
Hibernate: insert into Address (idAddr, city, state, street, zip) values (null, ?, ?, ?, ?)
Hibernate: call identity()
Hibernate: insert into Client (id, address_fk, nom) values (null, ?, ?)
Hibernate: call identity()
Hibernate: insert into PurchaseOrder (PNo, client_info, orderDate, shipDate) values (null,
Hibernate: call identity()
Hibernate: select customer0_.id as id0_1_, customer0_.address_fk as address3_0_1_, customer
Id:1 name: Hala1
```

Association M:N (1)

- Annotation @ManyToMany
- Représentée par une table association
- Les valeurs par défaut :
  - le nom de la table association est la concaténation des 2 tables, séparées par « _ »
  - les noms des colonnes clés étrangères sont les concaténations de la table référencée « _ » et de la colonne « Id » de la table référencée

```java
public class PurchaseOrder {
private int pNo;
private int orderDate;
private int shipDate;
private Collection<Customer> Customers;


@ManyToMany
public Collection<Customer> getCustomers() {
    return Customers;
}

public void setCustomers(Collection<Customer> custome
    Customers = customers;
}
```

```java
public class Customer {
    private int id;
    private  String custName;
    private Address address;
    private Collection<PurchaseOrder> orders;

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public int getId() {
        return this.id;
    }
    public void setId(int id){
        this.id =id;
    }

    @ManyToMany(cascade =CascadeType.ALL)

    public Collection<PurchaseOrder> getOrders() {
        return orders;
```

Problems | Javadoc | Declaration | Console | Search

<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (6 févr. 08 22:59:08)

```
log4j:WARN Please initialize the log4j system properly.
Hibernate: insert into Address (idAddr, city, state, street, zip) valu
Hibernate: call identity()
Hibernate: insert into Client (id, address_fk, nom) values (null, ?, ?
Hibernate: call identity()
Hibernate: insert into PurchaseOrder (PNo, orderDate, shipDate) values
Hibernate: call identity()
Hibernate: insert into Client_PurchaseOrder (Client_id, orders_PNo) va
Hibernate: select customer0_.id as id0_1_, customer0_.address_fk as ad
```

Association M:N (2)

- Si les valeurs par défaut ne conviennent pas,
- Le côté propriétaire doit comporter une annotation **@JoinTable**
- L'autre côté doit toujours comporter l'attribut **mappedBy**

```java
@Entity
public class PurchaseOrder {
private int pNo;
private int orderDate;
private int shipDate;
private Collection<Customer> Customers;

@ManyToMany(mappedBy="orders")
public Collection<Customer> getCustomers() {
    return Customers;
}

public void setCustomers(Collection<Customer>
    Customers = customers;
}

public PurchaseOrder () {};
```

```java
public class Customer {
    private int id;
    private  String custName;
    private Address address;
    private Collection<PurchaseOrder> orders;

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public int getId() {
        return this.id;
    }
    public void setId(int id){
        this.id =id;
    }
    @ManyToMany(cascade =CascadeType.ALL)
     @JoinTable(name="Customer_Purchases",
     joinColumns = @JoinColumn(name="puchase_ID"),
      inverseJoinColumns = @JoinColumn(name="Cust_ID")
    )
    public Collection<PurchaseOrder> getOrders() {
        return orders;
```

oblems | Javadoc | Declaration | Console | Search

nated> Main (1) [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (6 févr. 08 23:09:05)

```
rnate: insert into Address (idAddr, city, state, street, zip) values (nul
rnate: call identity()
rnate: insert into Client (id, address_fk, nom) values (null, ?, ?)
rnate: call identity()
rnate: insert into PurchaseOrder (pNo, orderDate, shipDate) values (null,
rnate: call identity()
rnate: insert into Customer_Purchases (puchase_ID, Cust_ID) values (?, ?)
```