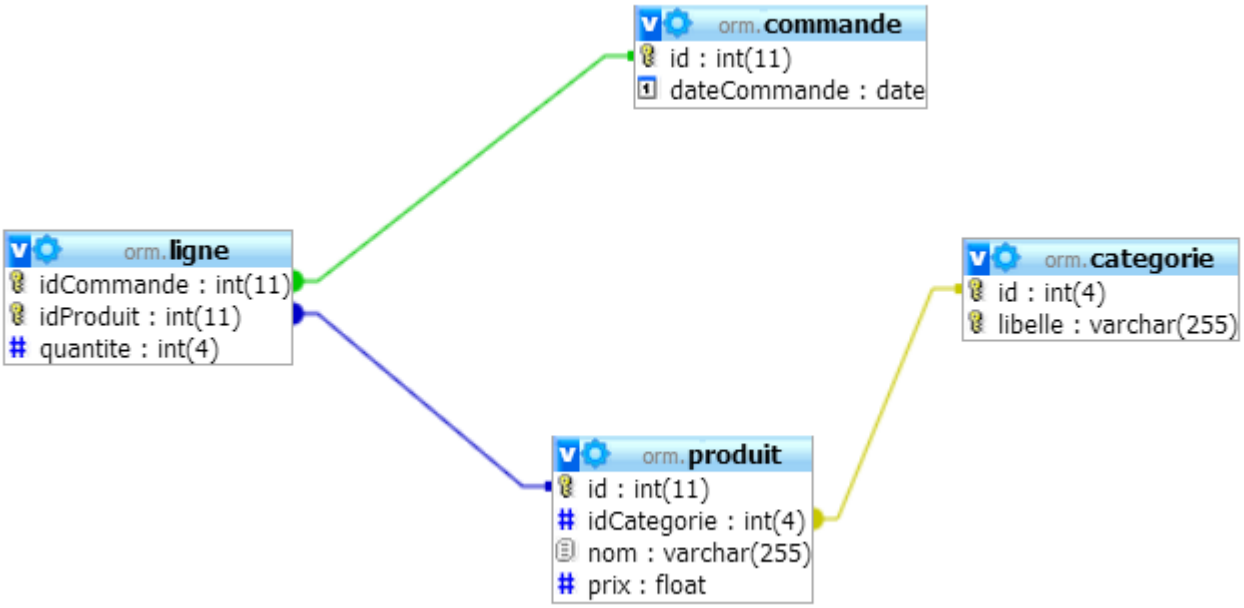
Documentation Kobject

Modèle relationnel



1) Produit/Catégorie
- Un produit appartient à une et une seule catégorie.
- Une catégorie peut contenir zéro ou plusieurs produits.

2) Produit/Ligne/Commande
- Un produit correspond à une ligne dans une commande.
- Une commande peut comporter plusieurs produits.

Première partie

Création des classes métier: **Produits et catégories.**

1)Comment est déclarée la table assurant la persistance d'un objet ?

–> Elle hérite de la classe Kobject.

2)Comment est déclaré le mapping entre un membre de la classe et un champ de la table relationnelle ?

–> Le mapping entre un membre de la classe et un champ de la table relationnelle est déclaré dans le constructeur en tant que membre.

code Java

```
hasMany(KLigne.class);
belongsTo(KCategorie.class);
```

3)Comment est déclarée la clé primaire de la table ?

–> la clé primaire de la table est déclarée avec un keyFields.

code Java

```
keyFields="id";
```

4)Réaliser un tableau montrant la correspondance de type (entier, chaine, etc.) entre les propriétés d'une classe et les champs d'une table.

propriété d'une classe	champs d'une table	type
int	int	entier
string	varchar	chaine

5)Montrez à l'aide d'un schéma (par ex. deux classes liées au dessus de deux tables liées) comment se paramètre le lien bidirectionnel entre deux classes (en spécifiant les éléments à fournir dans le constructeur)

Modèle relationnel :

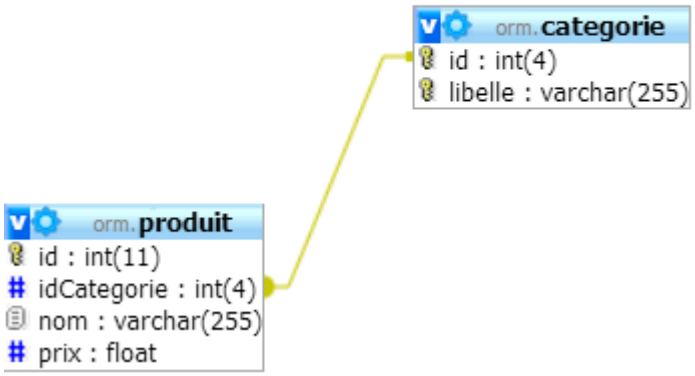
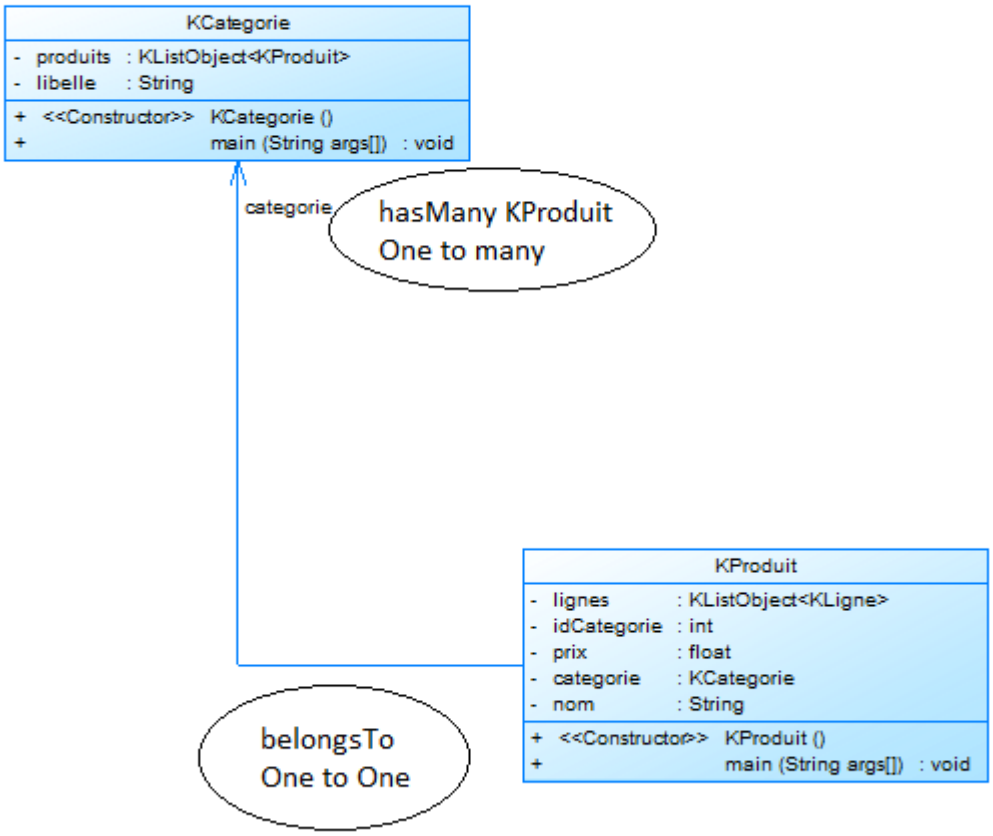


Diagramme de classes :



Deuxième partie

Programme de test:

1) À quoi correspond la méthode kstart() ?
La méthode kstart() permet le lancement d'une application Kobject.

code Java

```
Ko.kstart();
```

2)Comment ont été traduits les liens objet entre le membre categorie et produits entre ces classes dans les tables de la base ?

L'id de la categorie correspond à la clé primaire de la table catégorie.
Dans la table produit l'idCategorie correspond à la clé étrangère en relation à l'id de la table categorie.
Ces deux tables sont donc en relation par le biais de l'id categorie.

3)Quelles requêtes SQL ont été créées par KObject pour réaliser la persistance ?

Pour réliser ce test qui consisté à instancier une catégorie et un produit appartenant à cette catégorie, KObject a utilisé deux requète SQL d'insertion.
Ces requêtes sont visibles dans la console lors de l'execution du test.
SQL : KDataBase.execute → INSERT INTO categorie(libelle) VALUES('Presse')
SQL : KDataBase.execute → INSERT INTO produit(idCategorie,prix,nom) VALUES('19','3.0','Programmez!')

4)Que se passe t-il si l'insertion de la catégorie échoue ?

Si l'insertion de la catégorie échoue, l'insertion du produit ne pourra pas s'exécuter.
En effet le produit que l'on cherche à instancier doit appartenir à la catégorie que l'on cherche à instancier.
Donc si l'insertion de la catégorie échoue il est évident que l'on pourra pas créer un produit sans catégorie puisque ces deux objets sont liés par l'id de la catégorie.

Troisième partie

Chargement d'un objet:

1)Précisez ce que charge exactement KObject lors du chargement d'un Objet

Lors du chargement d'un Objet, KObject va charger tous les objets en relation avec l'élu.
Par exemple en chargeant une catégorie, tous les produits de cette catégorie seront chargés eux aussi.

2)Précisez comment sont chargées les instances liées à un objet chargé pour les liens belongsTo et hasMany

belongsTo:
Pour charger les instances liées à un objet en Many to One, par exemple un produit, KObject va dans un premier temps récupérer le produit à l'aide d'un select. Ensuite KObject va récupérer la catégorie correspondante.
Console:
SQL : KDataBase.sendQuery → SELECT * FROM produit WHERE produit.id='50'
SQL : KDataBase.sendQuery → SELECT * FROM Ligne WHERE idProduit='50'
SQL : KDataBase.sendQuery → SELECT * FROM categorie WHERE id='2'
hasMany:
Pour le cas d'un One To Many KObject va effectuer deux requêtes SQL. Dans un premier temps la catégorie sera chargée ensuite tous les produits appartenant à cette catégorie seront chargés.
Console:
SQL : KDataBase.sendQuery → SELECT * FROM categorie WHERE categorie.id='3'
SQL : KDataBase.sendQuery → SELECT * FROM produit WHERE idCategorie='3'

3)En quoi consiste le chargement paresseux de KObject ?

Le chargement de KObject est qualifié de paresseux parce qu'il ne charge pas tous les objets associé à un objet si on le lui demande pas.
Si on affiche une catégorie KObject n'affichera pas les produits de cette catégorie à moins de le lui demander.

Quatrième partie

Chargement de listes d'objets:

1)Interprétez et expliquez le résultat obtenu avec cette ligne de code :

code java

```
    System.out.println(categories.showWithMask("{libelle}:\n{produits}\n"));
```

Cette ligne va permettre de classer l'afficher de tous les produits par catégorie.
Les catégories seront affichées par ordre alphabétique.
Pour chaque catégorie on aura tous les produits correspondant, classés par ordre alphabétique.

A partir du programme : Souligné

```
Combien de requêtes SQL sont exécutées par KObject ?
Comment l'interprétez vous ?
```