BTS SIO
SLAM4
PHP Objet - Avancé

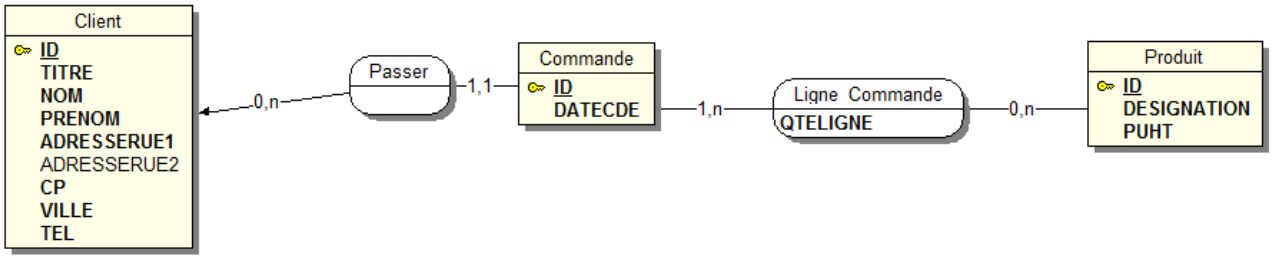
TP Employés

Contexte

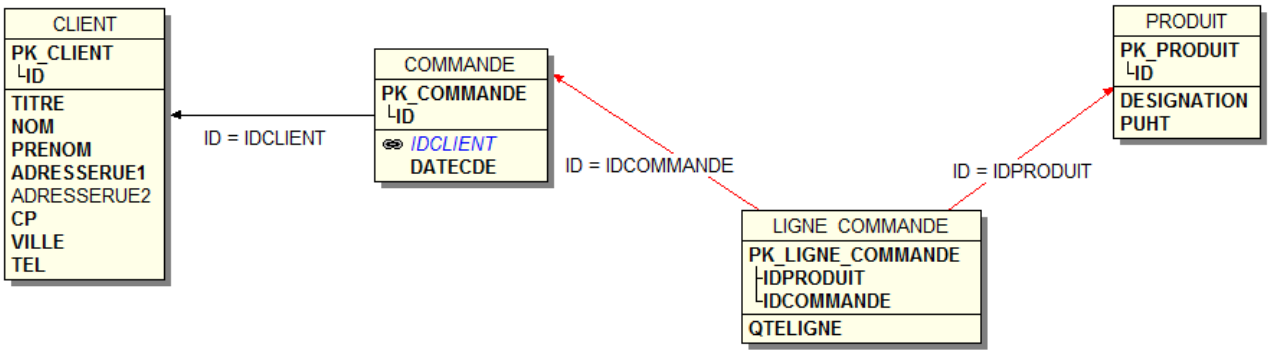
Cours Programmation objet.
Mise en œuvre des classes

Partie 1 : Préparation du travail

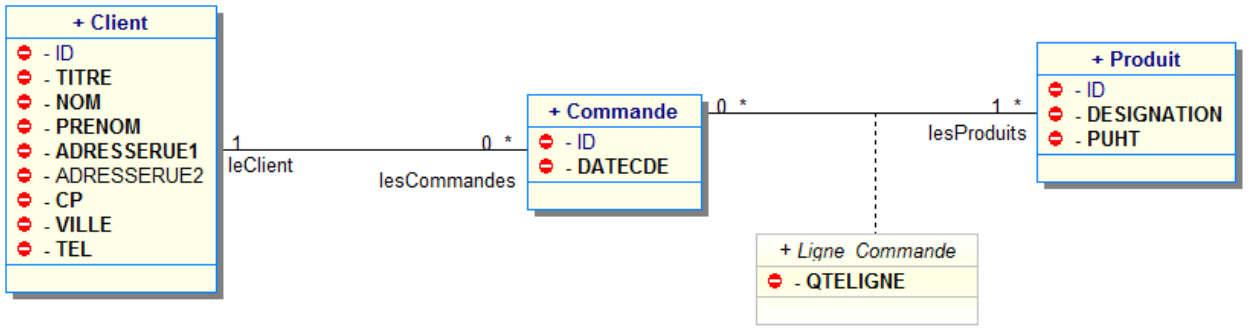
Soit le Modèle Conceptuel des Données suivant :



Le Modèle logique des données correspondant :



Et le diagramme de classe suivant :



Il faudra créer la base de données et charger les tables en exécutant le script Crebase_client_Export.sql

Nous ne nous intéresserons pour l'instant uniquement aux clients et aux commandes. On ne s'intéresse pas aux produits et aux lignes de commande

Voici la description partielle de la classe Client

```
class Client
{
    private $id;
    private $titre;
    private $nom;
    private $prenom;
    private $adresserue1;
    private $adresserue2;
    private $cp;
    private $ville;
    private $tel;
    private $lesCommandes; // tableaux d'objets de la classe Commande
}
```

Et la description partielle de la classe Commande

```
class Commande
{
    private $id;
    private $datecde;
    private $leClient; // objet de la classe Client
}
```

Partie 2 : Travail à faire

Il vous est demandé de créer la classe BDD suivante :

```php
class Bdd {

    private $_dbmysql;

    public function __construct() {
        try {
            $connect_str = "mysql:host=localhost;dbname=port";
            $connect_user = "root";
            $connect_pass = "";
            $options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
            $this->_dbmysql = new PDO($connect_str, $connect_user, $connect_pass, $options);
        } catch (Exception $e) {
            throw new Exception("Erreur à la connexion \n" . $e->getMessage());
        }
    }

    public function requeteSQL($sql) {

    }
}
```

Vous créerez également pour chaque classe métier (Client et Commande), une classe repository à savoir ClientRepository et CommandeRepository.

Ces classe contiennent en attribut au moins l'attribut $classe qui contient le nom de la classe dont il est le repository.

Elles contiennent chacune les méthodes :

- ✓ **findById($id)** : va chercher l'enregistrement de la table ayant pour id l'id passé en paramètre et renvoie un objet de la classe dont elle est le repository. Par exemple, la dans la classe ClientRepository, l'appel à la méthode findById(5) ira chercher le client dont l'id est égal à 5 et renverra l'objet Client correspondant.

- ✓ **findAll()** : sur le même principe que findById, mais renverra un tableau d'objets contenant tous les objets correspondant aux enregistrements trouvés dans la base. Par exemple l'appel à la méthode findAll du repository de la classe Commande renverra un tableau d'objets de la classe Client.
  Attention : dans la classe Commande, il y a un attribut $leClient qu'il faudra valoriser.

- ✓ Insert($objet) : ira insérer dans la base de données la ou les enregistrements issus de l'objet passé en paramètre.

Pour le repository de Client, vous prévoirez les méthodes suivantes :
findByCp($unCp), **findByVille($uneVille)**

Pour le repository de commande, vous prévoirez les méthodes suivantes :
findByDate($uneDate), **findByIntervalleDate**($dateDepart, $dateFin)

Partie 4 : Gestion des exceptions

Vous prendrez en charge la gestion des exceptions. Vos classes techniques se contenteront d'intercepter les exceptions, voire les déclencher, mais elles ne les traiteront jamais. Ce sera le rôle de la partie graphique.

Par exemple si une des méthodes findById ne ramène aucune ligne, elle déclenche une exception qui sera traitée dans la partie applicative du programme.

Partie 3 : Interface graphique

Le but du TP est de vous faire réfléchir à une architecture logicielle et non pas de vous faire développer une application web ou un site vitrine.
Ainsi, vous testerez vos classes à travers une interface graphique la plus dépouillée possible, seuls les affichages des résultats seront à faire.