Itération 4

Début de séance :

-Faire la programmation qui correspond aux représentations (spectacle dans notre projet)

-Faire le code qui concerne le lieu

Classe lieu :

```php
<?php


namespace modele\metier;


class Lieu
{

    /** @var int ...*/
    private $id;
    /**
     * @var string
     */
    private $nom;
    /**
     * @var string
     */
    private $adresse;
    /**
     * @var integer
     */
    private $capacite;

    /** Lieu constructor. ...*/
    public function __construct(int $id, string $nom, string $adresse, int $capacite){...}

    /**
     * @return int
     */
    public function getId(): int
```

Classe LieuDAO :

```php
class LieuDao
{

    /**
     * Instancier un objet de la classe Lieu à partir d'un enregistrement de la table Lieu
     * @param array $enr
     * @return Lieu
     */
    protected static function enregVersMetier(array $enr)
    {
        return new Lieu($enr["ID"], $enr["NOM"], $enr["ADRESSE"], $enr["CAPACITE"]);
    }

    /**
     * Valorise les paramètres d'une requête préparée avec l'état d'un objet Lieu
     * @param Lieu $lieu
     * @param PDOStatement $stmt
     */
    protected static function metierVersEnreg(Lieu $lieu, PDOStatement $stmt)
    {
        // On utilise bindValue plutôt que bindParam pour éviter des variables intermédiaires
        // Note : bindParam requiert une référence de variable en paramètre n°2 ;
        // avec bindParam, la valeur affectée à la requête évoluerait avec celle de la variable sans
        // qu'il soit besoin de refaire un appel explicite à bindParam
        $stmt->bindValue( parameter: ':id', $lieu->getId());
        $stmt->bindValue( parameter: ':nom', $lieu->getNom());
        $stmt->bindValue( parameter: ':adresse', $lieu->getAdresse());
        $stmt->bindValue( parameter: ':capacite', $lieu->getCapacite());
    }

    /**
```

LieuTest:

Test unitaire de la classe métier Lieu

```
F:\Cours\Deuxième Année\PPE\Projet1\files\festivalphp_2019\test\metier\LieuTest.php:15:
object(modele\metier\Lieu)[1]
  private 'id' => int 1
  private 'nom' => string 'Stade de France' (length=15)
  private 'adresse' => string 'Au stade de France' (length=18)
  private 'capacite' => int 10000
```

LieuDAOTest :

```
F:\Cours\Deuxième Année\PPE\Projet1\files\festivalphp_2019\test\dao\LieuDaoTest.php:31:
object(modele\metier\Lieu)[3]
  private 'id' => int 2
  private 'nom' => string 'Salle de champagne' (length=18)
  private 'adresse' => string '76 Boulevard de l'allée' (length=24)
  private 'capacite' => int 6000
```

2- Test getAll

```
F:\Cours\Deuxième Année\PPE\Projet1\files\festivalphp_2019\test\dao\LieuDaoTest.php:40:
array (size=4)
  0 =>
    object(modele\metier\Lieu)[4]
      private 'id' => int 1
      private 'nom' => string 'Champ de la Plaine' (length=18)
      private 'adresse' => string '18 Rue simpa' (length=12)
      private 'capacite' => int 10000
  1 =>
    object(modele\metier\Lieu)[5]
      private 'id' => int 2
      private 'nom' => string 'Salle de champagne' (length=18)
      private 'adresse' => string '76 Boulevard de l'allée' (length=24)
      private 'capacite' => int 6000
  2 =>
    object(modele\metier\Lieu)[6]
      private 'id' => int 3
      private 'nom' => string 'Terrain du paysan' (length=17)
      private 'adresse' => string '2 Impasse de la ferme' (length=21)
      private 'capacite' => int 50000
  3 =>
    object(modele\metier\Lieu)[7]
      private 'id' => int 4
      private 'nom' => string 'Chateau de l'ancêtre' (length=21)
      private 'adresse' => string 'Parc du chateau' (length=15)
      private 'capacite' => int 1000
```

3- insert

ooo réussite de l'insertion ooo

```
F:\Cours\Deuxième Année\PPE\Projet1\files\festivalphp_2019\test\dao\LieuDaoTest.php:54:
object(modele\metier\Lieu)[9]
  private 'id' => int 6
  private 'nom' => string 'Lieu de test' (length=12)
  private 'adresse' => string 'Adresse des tests' (length=17)
  private 'capacite' => int 500000
```

Classe Spectacle:

```php
<?php
    namespace modele\metier;

    class Spectacle{

        /**
        * @var int
        */
        private $idGroup;

        /**
         * @var int
         */
        private $idLieu;

        /**
         * @var date
         */
        private $date;

        /**
         * @var float
         */
        private $heureDeb;

        /**
         * @var float
         */
        private $heureFin;
```

SpectacleTest.class:

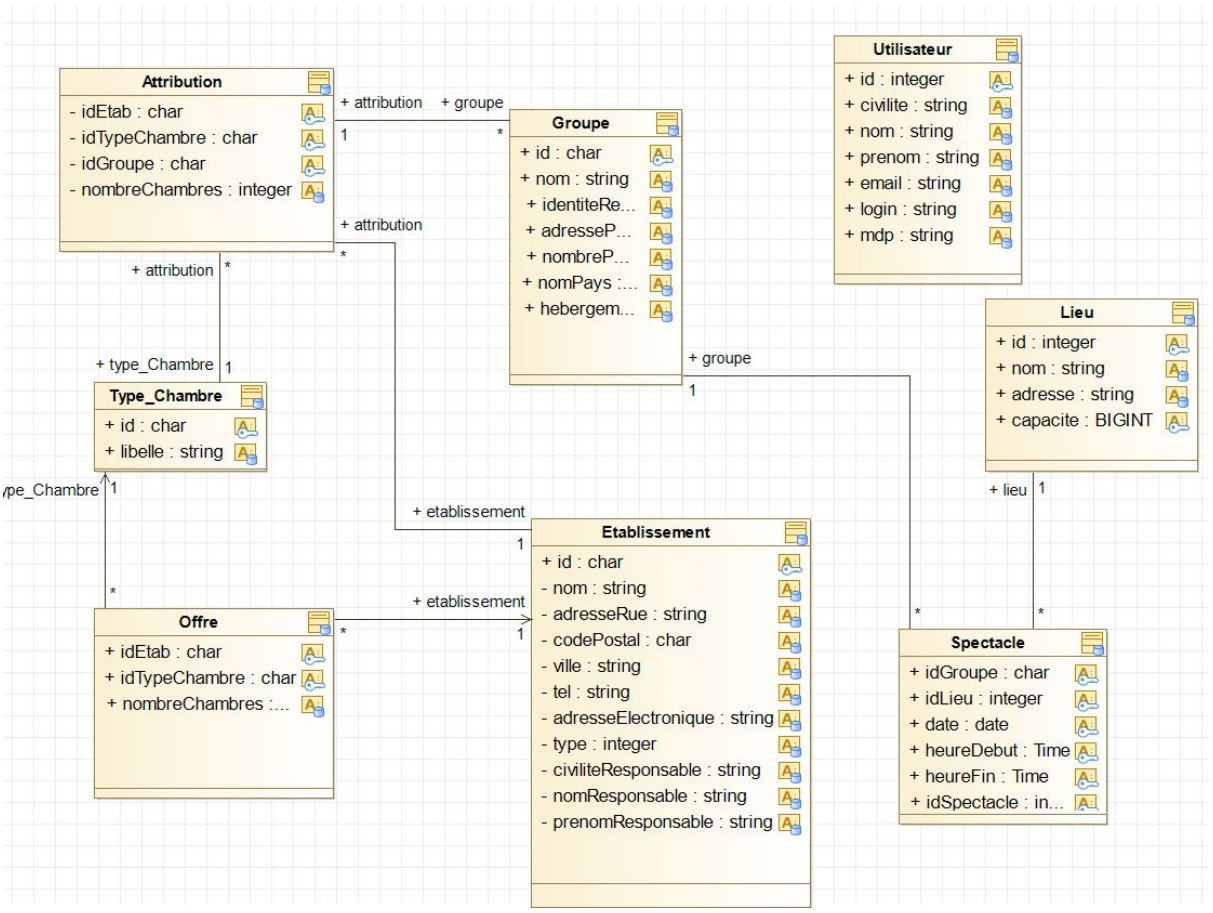
```php
<!Doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Spectacle:</title>
</head>
<body>
    <?php
    use modele\metier\Spectacle;

    require_once __DIR__ . '/../../includes/autoload.inc.php';
    echo "<h2>Test unitaire de la classe métier Spectacle</h2>";
    $spec = new Spectacle("g014",1,"2019-06-11<","11","12");
    var_dump($spec);
    ?>
</body>
</html>
```

Test de la classe SpectacleTest.class:

Test unitaire de la classe métier Spectacle

objet (modele \ metier \ Spectacle) # 1 (5) {["idGroup": "modele \ metier \ Spectacle": privé] => string (4) "g014" ["idLieu": "modele \ metier \ Spectacle" : private] => int (1) ["date": "modele \ metier \ Spectacle": private] => chaîne (10) "2019-06-11" ["heureDeb": "modele \ metier \ Spectacle": private] => string (2) "11" ["heureFin": "modele \ metier \ Spectacle": private] => string (2) "12"}

Screen :


Attribution
- idEtab : char
- idTypeChambre : char
- idGroupe : char
- nombreChambres : integer
+ attribution
+ groupe
1
*
Groupe
+ id : char
+ nom : string
+ identiteRe...
+ adresseP...
+ nombreP...
+ nomPays :...
+ hebergem...
Utilisateur
+ id : integer
+ civilite : string
+ nom : string
+ prenom : string
+ email : string
+ login : string
+ mdp : string
+ attribution
*
+ attribution
*
+ type_Chambre
1
Type_Chambre
+ id : char
+ libelle : string
Lieu
+ id : integer
+ nom : string
+ adresse : string
+ capacite : BIGINT
+ groupe
1
+ lieu
1
+ etablissement
1
Etablissement
+ id : char
- nom : string
- adresseRue : string
- codePostal : char
- ville : string
- tel : string
- adresseElectronique : string
- type : integer
- civiliteResponsable : string
- nomResponsable : string
- prenomResponsable : string
+ etablissement
1
Offre
+ idEtab : char
+ idTypeChambre : char
+ nombreChambres :...
Spectacle
+ idGroupe : char
+ idLieu : integer
+ date : date
+ heureDebut : Time
+ heureFin : Time
+ idSpectacle : in...

```php
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>LieuDAO : test</title>
</head>

<body>

<?php

use controleur\Session;
use modele\dao\Bdd;
use modele\dao\LieuDao;
use modele\metier\Lieu;

require_once __DIR__ . '/../../includes/autoload.inc.php';

Session::demarrer();
Bdd::connecter();

$id = 2;
$lieu = null;

echo "<h2>Test LieuDAO</h2>";

// Test n°1
echo "<h3>1- Test getOneById</h3>";
try {
    $objet = LieuDao::getOneById($id);
    var_dump($objet);
} catch (Exception $ex) {
    echo "<h4>*** échec de la requête ***</h4>" . $ex->getMessage();
}

// Test n°2
echo "<h3>2- Test getAll</h3>";
try {
    $lesObjets = LieuDao::getAll();
    var_dump($lesObjets);
} catch (Exception $ex) {
    echo "<h4>*** échec de la requête ***</h4>" . $ex->getMessage();
}
```

```php
// Test n°3
echo "<h3>3- insert</h3>";
try {
    $id = 6;
    $lieu = new Lieu($id, "Lieu de test", "Adresse des tests", 500000);
    $ok = LieuDao::insert($lieu);
    if ($ok) {
        echo "<h4>ooo réussite de l'insertion ooo</h4>";
        $objetLu = LieuDao::getOneById($id);
        var_dump($objetLu);
    } else {
        echo "<h4>*** échec de l'insertion ***</h4>";
    }
} catch (Exception $e) {
    echo "<h4>*** échec de la requête ***</h4>" . $e->getMessage();
}

// Test n°3-bis
echo "<h3>3-bis insert déjà présent</h3>";
try {
    $newLieu = new Lieu($id, "Lieu de test", "Adresse des tests", 500000);
    $ok = LieuDao::insert($newLieu);
    if ($ok) {
        echo "<h4>*** échec du test : l'insertion ne devrait pas réussir  ***</h4>";
        $objetLu = LieuDao::getOneById($id);
        var_dump($objetLu);
    } else {
        echo "<h4>ooo réussite du test : l'insertion a logiquement échoué ooo</h4>";
    }
} catch (Exception $e) {
    echo "<h4>ooo réussite du test : la requête d'insertion a logiquement échoué ooo</h4>" . $e->getMessage();
}
```

```php
// Test n°4
echo "<h3>4- update</h3>";
try {
    $lieu->setNom("En fait c'était l'autre nom de test");
    $lieu->setAdresse("Une autre adresse");
    $lieu->setCapacite(10);
    $ok = LieuDao::update($id, $lieu);
    if ($ok) {
        echo "<h4>ooo réussite de la mise à jour ooo</h4>";
        $objetLu = LieuDao::getOneById($id);
        var_dump($objetLu);
    } else {
        echo "<h4>*** échec de la mise à jour ***</h4>";
    }
} catch (Exception $e) {
    echo "<h4>*** échec de la requête ***</h4>" . $e->getMessage();
}

// Test n°5
echo "<h3>5- delete</h3>";
try {
    $ok = LieuDao::delete($id);
//            $ok = GroupeDAO::delete("xxx");
    if ($ok) {
        echo "<h4>ooo réussite de la suppression ooo</h4>";
    } else {
        echo "<h4>*** échec de la suppression ***</h4>";
    }
} catch (Exception $e) {
    echo "<h4>*** échec de la requête ***</h4>" . $e->getMessage();
}

Bdd::deconnecter();
Session::arreter();
?>


</body>
</html>
```

```php
<?php


namespace modele\dao;


use modele\metier\Lieu;
use PDO;
use PDOStatement;

class LieuDao
{

    /**
     * Instancier un objet de la classe Lieu à partir d'un enregistrement de la table Lieu
     * @param array $enr
     * @return Lieu
     */
    protected static function enregVersMetier(array $enr)
    {
        return new Lieu($enr["ID"], $enr["NOM"], $enr["ADRESSE"], $enr["CAPACITE"]);
    }

    /**
     * Valorise les paramètres d'une requête préparée avec l'état d'un objet Lieu
     * @param Lieu $lieu
     * @param PDOStatement $stmt
     */
    protected static function metierVersEnreg(Lieu $lieu, PDOStatement $stmt)
    {
        // On utilise bindValue plutôt que bindParam pour éviter des variables intermédiaires
        // Note : bindParam requiert une référence de variable en paramètre n°2 ;
        // avec bindParam, la valeur affectée à la requête évoluerait avec celle de la variable sans
        // qu'il soit besoin de refaire un appel explicite à bindParam
        $stmt->bindValue(':id', $lieu->getId());
        $stmt->bindValue(':nom', $lieu->getNom());
        $stmt->bindValue(':adresse', $lieu->getAdresse());
        $stmt->bindValue(':capacite', $lieu->getCapacite());
    }
```

```php
/**
 * Retourne la liste de tous les lieux
 * @return array tableau d'objets de type Lieu
 */
public static function getAll()
{
    $lesObjets = array();
    $requete = "SELECT * FROM Lieu";
    $stmt = Bdd::getPdo()->prepare($requete);
    $ok = $stmt->execute();
    if ($ok) {
        // Tant qu'il y a des enregistrements dans la table
        while ($enreg = $stmt->fetch(PDO::FETCH_ASSOC)) {
            //ajoute un nouveau groupe au tableau
            $lesObjets[] = self::enregVersMetier($enreg);
        }
    }
    return $lesObjets;
}

/**
 * Recherche un lieu selon la valeur de son identifiant
 * @param string $id
 * @return Lieu le lieu trouvé ; null sinon
 */
public static function getOneById($id)
{
    $objetConstruit = null;
    $requete = "SELECT * FROM lieu WHERE id = :id";
    $stmt = Bdd::getPdo()->prepare($requete);
    $stmt->bindParam(":id", $id);
    $ok = $stmt->execute();
    // attention, $ok = true pour un select ne retournant aucune ligne
    if ($ok && $stmt->rowCount() > 0) {
        $objetConstruit = self::enregVersMetier($stmt->fetch(PDO::FETCH_ASSOC));
    }
    return $objetConstruit;
}
```

```php
    /**
     * Insérer un nouvel enregistrement dans la table à partir de l'état d'un objet métier
     * @param Lieu $lieu
     * @return bool
     */
    public static function insert(Lieu $lieu)
    {
        $requete = "INSERT INTO Lieu VALUES (:id, :nom, :adresse, :capacite)";
        $stmt = Bdd::getPdo()->prepare($requete);
        self::metierVersEnreg($lieu, $stmt);
        $ok = $stmt->execute();
        return ($ok && $stmt->rowCount() > 0);
    }
    /**
     * modifier un lieu (nom, adresse, capacite) en fonction de son id
     * @param Lieu $lieu
     * @return bool
     */
    public static function update($id, Lieu $lieu)
    {
        $ok = false;
        $requete = "UPDATE Lieu SET nom = :nom, adresse = :adresse, capacite = :capacite WHERE id = :id";
        $stmt = Bdd::getPdo()->prepare($requete);
        self::metierVersEnreg($lieu, $stmt);
        $stmt->bindParam(':id', $id);
        $ok = $stmt->execute();
        return ($ok && $stmt->rowCount() > 0);
    }
```

```php
/**
     * supprimez un lieu en fonction de son id
     * @param Lieu $lieu
     * @return bool
     */
    public static function delete($id)
    {
        $ok = false;
        $requete = "DELETE FROM Lieu WHERE id = :id";
        $stmt = Bdd::getPdo()->prepare($requete);
        $stmt->bindParam(":id", $id);
        $ok = $stmt->execute();
        $ok = $ok && ($stmt->rowCount() > 0);
        return $ok;
    }
}
```

Spectacle class

```php
<?php

namespace modele\metier;

class Spectacle
{

    /**
     * identifiant du spectacle
     * @var int
     */
    private $id;

    /**
     * identifiant du groupe qui va faire une représentation
     * @var string
     */
    private $idGroup;

    /**
     * lieu du spectacle
     * @var int
     */
    private $idLieu;

    /**
     * date du spectacle
     * @var string
     */
    private $date;

    /**
     * horaire de début du spectacle
     * @var string
     */
    private $heureDeb;

    /**
     * horaire de fin du spectacle
     * @var string
     */
    private $heureFin;
```

```php
/**
 * Spectacle constructor.
 * @param int $id
 * @param string $idGroup
 * @param int $idLieu
 * @param string $date
 * @param string $heureDeb
 * @param string $heureFin
 */
public function __construct(int $id, string $idGroup, int $idLieu, string $date, string $heureDeb, string $heureFin)
{
    $this->id = $id;
    $this->idGroup = $idGroup;
    $this->idLieu = $idLieu;
    $this->date = $date;
    $this->heureDeb = $heureDeb;
    $this->heureFin = $heureFin;
}

/**
 * @return int
 */
public function getId(): int
{
    return $this->id;
}

/**
 * @param int $id
 */
public function setId(int $id): void
{
    $this->id = $id;
}


/**
 * @return string
 */
public function getIdGroup(): string
{
    return $this->idGroup;
}
```

```php
/**
 * @param string $idGroup
 */
public function setIdGroup(string $idGroup): void
{
    $this->idGroup = $idGroup;
}

/**
 * @return int
 */
public function getIdLieu(): int
{
    return $this->idLieu;
}

/**
 * @param int $idLieu
 */
public function setIdLieu(int $idLieu): void
{
    $this->idLieu = $idLieu;
}

/**
 * @return string
 */
public function getDate(): string
{
    return $this->date;
}

/**
 * @param string $date
 */
public function setDate(string $date): void
{
    $this->date = $date;
}

/**
 * @return string
 */
public function getHeureDeb(): string
{
    return $this->heureDeb;
}
```

```php
    /**
     * @return string
     */
    public function getHeureFin(): string
    {
        return $this->heureFin;
    }

    /**
     * @param string $heureFin
     */
    public function setHeureFin(string $heureFin): void
    {
        $this->heureFin = $heureFin;
    }


}
```

SPECTACLEDAOTEST

```php
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>SpectacleDAOTest : test</title>
</head>

<body>

<?php

use controleur\Session;
use modele\dao\Bdd;
use modele\dao\SpectacleDAO;
use modele\metier\Spectacle;

require_once __DIR__ . '/../../includes/autoload.inc.php';

$id = 2;
Session::demarrer();
Bdd::connecter();

echo "<h2>1- SpectacleDAO</h2>";

// Test n°1
echo "<h3>Test getOneById</h3>";
try {
    $objet = SpectacleDAO::getOneById($id);
//    var_dump($objet);
} catch (Exception $ex) {
    echo "<h4>*** échec de la requête ***</h4>" . $ex->getMessage();
}

// Test n°2
echo "<h3>2- getAll</h3>";
try {
    $lesObjets = SpectacleDAO::getAll();
    var_dump($lesObjets);
} catch (Exception $ex) {
    echo "<h4>*** échec de la requête ***</h4>" . $ex->getMessage();
}
```

```php
// Test n°3
echo "<h3>3- insert</h3>";
try {
    $id = 6;
    $objet = new Spectacle($id, "g047", 2, "2019-03-17", "10:00", "15:00");
    $ok = SpectacleDAO::insert($objet);
    if ($ok) {
        echo "<h4>ooo réussite de l'insertion ooo</h4>";
        $objetLu = SpectacleDAO::getOneById($id);
        var_dump($objetLu);
    } else {
        echo "<h4>*** échec de l'insertion ***</h4>";
    }
} catch (Exception $e) {
    echo "<h4>*** échec de la requête ***</h4>" . $e->getMessage();
}

// Test n°3-bis
echo "<h3>3- insert déjà présent</h3>";
try {
    $objet = new Spectacle($id, "g047", 2, "2019-03-17", "10:00", "15:00");
    $ok = SpectacleDAO::insert($objet);
    if ($ok) {
        echo "<h4>*** échec du test : l'insertion ne devrait pas réussir  ***</h4>";
        $objetLu = SpectacleDAO::getOneById($id);
        var_dump($objetLu);
    } else {
        echo "<h4>ooo réussite du test : l'insertion a logiquement échoué ooo</h4>";
    }
} catch (Exception $e) {
    echo "<h4>ooo réussite du test : la requête d'insertion a logiquement échoué ooo</h4>" . $e->getMessage();
}
```

```php
// Test n°4
echo "<h3>4- update</h3>";
try {
    $objet->setDate("2019-05-16");
    $objet->setHeureDeb("09:00");
    $objet->setHeureFin("10:00");
    $ok = SpectacleDAO::update($id, $objet);
    if ($ok) {
        echo "<h4>ooo réussite de la mise à jour ooo</h4>";
        $objetLu = SpectacleDAO::getOneById($id);
        var_dump($objetLu);
    } else {
        echo "<h4>*** échec de la mise à jour ***</h4>";
    }
} catch (Exception $e) {
    echo "<h4>*** échec de la requête ***</h4>" . $e->getMessage();
}

// Test n°5
echo "<h3>5- delete</h3>";
try {
    $ok = SpectacleDAO::delete($id);
    if ($ok) {
        echo "<h4>ooo réussite de la suppression ooo</h4>";
    } else {
        echo "<h4>*** échec de la suppression ***</h4>";
    }
} catch (Exception $e) {
    echo "<h4>*** échec de la requête ***</h4>" . $e->getMessage();
}


Bdd::deconnecter();
Session::arreter();
?>


</body>
</html>
```

SPECTACLETEST

```php
<!Doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Spectacle:</title>
</head>
<body>
<?php

use modele\metier\Spectacle;

require_once __DIR__ . '/../../includes/autoload.inc.php';
echo "<h2>Test unitaire de la classe métier Spectacle</h2>";
$spec = new Spectacle(0, "g014", 1, "2019-06-11", "11", "12");
var_dump($spec);
?>
</body>
</html>
```

SPECTACLEDAO

```php
<?php

namespace modele\dao;

use modele\metier\Spectacle;
use PDO;
use PDOStatement;

/**
 * Description of SpectacleDAO
 *
 * @author Antoine
 */
class SpectacleDAO
{
    protected static function enregVersMetier(array $enreg)
    {
        $id = $enreg["ID"];
        $idGroup = $enreg['IDGROUPE'];
        $idLieu = $enreg['IDLIEU'];
        $date = $enreg['DATE'];
        $heureDeb = $enreg['HEUREDEBUT'];
        $heureFin = $enreg['HEUREFIN'];

        $unSpect = new Spectacle($id, $idGroup, $idLieu, $date, $heureDeb, $heureFin);

        return $unSpect;
    }

    protected static function metierVersEnreg(Spectacle $objetMetier, PDOStatement $stmt)
    {
        // On utilise bindValue plutôt que bindParam pour éviter des variables intermédiaires
        // Note : bindParam requiert une référence de variable en paramètre n°2 ;
        // avec bindParam, la valeur affectée à la requête évoluerait avec celle de la variable sans
        // qu'il soit besoin de refaire un appel explicite à bindParam
        $stmt->bindValue(':idGroup', $objetMetier->getIdGroup());
        $stmt->bindValue(':idLieu', $objetMetier->getidLieu());
        $stmt->bindValue(':date', $objetMetier->getdate());
        $stmt->bindValue(':heureDeb', $objetMetier->getheureDeb());
        $stmt->bindValue(':heureFin', $objetMetier->getheureFin());
    }
```

```php
public static function getAll()
{
    $lesObjets = array();
    $requete = "SELECT * FROM Spectacle";
    $stmt = Bdd::getPdo()->prepare($requete);
    $ok = $stmt->execute();
    if ($ok) {
        // Pour chaque enregisterement
        while ($enreg = $stmt->fetch(PDO::FETCH_ASSOC)) {
            // instancier un Spectacle et l'ajouter au tableau
            $lesObjets[] = self::enregVersMetier($enreg);
        }
    }
    return $lesObjets;
}

public static function getOneById(int $id)
{
    $objetConstruit = null;
    $requete = "SELECT * FROM Spectacle WHERE id = :id";
    $stmt = Bdd::getPdo()->prepare($requete);
    $stmt->bindParam(':id', $id);
    $ok = $stmt->execute();
    // attention, $ok = true pour un select ne retournant aucune ligne
    if ($ok && $stmt->rowCount() > 0) {
        $objetConstruit = self::enregVersMetier($stmt->fetch(PDO::FETCH_ASSOC));
    }
    return $objetConstruit;
}

/**
 * Insérer un nouvel enregistrement dans la table à partir de l'état d'un objet métier
 * @param Spectacle
 * $objet objet métier à insérer
 * @return boolean =FALSE si l'opération échoue
 */
public static function insert(Spectacle $objet)
{
    $requete = "INSERT INTO Spectacle (idGroupe, idLieu, date, heureDebut, heureFin) VALUES (:idGroup, :idLieu, :date, :heureDeb, :heureFin)";
    $stmt = Bdd::getPdo()->prepare($requete);
    self::metierVersEnreg($objet, $stmt);
    $ok = $stmt->execute();
    return ($ok && $stmt->rowCount() > 0);
}
```

```php
    /**
     * Mettre à jour enregistrement dans la table à partir de l'état d'un objet métier
     * @param int identifiant de l'enregistrement à mettre à jour
     * @param Spectacle $objet objet métier à mettre à jour
     * @return boolean =FALSE si l'opérationn échoue
     */
    public static function update(int $id, Spectacle $objet)
    {
        $ok = false;
        $requete = "UPDATE  Spectacle SET idGroupe=:idGroup, idLieu=:idLieu, date=:date, heureDebut=:heureDeb, heureFin=:heureFin
         WHERE id=:id";
        $stmt = Bdd::getPdo()->prepare($requete);
        self::metierVersEnreg($objet, $stmt);
        $stmt->bindParam(':id', $id);
        $ok = $stmt->execute();
        return ($ok && $stmt->rowCount() > 0);
    }
    /**
     * supprimez un enregistrement dans la table à partir de l'état d'un objet métier
     * @param $id métier à supprimer
     * @return boolean =FALSE si l'opérationn échoue
     */
    public static function delete($id)
    {
        $ok = false;
        $requete = "DELETE FROM Spectacle WHERE id=:id";
        $stmt = Bdd::getPdo()->prepare($requete);
        $stmt->bindParam(":id", $id);
        $ok = $stmt->execute();
        $ok = $ok && ($stmt->rowCount() > 0);
        return $ok;
    }
}
```