```php
<?php

/**
 * @file
 *
 * Courses API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

require_once('Database.php');

class Courses
{
    /**
     * Returns all courses.
     *
     * @return array
     */
    public static function getCourses()
    {
        $courses = array();
        $dbh = Database::getInstance();
        $sth = $dbh->query('SELECT * FROM courses');
        while ($row = $sth->fetch(PDO::FETCH_ASSOC))
            $courses[] = $row;
        return $courses;
    }
}

?>
```

```php
<?php

/**
 * @file
 *
 * Database API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

class Database
{
    /**
     * @var DSN
     */
    const DSN = 'mysql:dbname=jharvard_lecture3;host=127.0.0.1';

    /**
     * @var PASSWORD
     */
    const PASSWORD = 'crimson';

    /**
     * @var USER
     */
    const USER = 'jharvard';

    /**
     * Returns a PDO singleton.
     *
     * @return PDO
     */
    public static function getInstance()
    {
        static $dbh;
        if (!isset($dbh))
            $dbh = new PDO(self::DSN, self::USER, self::PASSWORD);
        return $dbh;
    }
}

?>
```

```
api/0/README

David J. Malan
malan@harvard.edu

Computer Science 164
Harvard College

Implements an API for courses using the singleton pattern.

Courses.php - DAO for courses
Database.php - Database singleton
test.php - test script
```

```php
<?php

require_once('Course.php');
require_once('Courses.php');

// get courses
$courses = Courses::getCourses();
print_r($courses);

?>
```

```php
<?php

/**
 * @file
 *
 * Course API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

class Course
{}

?>
```

```php
<?php

/**
 * @file
 *
 * Courses API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

require_once('Database.php');

class Courses
{
    /**
     * Returns all courses.
     *
     * @return array
     */
    public static function getCourses()
    {
        $courses = array();
        $dbh = Database::getInstance();
        $sth = $dbh->query('SELECT * FROM courses');
        while ($course = $sth->fetchObject('Course'))
            $courses[] = $course;
        return $courses;
    }
}

?>
```

```php
<?php

/**
 * @file
 *
 * Database API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

class Database
{
    /**
     * @var DSN
     */
    const DSN = 'mysql:dbname=jharvard_lecture3;host=127.0.0.1';

    /**
     * @var PASSWORD
     */
    const PASSWORD = 'crimson';

    /**
     * @var USER
     */
    const USER = 'jharvard';

    /**
     * Returns a PDO singleton.
     *
     * @return PDO
     */
    public static function getInstance()
    {
        static $dbh;
        if (!isset($dbh))
            $dbh = new PDO(self::DSN, self::USER, self::PASSWORD);
        return $dbh;
    }
}

?>
```

```
api/1/README

David J. Malan
malan@harvard.edu

Computer Science 164
Harvard College

Implements an API for courses using the singleton pattern.
Introduces entity.

Course.php - course entity
Courses.php - DAO for courses
Database.php - Database singleton
test.php - test script
```

```php
<?php

require_once('Course.php');
require_once('Courses.php');

// get courses
$courses = Courses::getCourses();
foreach ($courses as $course)
    echo $course->title . "\n";

?>
```

```php
<?php

/**
 * @file
 *
 * Course API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

require_once('Database.php');
require_once('Instructor.php');

class Course
{
    /**
     * Returns course's instructor(s).
     *
     * @return array
     */
    public function instructors()
    {
        $instructors = array();
        $dbh = Database::getInstance();
        $sth = $dbh->prepare('SELECT * FROM instructors ' .
         'WHERE id IN (SELECT instructor_id FROM course_instructors WHERE cat_num = :cat_num)');
        $sth->execute(array('cat_num' => $this->cat_num));
        while ($instructor = $sth->fetchObject('Instructor'))
            $instructors[] = $instructor;
        return $instructors;
    }
}

?>
```

```php
<?php

/**
 * @file
 *
 * Courses API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

require_once('Database.php');

class Courses
{
    /**
     * Returns all courses.
     *
     * @return array
     */
    public static function getCourses()
    {
        $courses = array();
        $dbh = Database::getInstance();
        $sth = $dbh->query('SELECT * FROM courses');
        while ($course = $sth->fetchObject('Course'))
            $courses[] = $course;
        return $courses;
    }
}

?>
```

```php
<?php

/**
 * @file
 *
 * Database API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

class Database
{
    /**
     * @var DSN
     */
    const DSN = 'mysql:dbname=jharvard_lecture3;host=127.0.0.1';

    /**
     * @var PASSWORD
     */
    const PASSWORD = 'crimson';

    /**
     * @var USER
     */
    const USER = 'jharvard';

    /**
     * Returns a PDO singleton.
     *
     * @return PDO
     */
    public static function getInstance()
    {
        static $dbh;
        if (!isset($dbh))
            $dbh = new PDO(self::DSN, self::USER, self::PASSWORD);
        return $dbh;
    }
}

?>
```

```php
<?php

/**
 * @file
 *
 * Instructor API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

class Instructor
{
    /**
     * Returns instructor's name.
     *
     * @return string
     */
    public function name()
    {
        return $this->first . ' ' . $this->last;
    }
}

?>
```

```
api/2/README

David J. Malan
malan@harvard.edu

Computer Science 164
Harvard College

Implements an API for courses using the singleton pattern.
Introduces a many-to-many relationship.

Course.php - course entity
Courses.php - DAO for courses
Database.php - Database singleton
Instructor.php - instructor entity
test.php - test script
```

```php
<?php

require_once('Course.php');
require_once('Courses.php');

// get courses
$courses = Courses::getCourses();
foreach ($courses as $course)
{
    echo $course->title . "\n";
    foreach ($course->instructors() as $instructor)
        echo " " . $instructor->name() . "\n";
}

?>
```

```php
<?php

/**
 * @file
 *
 * Course API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

require_once('Database.php');
require_once('Instructor.php');

class Course
{
    /**
     * Returns course's instructor(s).
     *
     * @return array
     */
    public function instructors()
    {
        $instructors = array();
        $dbh = Database::getInstance();
        $sth = $dbh->prepare('SELECT * FROM instructors ' .
         'WHERE id IN (SELECT instructor_id FROM course_instructors WHERE cat_num = :cat_num)');
        $sth->execute(array('cat_num' => $this->cat_num));
        while ($row = $sth->fetch(PDO::FETCH_ASSOC))
            $instructors[] = new Instructor($row['id'], $row['first'], $row['last']);
        return $instructors;
    }
}

?>
```

```php
<?php

/**
 * @file
 *
 * Courses API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

require_once('Database.php');

class Courses
{
    /**
     * Returns all courses.
     *
     * @return array
     */
    public static function getCourses()
    {
        $courses = array();
        $dbh = Database::getInstance();
        $sth = $dbh->query('SELECT * FROM courses');
        while ($course = $sth->fetchObject('Course'))
            $courses[] = $course;
        return $courses;
    }
}

?>
```

```php
<?php

/**
 * @file
 *
 * Database API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

class Database
{
    /**
     * @var DSN
     */
    const DSN = 'mysql:dbname=jharvard_lecture3;host=127.0.0.1';

    /**
     * @var PASSWORD
     */
    const PASSWORD = 'crimson';

    /**
     * @var USER
     */
    const USER = 'jharvard';

    /**
     * Returns a PDO singleton.
     *
     * @return PDO
     */
    public static function getInstance()
    {
        static $dbh;
        if (!isset($dbh))
            $dbh = new PDO(self::DSN, self::USER, self::PASSWORD);
        return $dbh;
    }
}

?>
```

```php
<?php

/**
 * @file
 *
 * Instructor API.
 *
 * @author David J. Malan <malan@harvard.edu>
 */

class Instructor
{
    /**
     * @var $id
     */
    private $id;

    /**
     * @var $first
     */
    private $first;

    /**
     * @var $last
     */
    private $last;

    /**
     * Constructor.
     *
     * @param string $id
     * @param string $first
     * @param string $last
     */
    public function __construct($id, $first, $last)
    {
        $this->id = $id;
        $this->first = $first;
        $this->last = $last;
    }

    /**
     * Returns instructor's name.
     *
     * @return string
     */
    public function name()
    {
```

```
49.         return $this->first . ' ' . $this->last;
50.     }
51. }
52.
53. ?>
```

```
api/3/README

David J. Malan
malan@harvard.edu

Computer Science 164
Harvard College

Implements an API for courses using the singleton pattern.
Introduces encapsulation and visibility.

Course.php - course entity
Courses.php - DAO for courses
Database.php - Database singleton
Instructor.php - instructor entity
test.php - test script
```

```php
<?php

require_once('Course.php');
require_once('Courses.php');

// get courses
$courses = Courses::getCourses();
foreach ($courses as $course)
{
    echo $course->title . "\n";
    foreach ($course->instructors() as $instructor)
        echo " " . $instructor->name() . "\n";
}

?>
```

```sql
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT=0;
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;


CREATE TABLE IF NOT EXISTS `courses` (
  `cat_num` char(5) NOT NULL,
  `course_group` varchar(128) NOT NULL,
  `num_int` int(11) DEFAULT NULL,
  `num_char` varchar(128) DEFAULT NULL,
  `title` varchar(1024) NOT NULL,
  `description` varchar(8192) DEFAULT NULL,
  PRIMARY KEY (`cat_num`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `course_instructors` (
  `cat_num` char(5) NOT NULL,
  `instructor_id` char(33) NOT NULL,
  KEY `cat_num` (`cat_num`),
  KEY `instructor_id` (`instructor_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `instructors` (
  `id` char(33) NOT NULL,
  `first` varchar(128) NOT NULL,
  `last` varchar(128) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


ALTER TABLE `course_instructors`
  ADD CONSTRAINT `course_instructors_ibfk_1` FOREIGN KEY (`cat_num`) REFERENCES `courses` (`cat_num`),
  ADD CONSTRAINT `course_instructors_ibfk_2` FOREIGN KEY (`instructor_id`) REFERENCES `instructors` (`id`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```