```php
<?php
/**
 * Lightweight Dummy ILS Driver -- Always returns hard-coded sample values.
 *
 * PHP version 5
 *
 * Copyright (C) Villanova University 2007.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2,
 * as published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * @category VuFind
 * @package  ILS_Drivers
 * @author   Andrew S. Nagy <vufind-tech@lists.sourceforge.net>
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link     http://vufind.org/wiki/building_an_ils_driver Wiki
 */
require_once 'Interface.php';

/**
 * Lightweight Dummy ILS Driver -- Always returns hard-coded sample values.
 *
 * @category VuFind
 * @package  ILS_Drivers
 * @author   Andrew S. Nagy <vufind-tech@lists.sourceforge.net>
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link     http://vufind.org/wiki/building_an_ils_driver Wiki
 */
class Sample implements DriverInterface
{
    /**
     * Get Status
     *
     * This is responsible for retrieving the status information of a certain
     * record.
     *
     * @param string $id The record id to retrieve the holdings for
     *
     * @return mixed     On success, an associative array with the following keys:
     * id, availability (boolean), status, location, reserve, callnumber; on
     * failure, a PEAR_Error.
     * @access public
     */
    public function getStatus($id)
    {
        $holding[] = array('availability' => 1,
                           'status' => 'Available',
                           'location' => '3rd Floor Main Library',
                           'reserve' => 'No',
                           'callnumber' => 'siehe Internformat',
                           'duedate' => '',
                           'number' => 1);
        return $holding;
    }

    /**
     * Get Statuses
     *
     * This is responsible for retrieving the status information for a
     * collection of records.
     *
     * @param array $ids The array of record ids to retrieve the status for
     *
     * @return mixed     An array of getStatus() return values on success,
     * a PEAR_Error object otherwise.
     * @access public
     */
    public function getStatuses($ids)
    {
        $items = array();
        foreach ($ids as $id) {
            $holding = array();
            $holding[] = array('availability' => 1,
                               'id' => $id,
                               'status' => 'Available',
                               'location' => '3rd Floor Main Library',
                               'reserve' => 'No',
                               'callnumber' => 'A1234.567',
                               'duedate' => '',
                               'number' => 1);
            $items[] = $holding;
        }
        return $items;
    }

    /**
     * Get Holding
     *
     * This is responsible for retrieving the holding information of a certain
     * record.
     *
     * @param string $id     The record id to retrieve the holdings for
     * @param array  $patron Patron data
     *
     * @return mixed     On success, an associative array with the following keys:
     * id, availability (boolean), status, location, reserve, callnumber, duedate,
     * number, barcode; on failure, a PEAR_Error.
     * @access public
     */
    public function getHolding($id, $patron = false)
    {
        return $this->getStatus($id);
```

```php
    }

    /**
     * Get Purchase History
     *
     * This is responsible for retrieving the acquisitions history data for the
     * specific record (usually recently received issues of a serial).
     *
     * @param string $id The record id to retrieve the info for
     *
     * @return mixed     An array with the acquisitions data on success, PEAR_Error
     * on failure
     * @access public
     */
    public function getPurchaseHistory($id)
    {
        return array();
    }

    /**
     * Get New Items
     *
     * Retrieve the IDs of items recently added to the catalog.
     *
     * @param int $page    Page number of results to retrieve (counting starts at 1)
     * @param int $limit   The size of each page of results to retrieve
     * @param int $daysOld The maximum age of records to retrieve in days (max. 30)
     * @param int $fundId  optional fund ID to use for limiting results (use a value
     * returned by getFunds, or exclude for no limit); note that "fund" may be a
     * misnomer - if funds are not an appropriate way to limit your new item
     * results, you can return a different set of values from getFunds. The
     * important thing is that this parameter supports an ID returned by getFunds,
     * whatever that may mean.
     *
     * @return array       Associative array with 'count' and 'results' keys
     * @access public
     */
    public function getNewItems($page, $limit, $daysOld, $fundId = null)
    {
        return array('count' => 0, 'results' => array());
    }

    /**
     * Find Reserves
     *
     * Obtain information on course reserves.
     *
     * @param string $course ID from getCourses (empty string to match all)
     * @param string $inst   ID from getInstructors (empty string to match all)
     * @param string $dept   ID from getDepartments (empty string to match all)
     *
     * @return mixed An array of associative arrays representing reserve items
     * (or a PEAR_Error object if there is a problem)
     * @access public
     */
    public function findReserves($course, $inst, $dept)
    {
        return array();
    }

    /**
     * Patron Login
     *
     * This is responsible for authenticating a patron against the catalog.
     *
     * @param string $username The patron username
     * @param string $password The patron password
     *
     * @return mixed           Associative array of patron info on successful login,
     * null on unsuccessful login, PEAR_Error on error.
     * @access public
     */
    public function patronLogin($username, $password)
    {
        return null;
    }
}
?>
```