Worksheet 7
Object-oriented PHP

IMPORTANT: You must catch up first!

Please note that if you are behind with any of the following exercises, you should catch up with them first before attempting object-oriented PHP. The exercises below are important to complete to get a C or B in the assignment or complete the TCA, while object-oriented PHP is only required for a Grade A on the assignment.

You MUST complete the following before you start this worksheet:

-Worksheet 3: PHP and databases, questions 1-8 (the main questions, not the "further questions")
-Worksheet 4: multi-page PHP applications, questions 1-7 (note that q5-7 are "further questions" but still important)
-Worksheet 5: logins, questions 1-9 (note that q8,9 are "further questions" but still important)
-Worksheet 6: SQL injection section. Ensure your SQL queries are converted to prepared statements.

1) Create a Song class to represent a song. It should have:

- six attributes, ID, title, artist, year, quantity and downloads
- a constructor, which takes the ID, title, artist, year, quantity and downloads as parameters, and sets the attributes to those values
- 'getter' methods for each attribute
- a printDetails() method, which prints the details of the song
- a setID() method, which sets the ID to a parameter passed in from outside.

Save it in a file called Song.php.

In a separate PHP file, e.g. song_test.php, create two test Song objects (using an ID of 0 initially), then set their IDs to 1 and 2 respectively using the setID() method, and display them using printDetails(). You'll need to include your song class with:

```
include("Song.php");
```

2) Convert your database code to use DAOs. First, add these additional methods to your Song class from question 1. The Song class will act as your data entity.

- download(): this should increase the downloads by one.
- order(): this should take a quantity as a parameter and reduce its quantity by that amount.

Next, create a SongDao class for accessing the 'wadsongs' table. It should include these methods:

- findSongById(): should take a song ID as a parameter, and return a Song object representing the song with that ID.

- searchByArtist(): should take an artist name as a parameter, search the wadsongs table for all songs by that artist, and return an array of Song objects, one Song object per result.

- updateSong(): should take a Song object as a parameter and update the appropriate record in the database (i.e. the record with the matching ID) to the values stored within the Song object.

Rewrite your searchresults.php, download.php and (if done) order2.php to use your DAO.

For downloading, you should call the methods in this order:

- call findSongById() to find the song with that ID
- call the download() method of Song to increase its downloads by one
- call updateSong() to update the record corresponding to the revised Song object in the database.

For ordering, you should call the methods in this order:

- call findSongById() to find the song using that ID
- call the order() method of Song to reduce its quantity by the specified amount
- call updateSong() to update the record corresponding to the revised Song object in the database.

3) a) Create a WebPage class to represent a web page. It should have the following attributes:

- $title : the page title.
- $stylesheet: the CSS stylesheet associated with this page: assume there's only one for the moment.
- $author - the page's author
- $year - the year of publication

and the following methods:

- a constructor, which takes the title, author and year as parameters.
- open(): Open the web page by echoing out the DOCTYPE and <html> lines.

- setCSS(): should set the CSS stylesheet. The stylesheet should be passed in as a parameter.
- writeHead(): should write the page's <head> section by echoing out the title as a <title> tag, and link in the page's stylesheet by writing a <link rel='stylesheet' … > line.
- writeHeading() : should give the page a heading by writing out the page title as an <h1> heading and displaying a horizontal rule underneath.
- writeFooter() : should write out a footer in a div containing a copyright message with the author and year of publication, e.g. "This website copyright (c) Fred Smith 2013"
- close(): closes the web page (echoes out </body> and </html>)

b) Modify your main HitTastic! pages (the index, the search results page, the buy scripts) to use your WebPage class. Ensure all the pages have a title and a footer. The WebPage class should be in a separate file and included with an **include** statement.

Note that the class will not be used for the main content of your web page (between the heading and the footer) - you will need to go out of PHP, add the HTML for the main content of the page, then go back into PHP again to write out the footer.

More advanced question

Modify your WebPage class to have the following attributes:

- $title : the page title.
- $stylesheets : an **array** of CSS stylesheets to be linked into the page.
- $links: an **associative array** of links to other pages, with the link text as the key and the page to link to as the value. The idea is that these will be written out as a sidebar.
- $author - the page's author
- $year - the year of publication

and the following methods:

- a constructor, which takes the title, author and year (as before)
- open(): Open the web page by echoing out the DOCTYPE and <html> lines (as before)
- addCSS(): should add a new CSS stylesheet to the array of stylesheets defined above. The stylesheet should be passed in as a parameter.
- writeHead(): should write the page's <head> section by echoing out the title as a <title> tag, and link in the page's stylesheet(s) by looping through the contents of $stylesheets and writing a <link rel='stylesheet' … > line for each.
- writeHeading() : should give the page a heading by writing out the page title as an <h1> heading and displaying a horizontal rule underneath.
- addLink() : add a link to the list of links. This should take two parameters: the link text and the page to link to, and add a corresponding entry to the associative array $links.

- writeSidebar(): this should write out a sidebar in a with an ID of *sidebar*, containing the links in $links (one per line)
- writeFooter(), as before
- close(), as before

Test it out by creating a page with an object of the WebPage class. The main page body should go in its own using ordinary HTML: just use the WebPage object to do the rest of the page.