Magic Methods

- Magic methods
  - Methods that start with two underscores (_ _). Do not define any functions starting with two underscores unless you want the magic functionality
- http://php.net/manual/en/language.oop5.magic.php

Object-Oriented Programming with PHP

- Online source:
  http://www.php.net/manual/en/language.oop5.php
- Syntax and semantic similar to Java
- Defining classes using class construct and creating objects using new
- Access specifiers (public, private, protected) are available and have the same meaning as in Java
- Access of members is through -> (rather than a period)
- Variables declared inside of an object are called properties
- You can verify whether a variable is associated with an object by using the is object function
- An object variable (as of PHP5) contains an identifier
  - Object accessors can use the identifier to find the actual object
- Relies on garbage collection
- **Example:** studentClassOne.php

Object-Oriented Programming with PHP

- $this reference allow us to access members of the object (cannot access members without the $this). Notice no $ for member
- A constructor has the following signature:

  __construct({parameters}); // **notice it is two underscores**
- **Example:** objects.php
- Parent constructors are not called implicitly if the child class defines a constructor. A call to the parent constructor is required in the child constructor
- A destructor has the following signature:

  __destruct()
- A destructor is called when no references to the object
  - Set variable to null ($myobj = null);
  - A destructor is called even if the script stopped using exit()
- Single Inheritance only
  - A class can extend another class via *extends*
  - Parent constructor can be called via parent::
- **Example:** objects.php

Object-Oriented Programming with PHP

- Method Overriding
  - A method can be overridden unless the parent class has defined it as final (using *final* construct)
  - A subclass method can call the method being overridden by using parent::
  - **Example:** objects.php
- Interfaces
  - PHP defines an interface concept similar to Java's interface
  - **Example:** objects.php
- Equality
  - == → true if two objects have same attributes and values and are instances of the same class
  - === → true if and only if the two variables refer to the same object instance
- Magic methods
  - Methods that start with two underscores __
  - __toString → similar to Java's toString
  - Do not define methods that start with __

Object-Oriented Programming with PHP

- Static
  - You can define class members using static
  - **Example:** objects.php
- Overloading
  - No support for method overloading as we know it in Java
  - PHP type system and support for variable number of arguments allows workarounds
- Constant declarations
  - Using const
  - Notice we do not use $ to access the constant
- Exception model similar to Java
  - A user-defined Exception class can be created by extending the ***Exception*** class.

Serialization

- serialization → converting data into a string of bytes
- unserialization → generating original data from a string of bytes
- serialization and unserialization is possible via the methods
  - Serialize
  - unserialize
- You can serialized arrays, objects, etc.
- **Example:** serialization.php
- Uses of serialization
  - To store information in a database
  - To pass information in sessions

Converting Special Characters to HTML Entities

- **htmlspecialchars**
  - Converts special characters (characters that have special significance in HTML) to HTML entities
  - Optional second parameter determines how to handle single and double quotes
  - Not all special characters are converted (only most useful for web programming). **Use htmlentities if you want to translate all html character entities**
  - Translations for htmlspecialchars
    - > → >
    - < → <
    - single quote → ' // when ENT_QUOTES is specified
    - double quote → " // when ENT_QUOTES is specified
    - & → &
  - **Example:** displayFileIncorrect.php/displayFile.php
    - displayFileIncorrect.php?filename=data.txt
    - displayFile.php?filename=data.txt
- **Htmlspecialchars/htmlentities**
  - Useful in preventing user-supplied text from containing HTML markup
  - Help us fight a web attack called Cross-Site Scripting (XSS)

Miscellaneous Functions

- file_get_contents
  - **Example:** getContents.php
- array_map
  - Allow us to apply processing to elements of array
  - **Example:** arrayMap.php
- Random integer in range (min,max) (inclusive)
  - rand(min,max)

Security

- **Protecting source code**
  - Although a user that executes a script cannot see the php source code they could see error messages which provide hints about code
  - Disable error reporting to the browser via the **display_errors** directive in php.ini
- **Email**
  - Least secure of internet protocols
  - Avoid sending sensitive information (e.g., passwords) over e-mail
  - Provide e-mail addresses in web sites in a way is not easily recognized by spam programs
    - Use at rather than @
    - Put an image with the e-mail
    - Avoid mailto
  - Encrypt the message using PGP (Pretty Good Privacy) or GPG (GNU Privacy Guard)

Security

- **Sanitize your input**
  - Use htmlspecialchars or htmlentities to turn input into harmless text
  - Use escapeshellcmd to sanitize string to be used in a program execution command
- **Validate your input**
  - Check the expected data type has been provided
  - Example:
    - Validating integers
    - Validating floats
- **Handle invalid/unexpected input**
  - readfileScript.php?recipe=cheesecake → OK
  - readfileScript.php?recipe=/etc/passwd → DANGER

Security

- **Controlling which files can be read**
  - Use php.ini option open_basedir → open_basedir, if set, limits all file operations to the defined directory
- **Do not run your web server as root**
- **For Password-protected sites**
  - **Approach not recommended**
    - Store encrypted password
    - Decrypt password and compare against user provided password
  - **Better approach**
    - Store encrypted password
    - Encrypt provided password and compare against stored password

Security

- **Set register_globals in php.in to Off**
  - With register_globals **on** a form variable name seat becomes available as a global variable $seat in the processing script
  - Malicious users could provide alternate values for variables like $seat (through GET parameters)
- **With register_globals set to Off**
  - Rather than variables being available as globals they are placed in arrays named after the environment that provides them
  - **Arrays:** $_POST, $_GET, $_SESSION, $_COOKIES, $_FILES

Security

- **File Uploads**
  - One of the most insecure operations for php
- **Things you can do through php.ini**
  - Leave **upload_tmp_dir** unassigned → Default appropriate for your system will be used
  - **upload_max_filesize** →limit the size of the file to upload
    - Keep in mind max_execution_time settings in php.ini. This represents the maximum execution time for each script in seconds
  - **post_max_size** → size of HTTP form submissions
    - Should take into account upload_max_filesize
- **What to do in the upload process**
  - Reduce the filename if necessary (long file names cause problems)
  - Remove spaces from the filename
  - Verify the extension of the file (e.g.,jpg)
  - After uploading a file change the file permissions to the most strict configuration (e.g., not executable)