LineCounter.class.php

```
<?php
//to get McCabes cyclomatic complexity of the Source-Code
//http://code.google.com/p/phpcyclo/source/browse/trunk/+phpcyclo+--username+bo
//http://github.com/pceres
define(MCCABE,true);
if(MCCABE){
require_once('lint_php_lib.php');
}
/* CLASS LineCounter
 *
 * This class counts the lines of code (LOC) in an existing Project-Folder.
 *
 * ############################
 * USAGE:
 *
 * $count = new LineCounter();
 * $count->setFolder('/var/www/search/');
 * $count->setFileTypes('php,txt');
 * OR
 * $count = new LineCounter('/var/www/','php,txt');
 *
 * $count->scanDirectory();
 * echo $count->getLines()." total Lines<br>";
 * echo $count->getSize()." byte<br>";
 * echo $count->getFiles()." Files<br>";
 * echo $count->getCommentLines()." CommentLine<br>";
 * echo $count->getEmptyLines()." EmtyLine<br>";
 * $count->displayExactOutput();
 *
 * ############################
 * METHODS:
 * __construct([string $folder, string $fileTypes])
 *
 * public void setFolder(string $folder)
 * public void setFileTypes(string $fileTypes)
 * public void setMcCabe(boolean $mcCabe)
 * public void scanDirectory(void)
 *
 * public int getLines(void)
 * public int getSize(void)
 * public int getFiles(void)
 * public int getCommentLines(void)
 * public int getEmptyLines(void)
 *
 * public void displayExactOutput(void)
 *
 * void countLines
```

```
 * void lineScan
 * void throwNotScannedException
 *
 *
 *
 *
 * @author Roy Bohn <roybohn11@web.de>  ->  www.roy-bohn.de
 * @date 2010-05-10 */
class LineCounter{
    // declare class variables
private $fileName;
private $folder;
private $file;
private $fileType;
private $fileHandle;
private $dirIterator;
private $line;
private $val;
private $value;
private $actVal;
private $percentComments;
private $cc;
private $ccWert;
private $ccFunction;
private $filePointer;
private $fileTypeArray  = array();
private $latestValues = array();
private $result = array();
/**
  * Constructor - Is called when the class is instanced
  *
  * @access: public
  * @param Str $folder
  * @param Str $fileTypes
  *
  * @return NONE */
public function __construct($folder='.',$fileTypes='php'){
//Deklarieren und Initialisieren der Variablen
$this->lines = 0;//globaler Zeilenzaehler
$this->commentLines = 0;//globaler Kommentarzaehler
$this->emptyLines = 0;//globaler Leerzeilenzaehler
$this->outputArray = array();//fuer die Statistikausgabe
$this->fileTypes = explode(",",$fileTypes);//inkludierte Dateitypen
$this->files = 0;//globaler Dateienzaehler
$this->size = 0;//globaler Dateigroessenzaehler
$this->folder = $folder;//Der Pfad
$this->isScanned = false;//fuer die Fehlerbehandlung scan-Indikator
$this->CC = array();//Array fuer McCabes cyclomatic complexity
```

```
if(MCCABE){
$this->mcCabe = true;
}else{
$this->mcCabe = false;
}
}
/**
  * scanDirectory() Scanning the given Directory recursive and filter the FileT
  *
  * @access: public
  *
  * @return NONE */
public function scanDirectory(){
//DirIterator setzen
$dirIterator = new RecursiveDirectoryIterator($this->folder);
//Ueber das DIR Iterieren
foreach (new RecursiveIteratorIterator($dirIterator) as $fileName => $file)
{
//fuer den Regex das Array mit | zusammenfuehern
$fileTypes = implode("|",$this->fileTypes);
//per Regex die Dateien filtern
if(preg_match("/^.+\.(" . $fileTypes . ")$/i",$fileName)){
$this->size += $file->getSize();
$this->countLines(basename($fileName), $file);
if($this->mcCabe)
$this->determineCC($fileName);
$this->files++;
}
}
//die Variable dient nur der Fehlerbehandlung weiter unten
$this->isScanned=true;
}//END:scanDirectory()
/**
  * countLines() Counts the occurent Lines for each file
  *
  * @access: private
  * @param Str $fileName
  * @param Str $file
  *
  * @return NONE */
private function countLines($fileName,$file){
//Letzten Stand festhalten, damit wir die aktiven Werte clever bestimmen koenne
$latestValues = array($this->commentLines,$this->emptyLines,$this->lines);
//Datei lesen
if ($fileHandle = fopen($file, 'r')) {
while (!feof($fileHandle)) {
if ($line = fgets($fileHandle)) {
$this->lineScan($line);
```

```
}
}
fclose($fileHandle);
}
//Das Array fuer die Detailierte Ausgabe fuellen
array_push($this->outputArray, array($fileName,$this->commentLines-
$latestValues[0],$this->emptyLines-$latestValues[1],$this->lines-
$latestValues[2]));
}//END:countLines()
/**
  * lineScan() categorize the given Line
  *
  * @access: private
  * @param Str $line
  *
  * @return NONE */
private function lineScan($line){
//der Regex passt auf Zeilen die folgendermaßen beginnen:
// '/*'  '*'  '*/'  '//'
if(preg_match('/^\/\/|^\/\*|^\*|^\*\//',trim($line))){
$this->commentLines++;
}
//wenn der Regex keine geschriebene Zeile erkennt deklarieren wir als leer
if(!preg_match('/./',trim($line))){
$this->emptyLines++;
}
//Zeilen hochzaehlen
$this->lines++;
}//END:lineScan();
/**
  * determineCC() determine the cyclomatic complexity of the given file
  *
  * @access: private
  * @param Str $file
  *
  * @return NONE */
private function determineCC($file){
//Wir oeffenen ein zweites mal die Datei, koennte optimiert werden
$filePointer = file_get_contents($file);
//wenn die Datei leer ist muehen wir uns nicht weiter
if($filePointer != ''){
$result = lint($filePointer,0);
foreach ($result[0]['lista_functions'] as $value){
if($value['function'] != ""){
$this->CC [basename($file)] [$value['function']] = $value['mc_count'];
}
}
}
```

```
}
/**
  * throwNotScannedException() Throw an Exception
  *
  * @access: private
  * @throws Exception
  * @return NONE */
private function throwNotScannedException(){
throw new Exception(
"Es wurde noch nicht gescannt bitte: LineCounter::scanDirectory() aufrufen!"
);
}
/* ### ### ### ### ### ### Getters and Setters ### ### ### ### ##*/
/**
  * getLines() get the whole Lines
  *
  * @access: public
  *
  * @return Int $lineCounter */
public function getLines(){
if($this->isScanned)
return $this->lines;
else
$this->throwNotScannedException();
}
/**
  * getFiles() get the whole Files
  *
  * @access: public
  *
  * @return Int $fileCounter */
public function getFiles(){
if($this->isScanned)
return $this->files;
else
$this->throwNotScannedException();
}
/**
  * getSize() get the calculated filesize
  *
  * @access: public
  *
  * @return Int $fileSize */
public function getSize(){
if($this->isScanned)
return $this->size;
else
$this->throwNotScannedException();
```

```
}
/**
  * getCommentLines() get the whole Commentlines
  *
  * @access: public
  *
  * @return Int $commentedLines */
public function getCommentLines(){
if($this->isScanned)
return $this->commentLines;
else
$this->throwNotScannedException();
}
/**
  * getEmptyLines() get the whole empty Lines
  *
  * @access: public
  *
  * @return Int $emptyLines */
public function getEmptyLines(){
if($this->isScanned)
return $this->emptyLines;
else
$this->throwNotScannedException();
}
/**
  * getExactOutput() get an array including statistics
  *
  * @access: public
  *
  * @return Array $statisticArray */
public function getExactOutput(){
if($this->isScanned)
return $this->outputArray;
else
$this->throwNotScannedException();
}
/**
  * displayExactOutput() show statistics
  *
  * @access: public
  * @throws Exception
  *
  * @return NONE */
public function displayExactOutput(){
if($this->isScanned){
echo "<table border=\"1\">";
 echo
"<tr><th>Dateiname</th><th>Kommentare</th><th>Leerzeilen</th><th>Gesamtzeilen</
;
```

```
if($this->mcCabe)
echo "<th>McCabes cc >10</th>";
echo "</tr>";
foreach($this->getExactOutput() as $val){
//Gesamtzeilen - Kommentarzeilen - Leerzeilen
$LOC = $val[3] - $val[2] - $val[1];
if($val[3]!=0){//wir duefen ja nicht durch 0 dividieren
$percentComments=round($val[1]/$val[3]*100);
}
//Man sagt, dass 30% Kommentare in den Quelltext gehoeren
if($percentComments <= 30){
echo "<tr style=\"background-color:#FFDDDD\">";
}else{
echo "<tr style=\"background-color:lightgreen\">";
}
foreach($val as $actVal){
echo "<td>$actVal</td>";
}
echo "<td style=\"color:red;font-weight:bold;\">".($LOC)."</td>";
echo "<td>$percentComments</td>";
if($this->mcCabe){
echo "<td>";
$cc = $this->CC [$val[0]];
if(isset($cc)){
foreach($cc as $ccFunction=>$ccWert){
if($ccWert>=10){
echo $ccFunction." : <b>".$ccWert."</b><br>";
}else{
//echo $ccFunction." : ".$ccWert."<br>";
}
}
}else{
echo "null";
}
echo "</td>";
}
echo "</tr>";
}
echo "</table>";
}else
$this->throwNotScannedException();
}//END:displayExactOutput();
/**
  * getCC() get the cyclomatic complexity
  *
  * @access: public
  *
  * @return Arr $cyclomaticComplexity */
```

```
public function getCC(){
return $this->CC;
}
/**
  * setFolder() set the folder to scan
  *
  * @access: public
  * @param Str $folder
  *
  * @return NONE */
public function setFolder($folder){
//Fehlerbehandlung
if($folder == '' || !isset($folder)){
throw new Exception("Keinen Pfad uebergeben!");
}elseif(!is_dir($folder)){
throw new Exception("'$folder' ist kein Verzeichnis!");
}elseif(!is_readable($folder)){
throw new Exception("Ich kann '$folder
' nicht lesen, bitte ueberpruefen Sie die Rechte!");
}else{
//alles okay
$this->folder = $folder;
}
}//END:setFolder
/**
  * setFileTypes() set the filetypes to include
  *
  * @access: public
  * @throws Exception
  * @param Str $fileType
  *
  * @return NONE */
public function setFileTypes($fileType){
//Fehlerbehandlung
if($fileType == '' || !isset($fileType)){
throw new Exception("Nichts uebergeben");
}elseif(preg_match('/,/',$fileType)){
//es ist eine Komma separierte Liste
$fileTypeArray = explode(",",$fileType);
$this->fileTypes = $fileTypeArray;
}else{
$this->fileTypes = array($fileType);
}
}
/**
  * setMcCabe() to turn McCabe off - this saves time
  *
  * @access: public
  * @throws Exception
  * @param bool $mcCabe
```

```
  *
  * @return NONE */
public function setMcCabe($mcCabe){
if(!MCCABE){
throw new Exception(
"Bitte MCCABE in der Datei LineCounter.class.php auf true setzen!");
}
if(is_bool($mcCabe)){
$this->mcCabe=$mcCabe;
}else{
throw new Exception("Bitte nur boolsche Werte uebergeben");
}
}
}//end LineCounter class
?>
```