

```

<?php
namespace base;
class EvidenceFileDetails {
    public $filename;
    public $originalFilename;
    public $filePath;
    public $fileType;
    public $fileSize;
    public $sha1Hash;
    public $directoryPath;
    function __construct($filePath) {
        $this->filePath = $filePath;
        $this->initialize();
    }
    private function initialize() {
        $methodsToCall =
        ['setFilename', 'setFileType', 'setFileSize', 'setSha1Hash', 'setDirectoryPath', 'setOriginalFilename'];
        foreach ($methodsToCall as $callMethod) $this->$callMethod();
    }
    private function setFilename() {
        $array = explode('/', $this->filePath);
        $this->filename = end($array);
    }
    private function setFileType() {
        $array = explode('.', $this->filename);
        $this->fileType = end($array);
    }
    private function setFileSize() {
        $this->fileSize = filesize($this->filePath);
    }
    private function setSha1Hash() {
        $this->sha1Hash = hash_file('sha1', $this->filePath);
    }
    private function setDirectoryPath() {
        $this->directoryPath = str_replace($this->filename, "", $this->filePath);
    }
    private function setOriginalFilename() {
        $this->originalFilename = str_replace('sha1_' . $this->sha1Hash . '_' . $this->filename);
    }
}

```

```

<?php
namespace base;
class DirectoryContents {
    private $directoryItems;
    private $directoryStructure;
    function __construct(DirectoryStructure $directoryStructure) {
        $this->directoryStructure = $directoryStructure;
        $this->directoryItems = new DirectoryItems;
    }
    public function getAllFilesInDirectory($directory) {
        $filesArray = [];
        $dirs = $this->directoryStructure->getDirectoryStructure($directory);
    }
}

```

```

        foreach ($this->directoryItems->getFiles($directory) as $file) $filesArray[] = $directory . $file;
        foreach ($dirs as $key => $item)
            foreach ($this->directoryItems->getFiles($item) as $file) $filesArray[] = $item . $file;
        return $filesArray;
    }
}

```

```

<?php
namespace base;
class DirectoryItems {
    public function getFiles($directory) {
        $files = [];
        foreach ($this->getDirectoryContents($directory) as $key => $content)
            if (strpos($content, '.') !== false) $files[] = $content;
        return $files;
    }
    public function getFolders($directory) {
        $folders = [];
        foreach ($this->getDirectoryContents($directory) as $key => $content)
            if (strpos($content, '.') === false) $folders[] = $content;
        return $folders;
    }
    public function getFilesAndFolders($directory) {
        return $this->getDirectoryContents($directory);
    }
    private function getDirectoryContents($directory) {
        $array = [];
        foreach (scandir($directory) as $key => $content)
            if (strpos($content, '.') !== 0 && strpos($content, '~') !== 0) $array[] = $content;
        return $array;
    }
}

```

```

<?php
namespace base;
class DirectoryStructure {
    private $directoryItems;
    private $dirs;
    function __construct() {
        $this->directoryItems = new DirectoryItems;
    }
    public function getDirectoryStructure($directory) {
        $this->dirs = [];
        $contents = $this->directoryItems->getFolders($directory);
        $this->recursivelyScanDirectoriesForSubDirectories($contents, $directory);
        return $this->dirs;
    }
    private function recursivelyScanDirectoriesForSubDirectories($contents, $dir) {
        $newDirs = [];
        foreach ($contents as $folder) $newDirs[] = $dir . $folder . '/';
        foreach ($newDirs as $newDir) $this->dirs[] = $newDir;
        foreach ($newDirs as $newDir)
            $this->recursivelyScanDirectoriesForSubDirectories($this->directoryItems->getFolders($newDir), $newDir);
    }
}

```

