

## AUTOMATYZACJA PRACY PROGRAMISTY WWW – COMPOSER

### SPIS TREŚCI

Spis treści .....	1
Cel zajęć.....	1
Rozpoczęcie .....	1
Jak wypełnić to sprawozdanie? .....	1
Pobranie i uruchomienie PHP .....	2
Inicjalizacja projektu z wykorzystaniem Composer .....	3
Utworzenie i wykonanie programu .....	6
Zarządzanie zależnościami .....	9
Wykorzystanie różnych poziomów logowania Monolog .....	10
Dependency Injection .....	11
Podsumowanie.....	13

### CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie umiejętności tworzenia projektów oraz łączenia zależności z wykorzystaniem narzędzia Composer.

### ROZPOCZĘCIE

Rozpoczęcie zajęć. Przedstawienie prowadzącego. Przedstawienie uczestników. Przedstawienie zasad laboratorium.

### JAK WYPEŁNIĆ TO SPRAWOZDANIE?

Zapisz ten plik na dysku twardym jako kopię. Zmień nazwę pliku:

- grN na odpowiedni numer grupy (np. gr3),
- nazwisko-imie na Twoje dane bez polskich znaków.

Otwórz kolejno Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe.

Zaktualizuj właściwości:

Właściwości:	Nazwa	Wartość	Typ
	Imie	Imie	Tekst
	Nazwisko	Nazwisko	Tekst
	Numer al...	00000	Tekst
	Kod kursu	AI2	Tekst
	Kod labor...	LAB A	Tekst
	Grupa	1	Liczba
	Wersja	1	Liczba

Czytaj tę instrukcję, wypełniaj polecenia, uzupełniaj zrzuty ekranu zgodnie z poleceniami.

Gotowe sprawozdanie wyślij w nieprzekraczalnym terminie **w postaci pliku PDF**.

## POBRANIE I URUCHOMIENIE PHP

Zaloguj się do systemu Windows / pulpitu zdalnego `rdp.wi.zut.edu.pl`:

- spoza sieci ZUT potrzebny VPN: <https://uci.zut.edu.pl/uslugi-uci/vpn.html>;
- nazwa użytkownika: `WIAD\ab12345`
- komputer: `rdp.wi.zut.edu.pl`

Utwórz katalog `I:\AI2-lab`. Jeśli musisz umieścić ten folder gdzie indziej – upewnij się, że nie ma spacji i ogonków.

Odwiedź stronę <https://windows.php.net/download/>. Pobierz `PHP 8.2.10 x64 NTS`.

Wypakuj pobrane repozytorium do `I:\AI2-lab\php-8.2.10-nts-win32-vs16-x64`.

Otwórz panel sterowania. W polu wyszukiwania wpisz `path`. Wybierz edycję zmiennych środowiskowych użytkownika. Znajdź zmienną `Path` i kliknij edycję. Dodaj ścieżkę `I:\AI2-lab\php-8.2.10-nts-win32-vs16-x64`.

Skopiuj plik `I:\AI2-lab\php-8.2.10-nts-win32-vs16-x64\php.ini-development` jako `php.ini`, po czym edytuj jego zawartość – odkomentuj poniższe ustawienia:

```
extension_dir = "ext"
...
extension=curl
extension=gd
extension=intl
extension=mbstring
extension=openssl
extension=pdo_sqlite
```

Utwórz katalog `I:\AI2-lab\labA`.

Otwórz ulubiony terminal (CMD, wiersz poleceń, PowerShell, Git Bash) i wejdź do katalogu `I:\AI2-lab\labA`.

Wykonaj komendę

```
php -i | Select-String -Pattern '(PHP Version)|(extension_dir)|(OpenSSL support)|(PDO drivers)|(GD Support)|intl|(cURL support)|multibyte'
```

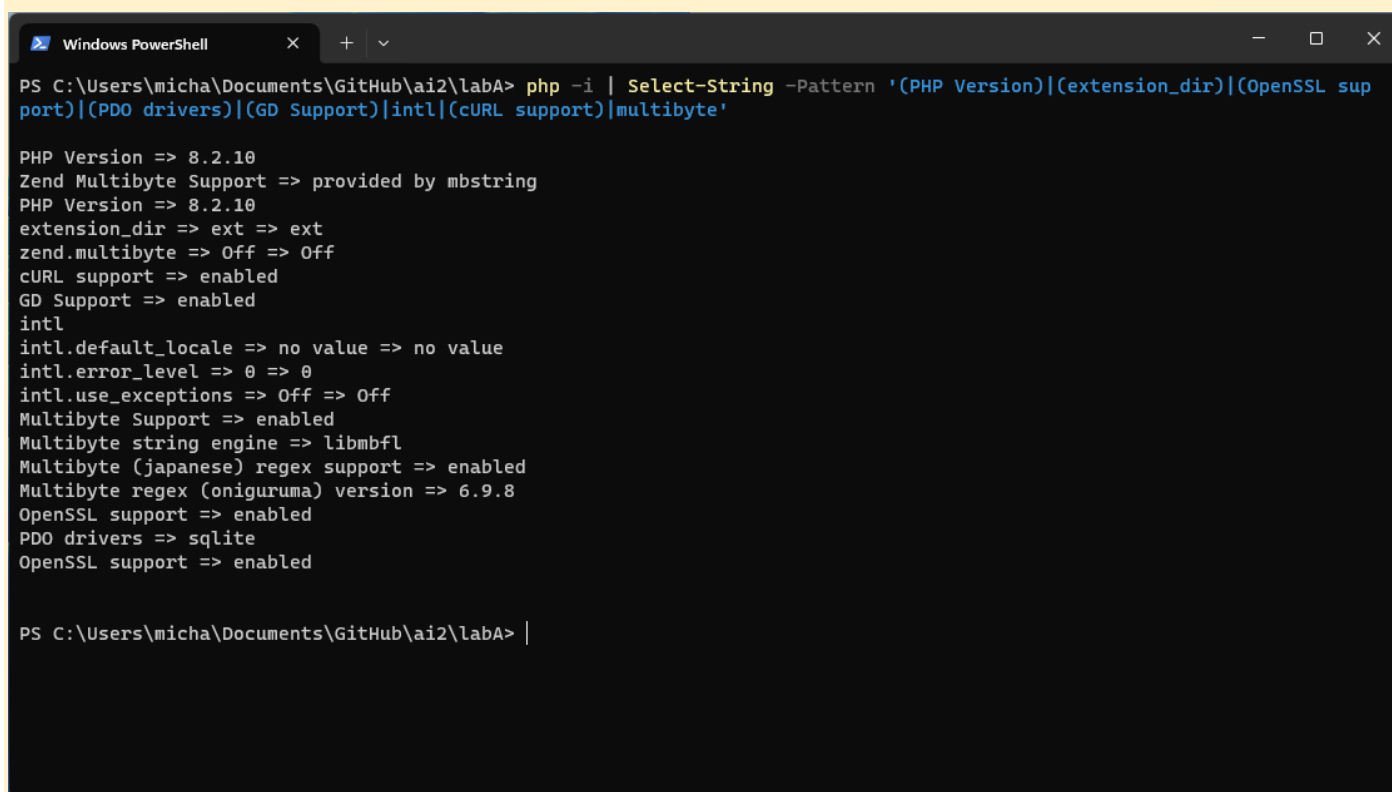
Oczekiwany wynik:

## AI2 LAB A – Nowakowski Michał – Wersja 1

```
PS C:\Users\artur\workspace\AI2-lab\labA> php -i | Select-String -Pattern '(PHP Version)|(extension_dir)|(OpenSSL support)|(PDO drivers)|(GD Support)|intl|(cURL support)|multibyte'
```

PHP Version => 8.2.10  
Zend Multibyte Support => provided by mbstring  
PHP Version => 8.2.10  
extension\_dir => ext => ext  
zend.multibyte => Off => Off  
cURL support => enabled  
GD Support => enabled  
intl  
intl.default\_locale => no value => no value  
intl.error\_level => 0 => 0  
intl.use\_exceptions => Off => Off  
Multibyte Support => enabled  
Multibyte string engine => libmbfl  
Multibyte (japanese) regex support => enabled  
Multibyte regex (oniguruma) version => 6.9.8  
OpenSSL support => enabled  
PDO drivers => sqlite  
OpenSSL support => enabled

Zastąp poniższy obrazek swoim zrzutem ekranu:



```
Windows PowerShell
PS C:\Users\micha\Documents\GitHub\ai2\labA> php -i | Select-String -Pattern '(PHP Version)|(extension_dir)|(OpenSSL support)|(PDO drivers)|(GD Support)|intl|(cURL support)|multibyte'
```

PHP Version => 8.2.10  
Zend Multibyte Support => provided by mbstring  
PHP Version => 8.2.10  
extension\_dir => ext => ext  
zend.multibyte => Off => Off  
cURL support => enabled  
GD Support => enabled  
intl  
intl.default\_locale => no value => no value  
intl.error\_level => 0 => 0  
intl.use\_exceptions => Off => Off  
Multibyte Support => enabled  
Multibyte string engine => libmbfl  
Multibyte (japanese) regex support => enabled  
Multibyte regex (oniguruma) version => 6.9.8  
OpenSSL support => enabled  
PDO drivers => sqlite  
OpenSSL support => enabled

```
PS C:\Users\micha\Documents\GitHub\ai2\labA> |
```

Punkty:	0	1
---------	---	---

## INICJALIZACJA PROJEKTU Z WYKORZYSTANIEM COMPOSER

Przejdź terminalem i eksploratorem plików do katalogu I : \AI2-lab\labA.

Pobierz archiwum PHAR composera w wersji 2.6.3 do katalogu I : \AI2-lab\labA:

- <https://getcomposer.org/download/2.6.3/composer.phar>

Sprawdź wersję composera:

```
php composer.phar --version
```

Zainicjuj projekt:

```
php composer.phar init
```

Ważne ustawienia:

- package name: inazwisko/lab-composer
- author: Imie Nazwisko
- package type: project
- license: MIT
- interaktywne wyszukiwanie pakietów: nie
- PSR-4 mapping: ENTER (zostawić domyślne)

Zweryfikuj ustawienia i zatwierdź utworzenie projektu:

```
Package name (<vendor>/<name>) [artur/lab-a]: akarczmarczyk/lab-composer
Description []:
Author [Artur Karczmarczyk <artur@ideaspot.pl>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
License []: MIT

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? noWould you li
ke to define your dev dependencies (require-dev) interactively [yes]? no
Add PSR-4 autoload mapping? Maps namespace "Akarczmarczyk\LabComposer" to the entered re
lative path. [src/, n to skip]:

{
    "name": "akarczmarczyk/lab-composer",
    "type": "project",
    "license": "MIT",
    "autoload": {
        "psr-4": {
            "Akarczmarczyk\\LabComposer\\": "src/"
        }
    },
    "authors": [
        {
            "name": "Artur Karczmarczyk",
            "email": "artur@ideaspot.pl"
        }
    ],
    "require": {}
}

Do you confirm generation [yes]?
Generating autoload files
Generated autoload files
PSR-4 autoloading configured. Use "namespace Akarczmarczyk\LabComposer;" in src/
Include the Composer autoloader with: require 'vendor/autoload.php':
```

Na koniec wykonaj polecenie:

```
php composer.phar install
```

Umieść poniżej zrzut ekranu z procesu inicjalizacji projektu composerem:

```

Windows PowerShell
"license": "MIT",
"autoload": {
    "psr-4": {
        "Mnowakowski\\LabComposer\\": "src/"
    }
},
"authors": [
    {
        "name": "Michał Nowakowski"
    }
],
"require": {}
}

Do you confirm generation [yes]?
Generating autoload files
Generated autoload files
PSR-4 autoloading configured. Use "namespace Mnowakowski\\LabComposer;" in src/
Include the Composer autoloader with: require 'vendor/autoload.php';
PS C:\Users\micha\Documents\GitHub\ai2\labA> php composer.phar install
No composer.lock file present. Updating dependencies to latest instead of installing from lock file. See https://getcomp
oser.org/install for more information.
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
PS C:\Users\micha\Documents\GitHub\ai2\labA> |

```

Umieść poniżej zrzut ekranu przedstawiający otrzymaną strukturę katalogów, z wykorzystaniem polecenia:

Get-ChildItem -Path . -Recurse -Force -ErrorAction SilentlyContinue | Select-Object  
FullName

```

Windows PowerShell
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
PS C:\Users\micha\Documents\GitHub\ai2\labA> Get-ChildItem -Path . -Recurse -Force -ErrorAction SilentlyContinue | Selec
t-Object FullName

FullName
-----
C:\Users\micha\Documents\GitHub\ai2\labA\src
C:\Users\micha\Documents\GitHub\ai2\labA\vendor
C:\Users\micha\Documents\GitHub\ai2\labA\composer.json
C:\Users\micha\Documents\GitHub\ai2\labA\composer.lock
C:\Users\micha\Documents\GitHub\ai2\labA\composer.phar
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\autoload.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\autoload_classmap.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\autoload_namespaces.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\autoload_psr4.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\autoload_real.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\autoload_static.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\ClassLoader.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\installed.json
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\installed.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\InstalledVersions.php
C:\Users\micha\Documents\GitHub\ai2\labA\vendor\composer\LICENSE

PS C:\Users\micha\Documents\GitHub\ai2\labA> |

```

Punkty:

0

1

## UTWORZENIE I WYKONANIE PROGRAMU

Otwórz katalog I:\AI2-lab\labA za pomocą Visual Studio Code lub PhpStorm. Utwórz dwa pliki, zmieniając odpowiednio przestrzeń nazw:

```
<?php // src/Duck.php
namespace Arkadiusz\LabComposer;

class Duck
{
    public function quack()
    {
        echo "Quack\n";
    }
}
```

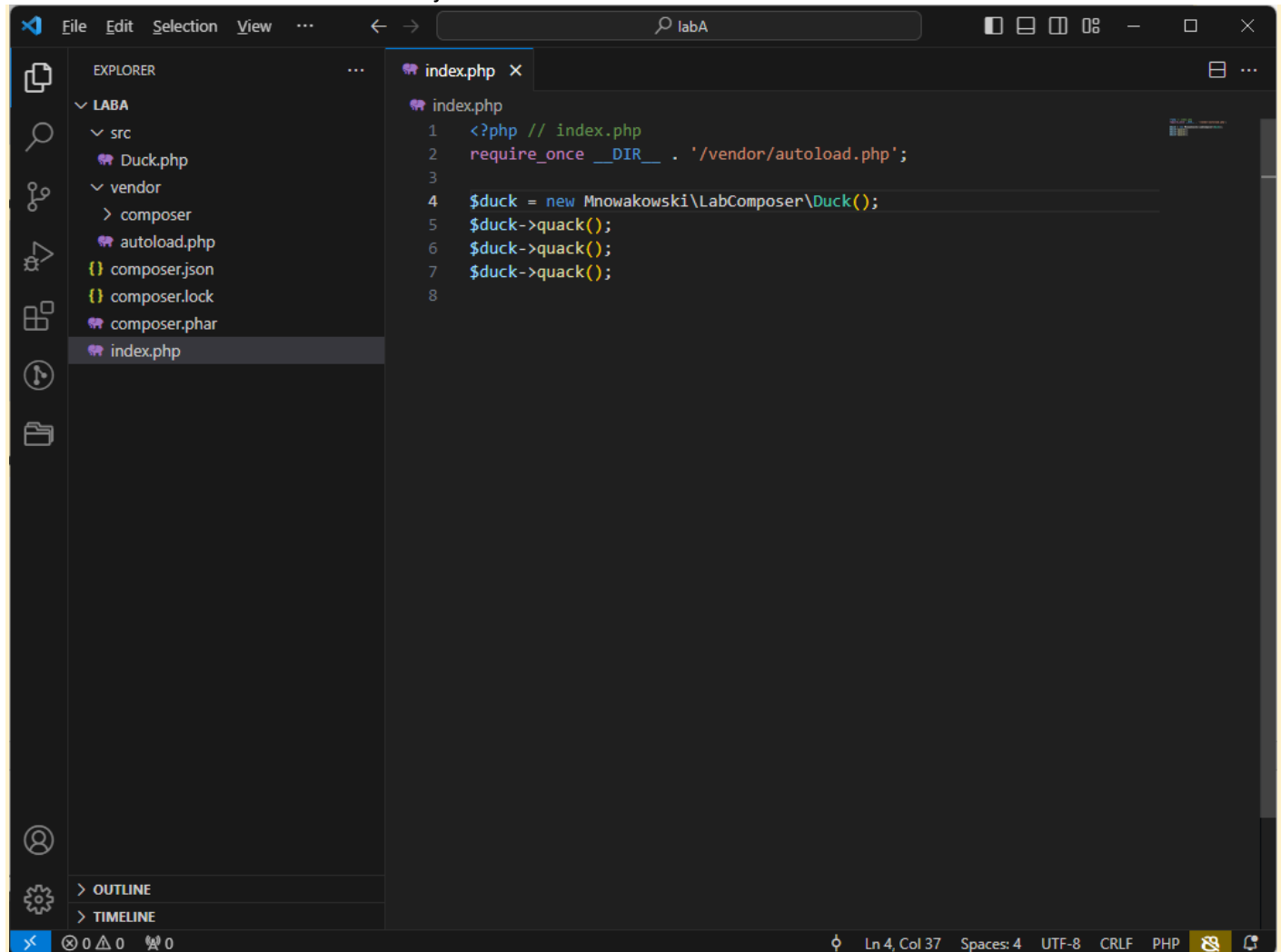
```
<?php // index.php
require_once __DIR__ . '/vendor/autoload.php';

$duck = new Arkadiusz\LabComposer\Duck();
$duck->quack();
$duck->quack();
$duck->quack();
```

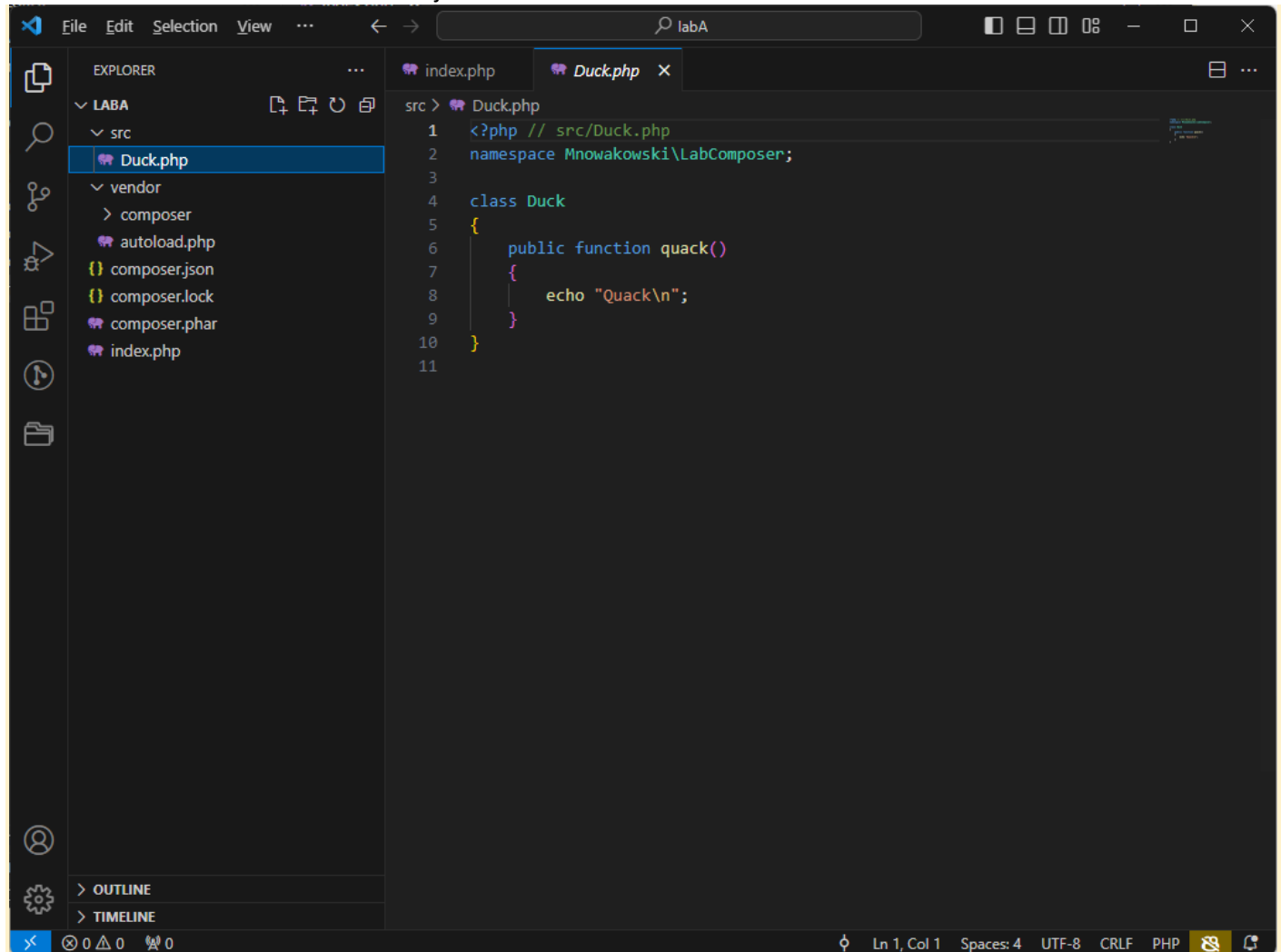
Następnie z poziomu terminala wykonaj program:

```
labA>php .\index.php
Quack
Quack
Quack
```

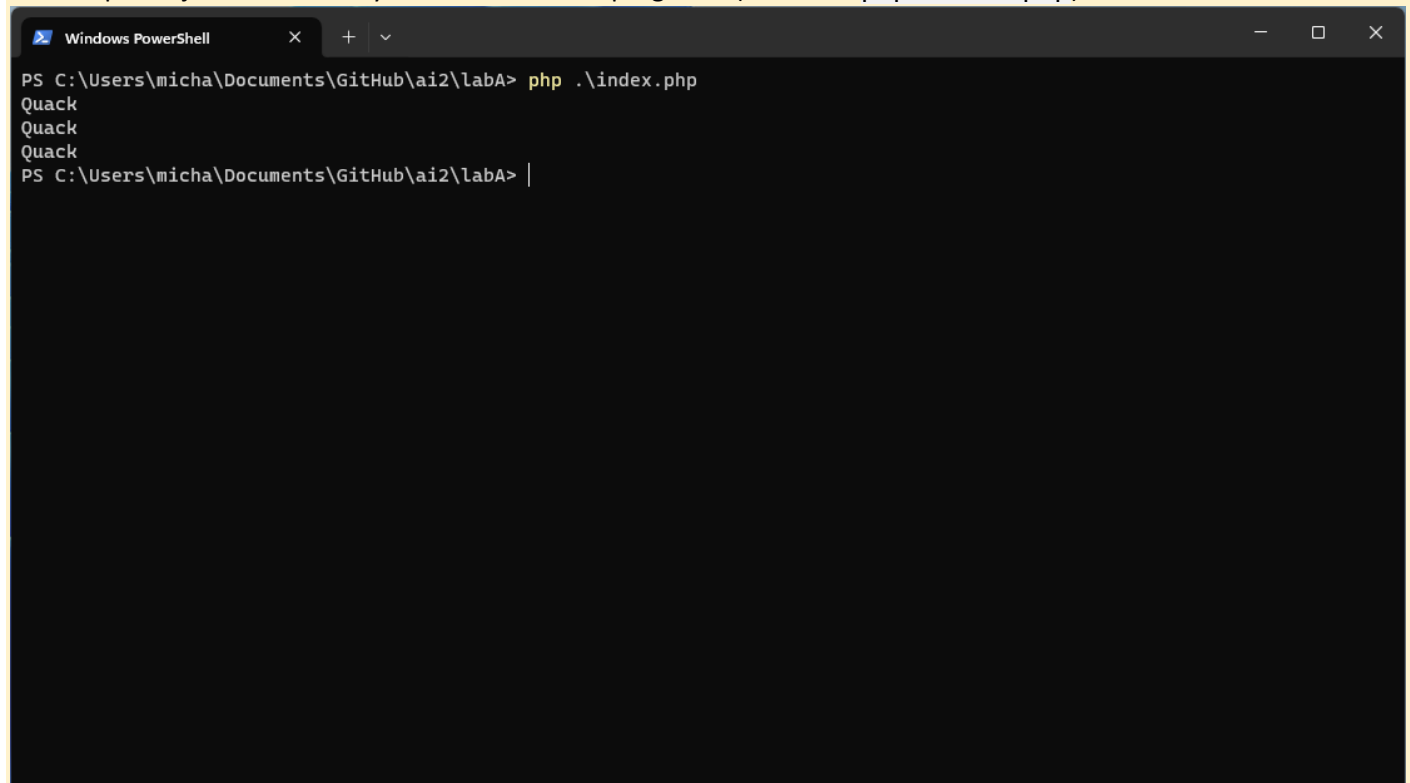
Umieść poniżej zrzut ekranu pliku index.php:



Umieść poniżej zrzut ekranu pliku src/Duck.php:



Umieść poniżej zrzut ekranu wywołania i działania programu (komenda `php index.php`):





Punkty:	0	1
---------	---	---

## ZARZĄDZANIE ZALEŻNOŚCIAMI

Przyjrzyj się strukturze programu, w szczególności plikom `composer.json` i `composer.lock`. Oba pliki muszą być commitowane. Dlaczego?

Zapoznaj się z zawartością katalogu `vendor`. Katalog `vendor` nie powinien być commitowany. Dlaczego?

Zainstaluj i odinstaluj pakiet `monolog/monolog`. Zbadaj jak zmienia się zawartość `composer.json`, `composer.lock` i katalogu `vendor`.

```
php composer.phar require monolog/monolog
php composer.phar remove monolog/monolog
```

Ponownie zainstaluj pakiet `monolog/monolog`:

```
php composer.phar require monolog/monolog
```

Po czym skasuj katalog `vendor` i spróbuj uruchomić program. Czy działa?

```
php index.php
```

Wykonaj `composer install` i zbadaj zawartość katalogu `vendor`. Spróbuj ponownie uruchomić program.

Omów jak zmienia się zawartość plików `composer.json`, `composer.lock` i katalogu `vendor`:

a) po zainstalowaniu pakietu `monolog/monolog`:

```
"require": {
  "monolog/monolog": "^3.5"
}
```

b) po odinstalowaniu pakietu `monolog/monolog`:

Nie zawiera `monolog/monolog`

Dlaczego po zainstalowaniu pakietu `monolog/monolog` i skasowaniu katalogu `vendor` aplikacja przestała się uruchamiać? Wyjaśnij. Umieść zrzut ekranu.

Po zainstalowaniu pakietu `Monolog`, jego kod jest dostępny w katalogu `vendor`. Skasowanie tego katalogu oznacza, że brakuje wymaganych plików i kodu źródłowego `Monolog`, co prowadzi do błędów podczas próby uruchomienia aplikacji.

```

or directory in C:\Users\micha\Documents\GitHub\ai2\labA\index.php on line 2
PHP Fatal error:  Uncaught Error: Failed opening required 'C:\Users\micha\Documents\GitHub\ai2\labA\vendor/autoload.php'
(include_path='.;C:\php\pear') in C:\Users\micha\Documents\GitHub\ai2\labA\index.php:2
Stack trace:
#0 {main}
  thrown in C:\Users\micha\Documents\GitHub\ai2\labA\index.php on line 2

Fatal error: Uncaught Error: Failed opening required 'C:\Users\micha\Documents\GitHub\ai2\labA\vendor/autoload.php' (inc
lude_path='.;C:\php\pear') in C:\Users\micha\Documents\GitHub\ai2\labA\index.php:2
Stack trace:
#0 {main}
  thrown in C:\Users\micha\Documents\GitHub\ai2\labA\index.php on line 2
PS C:\Users\micha\Documents\GitHub\ai2\labA> php .\index.php
PHP Warning:  require_once(C:\Users\micha\Documents\GitHub\ai2\labA\vendor/autoload.php): Failed to open stream: No such
file or directory in C:\Users\micha\Documents\GitHub\ai2\labA\index.php on line 2

Warning: require_once(C:\Users\micha\Documents\GitHub\ai2\labA\vendor/autoload.php): Failed to open stream: No such file
or directory in C:\Users\micha\Documents\GitHub\ai2\labA\index.php on line 2
PHP Fatal error:  Uncaught Error: Failed opening required 'C:\Users\micha\Documents\GitHub\ai2\labA\vendor/autoload.php'
(include_path='.;C:\php\pear') in C:\Users\micha\Documents\GitHub\ai2\labA\index.php:2
Stack trace:
#0 {main}
  thrown in C:\Users\micha\Documents\GitHub\ai2\labA\index.php on line 2

Fatal error: Uncaught Error: Failed opening required 'C:\Users\micha\Documents\GitHub\ai2\labA\vendor/autoload.php' (inc
lude_path='.;C:\php\pear') in C:\Users\micha\Documents\GitHub\ai2\labA\index.php:2
Stack trace:
#0 {main}
  thrown in C:\Users\micha\Documents\GitHub\ai2\labA\index.php on line 2
PS C:\Users\micha\Documents\GitHub\ai2\labA> |

```

Punkty:	0	1
---------	---	---

## WYKORZYSTANIE RÓŻNYCH POZIOMÓW LOGOWANIA MONOLOG

W tej części wykorzystamy bibliotekę monolog do logowania komunikatów i błędów. Zmodyfikuj kod `index.php`, żeby dodać logowanie do pliku `monolog.log`:

```

<?php // index.php
require_once __DIR__ . '/vendor/autoload.php';

$ds = DIRECTORY_SEPARATOR;
$log = new \Monolog\Logger("my_log");
$log->pushHandler(new \Monolog\Handler\StreamHandler(__DIR__ . $ds . 'monolog.log',
\Monolog\Logger::ERROR));
$log->error("This is some error.");
$log->warning("This is some warning.");
$log->notice("This is some notice.");
$log->info("This is some info.");
$log->debug("This is some debug.");

$duck = new \Akarczmarczyk\LabComposer\Duck();
$duck->quack();
$duck->quack();
$duck->quack();

```

Uruchom program:

```
php index.php
```

Kolejno zmieniaj `ERROR` w kodzie na `WARNING`, `NOTICE`, `INFO` i `DEBUG` i uruchamiaj program. Omów wpływ zmiany na liczbę zapisywanych logów. Omów korzyści praktyczne płynące z umieszczania funkcji logujących w całym programie i przełączania poziomu logów w jednym miejscu.

`ERROR` wyświetla tylko error

`Warning` wyświetla error i warning

`Notice` wyświetla error, warning, notice

`Info` wyświetla error, warning, notice, info

`Debug` wyświetla wszystkie

Wstaw reprezentatywny fragment pliku `monolog.log`:

```
[2023-10-29T15:59:44.742907+00:00] my_log.ERROR: This is some error. [] []  
[2023-10-29T15:59:44.745110+00:00] my_log.WARNING: This is some warning. [] []  
[2023-10-29T15:59:44.745180+00:00] my_log.NOTICE: This is some notice. [] []  
[2023-10-29T15:59:44.745240+00:00] my_log.INFO: This is some info. [] []  
[2023-10-29T15:59:44.745297+00:00] my_log.DEBUG: This is some debug. [] []
```

Punkty:	0	1
---------	---	---

## DEPENDENCY INJECTION

Dependency injection to technika programistyczna wchodząca w skład architektury heksagonalnej, który umożliwia uniezależnienie klasy od jej zależności. W tej sekcji wstrzykniemy do klasy `Duck` loggera, aby można było logować zdarzenia w instancjach tej klasy.

Zmodyfikuj klasę `Duck.php`, żeby:

- wykorzystywała technikę `Dependency Injection` do przekazywania obiektu klasy `LoggerInterface`;
- logowała (poziom `INFO`) stworzenie klasy `Duck`;
- logowała (poziom `DEBUG`) wykonanie metody `Duck::quack()`.

Następnie przetestuj uruchamianie kodu źródłowego z poziomami logowania `ERROR`, `WARNING`, `INFO`, `DEBUG`.

```
<?php // src/Duck.php  
namespace Akarczmarczyk\LabComposer;  
  
use Psr\Log\LoggerInterface;  
  
class Duck  
{  
    /** @var LoggerInterface */  
    private $logger;  
  
    public function __construct(LoggerInterface $logger = null)  
    {
```

```

        $this->logger = $logger;
        if ($this->logger) {
            $this->logger->info("Duck created.");
        }
    }

    public function quack()
    {
        if ($this->logger) {
            $this->logger->debug("Quack() executed.");
        }
        echo "Quack\n";
    }
}

```

```

<?php // index.php
require_once __DIR__ . '/vendor/autoload.php';

$ds = DIRECTORY_SEPARATOR;
$log = new \Monolog\Logger("my_log");
$log->pushHandler(new \Monolog\Handler\StreamHandler(__DIR__ . $ds . 'log.log',
\Monolog\Logger::ERROR));
$log->error("error");
$log->warning("warning");

$duck = new \Akarczmarczyk\LabComposer\Duck($log);
$duck->quack();
$duck->quack();
$duck->quack();

```

Kolejno zmieniaj **ERROR** w kodzie na **WARNING**, **NOTICE**, **INFO** i **DEBUG** i uruchamiaj program. Omów wpływ zmiany na liczbę zapisywanych logów. Zamieść odpowiedni wycinek pliku `monolog.log`:

**ERROR** wyświetla tylko error

**Warning** wyświetla error i warning

**Notice** wyświetla error, warning, notice

**Info** wyświetla error, warning, notice, info

**Debug** wyświetla wszystkie

```

... [2023-10-29T16:03:08.189599+00:00] my_log.ERROR: error [] []
[2023-10-29T16:03:08.191774+00:00] my_log.WARNING: warning [] []

```

```

1 [2023-10-29T16:05:27.407525+00:00] my_log.ERROR: error [] []
2 [2023-10-29T16:05:27.409716+00:00] my_log.WARNING: warning [] []
3 [2023-10-29T16:05:27.410146+00:00] my_log.INFO: Duck created. [] []
4 [2023-10-29T16:05:27.410213+00:00] my_log.DEBUG: Quack() executed. [] []
5 [2023-10-29T16:05:27.410460+00:00] my_log.DEBUG: Quack() executed. [] []
6 [2023-10-29T16:05:27.410647+00:00] my_log.DEBUG: Quack() executed. [] []
7

```

Dzięki temu możemy w łatwy sposób podmienić aktualnie używaną bibliotekę do logowania

Punkty:	0	1
---------	---	---

## PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Podczas tego laboratorium zdobyłem umiejętności związane z konfiguracją PHP, instalacją i korzystaniem z narzędzia Composer do zarządzania zależnościami w projektach PHP. Nauczyłem się, jak pobierać biblioteki i zarządzać nimi za pomocą plików `composer.json` i `composer.lock`. Zrozumiałem znaczenie katalogu `vendor` i dlaczego nie jest commitowany do repozytorium. Dodatkowo, poznałem podstawy obsługi Monolog do logowania różnych poziomów wiadomości, co jest istotne dla monitorowania i debugowania aplikacji.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.