AI2 LAB A	Bartman Aleksander Album 46466 Grupa 3	**Wersja 1**

AUTOMATYZACJA PRACY PROGRAMISTY WWW – COMPOSER

SPIS TREŚCI

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:

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 polecenia, 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:

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

```
PS C:\Users\Lenovo\Documents\GitHub\AI2\labA> php -i | Select-String -Pattern '(PHP Version)|(extension_dir)|(OpenSSL su
pport)|(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
```

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
```

AI2 LAB A – Bartman Aleksander – Wersja 1
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:

```
PS C:\Users\Lenovo\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://getc
omposer.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
```

Umieść poniżej zrzut ekranu przedstawiający otrzymaną strukturę katalogów, z wykorzystaniem polecenia:
`Get-ChildItem -Path . -Recurse -Force -ErrorAction SilentlyContinue | Select-Object FullName`

```
PS C:\Users\Lenovo\Documents\GitHub\AI2\labA> Get-ChildItem -Path . -Recurse -Force -ErrorAction SilentlyContinue | S
elect-Object FullName

FullName
--------
C:\Users\Lenovo\Documents\GitHub\AI2\labA\src
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor
C:\Users\Lenovo\Documents\GitHub\AI2\labA\composer.json
C:\Users\Lenovo\Documents\GitHub\AI2\labA\composer.lock
C:\Users\Lenovo\Documents\GitHub\AI2\labA\composer.phar
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\autoload.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\autoload_classmap.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\autoload_namespaces.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\autoload_psr4.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\autoload_real.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\autoload_static.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\ClassLoader.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\installed.json
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\installed.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\InstalledVersions.php
C:\Users\Lenovo\Documents\GitHub\AI2\labA\vendor\composer\LICENSE
```

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 przestrzenie nazw:

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

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

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

$duck = new \Akarczmarczyk\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:

index.php

```php
<?php
require_once __DIR__ . '/vendor/autoload.php';
$duck = new \Abartman\LabComposer\Duck();
$duck->quack();
$duck->quack();
$duck->quack();

```

Umieść poniżej zrzut ekranu pliku src/Duck.php:

src > Duck.php > Duck

```php
<?php
namespace Abartman\LabComposer;
class Duck
{
    public function quack()
    {
        echo "Quack\n";
    }
}
```

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

```
PS C:\Users\Lenovo\Documents\GitHub\AI2\labA> php .\index.php
Quack
Quack
Quack
```

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:

W pliku composer.json została dodana nowa zależność Monologa oraz wersje, którą Composer może zainstalować.
W pliku composer.lock znajdą się szczegółowe informacje o wersjach Monologa i jego zależności, które zostały zainstalowane.
Katalog vendor został dodany folder monolog, w którym znajduje się np. Kod źródłowy Monologa, katalog src (pliki źródłowe Monologa), pliki konfiguracyjne Monologa, pliki README oraz dokumentacja oraz pliki zależności Monologa i inne
b) po odinstalowaniu pakietu monolog/monolog:

W pliku composer.json została usunięta zależność Monologa oraz jego wersja.
W pliku composer.lock zostały usunięte informacje o Monologu oraz jego zależnościach.
W katalogu vendor został usunięty katalog Monolog wraz z całą zawartością.

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

Dzieje się tak, ponieważ katalog „vendor” zawiera wszystkie zależności i biblioteki, które są niezbędne do prawidłowego działania aplikacji. Po usunięciu katalogu aplikacja nie może znaleźć wymaganych plików i klas co prowadzi do błędów.

```
PS C:\Users\Lenovo\Documents\GitHub\Ai2\lab1\labA> php index.php
PHP Warning:  require_once(C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA/vendor/autoload.php): Failed to open stream
: No such file or directory in C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA\index.php on line 2

Warning: require_once(C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA/vendor/autoload.php): Failed to open stream: No
such file or directory in C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA\index.php on line 2
PHP Fatal error:  Uncaught Error: Failed opening required 'C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA/vendor/auto
load.php' (include_path='.;C:\php\pear') in C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA\index.php:2
Stack trace:
#0 {main}
  thrown in C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA\index.php on line 2

Fatal error: Uncaught Error: Failed opening required 'C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA/vendor/autoload.
php' (include_path='.;C:\php\pear') in C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA\index.php:2
Stack trace:
#0 {main}
  thrown in C:\Users\Lenovo\Documents\GitHub\AI2\lab1\labA\index.php on line 2
```

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
```

Sprawdź zawartość pliku `monolog.log`.

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.

Po każdej zmianie poziomu logowania pojawiały się nowe rodzaje wpisów na początku były same ERROR-y później zostały dodane jeszcze wpisy WARNING-ów, przy NOTICE zostały dodane wpisy dotyczące notatek itd.
Korzyści jakie płyną z takiego rozwiązania to przede wszystkim:
-łatwa kontrola nad tym jakie informacje są logowane bez konieczności modyfikowania kodu w wielu miejscach
-można łatwo przełączać poziomy logów co przyśpiesza prace podczas np. debugowania można przełączyć na DEBUG, aby uzyskać szczegółowe informacje o działaniu programu.
-gromadzenie logów w jednym miejscu przyśpiesza czas analizowania oraz debugowania aplikacji, ponieważ wszystko co jest nam potrzebne jest w jednym pliku

Wstaw reprezentatywny fragment pliku `monolog.log`:

```
10    [2023-10-09T20:32:39.093716+00:00] my_log.NOTICE: This is some notice. [] []
11    [2023-10-09T20:32:39.093997+00:00] my_log.INFO: This is some info. [] []
12    [2023-10-09T20:32:47.074114+00:00] my_log.ERROR: This is some error. [] []
13    [2023-10-09T20:32:47.075661+00:00] my_log.WARNING: This is some warning. [] []
14    [2023-10-09T20:32:47.075722+00:00] my_log.NOTICE: This is some notice. [] []
15    [2023-10-09T20:32:47.075763+00:00] my_log.INFO: This is some info. [] []
16    [2023-10-09T20:32:47.075787+00:00] my_log.DEBUG: This is some debug. [] []
17
```

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
<?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
<?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`:

```
7   [2023-10-09T21:01:35.857276+00:00] my_log.ERROR: error [] []
8   [2023-10-09T21:01:35.858738+00:00] my_log.WARNING: warning [] []
9   [2023-10-09T21:01:35.859039+00:00] my_log.INFO: Duck created. [] []
10  [2023-10-09T21:01:58.884148+00:00] my_log.ERROR: error [] []
11  [2023-10-09T21:01:58.885596+00:00] my_log.WARNING: warning [] []
12  [2023-10-09T21:01:58.885882+00:00] my_log.INFO: Duck created. [] []
13  [2023-10-09T21:01:58.886205+00:00] my_log.DEBUG: Quack() executed. [] []
14  [2023-10-09T21:01:58.886563+00:00] my_log.DEBUG: Quack() executed. [] []
15  [2023-10-09T21:01:58.886784+00:00] my_log.DEBUG: Quack() executed. [] []
```

Dzięki dodaniu konstruktora z argumentem obiektu klasy „LoggerInterface” możemy przekazywać logera do istancji klasy „Duck” z zewnątrz (mechanizm Dependency Injection). Klasa Duck teraz loguje informacje o swoim utworzeniu (poziom INFO) oraz wykonaniu metody quack() (poziom DEBUG). Poziom logowania można kontrolować z poziomu pliku index.php.

Punkty:	0	1

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Podczas tego laboratorium nauczyłem się inicjalizować projekt za pomocą narzędzia Composer. Zaznajomiłem się z budową projektu oraz zawartością ważnych plików takich jak np. composer.json i composer.lock oraz katalogu vendor. Zarządzaniem zależnościami w projekcie za pomocą Composer (dodawanie oraz usuwanie pakietów). Wykorzystania różnych poziomów logowania przy użyciu biblioteki Monolog. Jak zaimplementować mechanizm Dependency Injection do klasy „Duck” co pozwoliło mi na przekazywanie logera jako zależności do klasy.

AI2 LAB A – Bartman Aleksander – Wersja 1
Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.