

PHP obiektowy — sklep w klasach

Czas: 2 × 45 min

praca indywidualna

Cel: Zaprojektowanie i napisanie kilku współpracujących klas PHP z hermetyzacją, interfejsem i przestrzeniami nazw oraz sprawdzenie ich działania skryptem testów.

Po zajęciach potrafisz:

- tworzy klasy z konstruktorem z promocją właściwości i walidacją danych (wyjątki)
- ukrywa stan obiektu (`private`, `readonly`) i udostępnia go przez metody
- definiuje interfejs i kilka jego implementacji, korzystając z polimorfizmu
- porządkuje klasy w przestrzeniach nazw i plikach zgodnie z PSR-4
- uruchamia skrypt testów w konsoli i interpretuje jego wyniki

Sprzęt i materiały

- komputer z Windows 11 i XAMPP (PHP 8.2 lub nowszy)
- folder `r2_15_start` z Pliki_do_zadan

Oprogramowanie

- **XAMPP** — PHP w konsoli:
`C:\xampp\php\php.exe` (przycisk Shell w panelu XAMPP)
[apachefriends.org/download.html](https://httpd.apache.org/download.html)
- **Visual Studio Code** — terminal: Ctrl+`;
rozszerzenie PHP Intelephense
podpowiada metody klas
code.visualstudio.com

Przygotowanie przed zajęciami

- Przeglądnij wykład 2.12: klasy, konstruktor z promocją właściwości, interfejsy, przestrzenie nazw.

Bezpieczeństwo

- Nie modyfikuj testów w `test.php`, aby „przeszły” — popraw klasy. Możesz dopisywać własne testy na końcu pliku.

Przebieg zadania

1 Klasa Produkt

Skopiuj `r2_15_start` do `htdocs\sklep_oop`. W terminalu VS Code uruchamiaj `php test.php` (jeśli polecenie `php` nie działa, użyj `C:\xampp\php\php.exe test.php`). Każdy plik zaczyna od `<?php`, `declare(strict_types=1);` i deklaracji `namespace`.

```
namespace Sklep;                                // plik src/Produkt.php
class Produkt
{
    public const STAWKI_VAT = [0, 5, 8, 23];
    __construct(string $nazwa, float $cenaNetto, int $stawkaVat = 23)
    {
        getNazwa(): string    getCenaNetto(): float    getStawkaVat(): int
        cenaBrutto(): float    // zaokrąglona do 2 miejsc
        obnizCene(float $procent): void // procent z przedziału (0, 100)
    }
}
```

1. Uruchom `php test.php` przed napisaniem kodu — wszystkie sekcje mają wynik `[--]`.
2. Napisz klasę `Produkt` według ramki. Właściwości zadeklaruj w konstruktorze (promocja) jako `private`. Pusta nazwa, ujemna cena lub stawka spoza `STAWKI_VAT` mają zgłaszać `InvalidArgumentException`.
3. Uruchom testy — sekcja `Produkt` ma $6 \times [OK]$. Sprawdź, co się stanie, gdy w `test.php` dopiszesz na chwilę `echo $mysz->cenaNetto;` (zanotuj komunikat i usuń wiersz).

2 Klasa Koszyk

Koszyk przechowuje pozycje w prywatnej tablicy: klucz = nazwa produktu, wartość = `['produkt' => Produkt, 'ilosc' => int]`.

```
class Koszyk                                // src/Koszyk.php
{
    dodaj(Produkt $produkt, int $ilosc = 1): void // ten sam produkt → zwiększa ilość
    usun(string $nazwa): void
    zmienIlosc(string $nazwa, int $ilosc): void    // 0 lub mniej → usuwa pozycję
    czyPusty(): bool    liczbaSztuk(): int    wartoscBrutto(): float
    getPozycje(): array
}
```

1. Napisz klasę `Koszyk` według ramki. Ilość mniejsza niż 1 w `dodaj()` i zmiana ilości produktu, którego nie ma w koszyku, zgłaszają `InvalidArgumentException`.
2. Uruchom testy — sekcja `Koszyk` ma $6 \times [OK]$. W sprawozdaniu wyjaśnij, dlaczego `getPozycje()` zwracająca tablicę nie pozwala zmienić zawartości koszyka z zewnątrz.

3 Interfejs SposobDostawy

Przestrzeń nazw `Sklep\Dostawa`, pliki w `src/Dostawa/`.

```
interface SposobDostawy
{
    nazwa(): string
    koszt(float $wartoscKoszyka): float
}

Kurier          — „Kurier”, 19.99 zł; od 300 zł gratis
Paczkomat       — konstruktor z kodem punktu, nazwa „Paczkomat KOD”, 12.99 zł; od 200 zł gratis
OdbiorOsobisty — „Odbiór osobisty w sklepie”, 0 zł
```

1. Napisz interfejs `SposobDostawy` i trzy klasy, które go implementują (`implements`).
2. Uruchom testy — sekcja `Dostawa` ma $4 \times [OK]$. Dopisz własną klasę `PocztaPolska` (koszt 15.50 zł, bez darmowej dostawy) i własny test jej kosztu.

4 Klasa Zamowienie i podsumowanie

```
class Zamowienie // src/Zamowienie.php, use
{
    Sklep\Dostawa\SposobDostawy;
    private static int $licznik = 0;
    public readonly string $numer; // ZAM/2026/0001, ZAM/2026/0002, ...
    __construct(Koszyk $koszyk, SposobDostawy $dostawa, string $email)
    {
        $this->kosztDostawy();
        $this->razem();
        $this->podsumowanie();
    }
}
```

1. Napisz klasę `Zamowienie`. Konstruktor odrzuca pusty koszyk i niepoprawny e-mail (`filter_var`), zwiększa statyczny licznik i nadaje numer `sprintf('ZAM/%s/%04d', date('Y'), ...)`.
2. `kosztDostawy()` pyta obiekt dostawy o koszt dla wartości koszyka — klasa `Zamowienie` nie wie, czy to kurier, czy paczkomat. `podsumowanie()` zwraca wielowierszowy tekst: numer i e-mail, pozycje (nazwa, ilość, cena), dostawa z kosztem, RAZEM; kwoty formatuj `number_format($x, 2, ',', ' ')`.
3. Uruchom testy — wynik ma być 23 OK, 0 błędów (plus Twoje testy). Zrób zrzut terminala.
4. Utwórz `sklep.php` — stronę HTML, która tworzy 3 produkty, koszyk i zamówienie z odbiorem osobistym, a podsumowanie wyświetla w `<pre>` (pamiętaj o `htmlspecialchars`). Otwórz ją przez `http://localhost/sklep_oop/sklep.php`. Spakuj projekt jako `nazwisko_imie_zad15.zip` — będzie potrzebny w zadaniu 16.

W sprawozdaniu umieść

- ☐ tabela wykonanych poleceń (krok.punkt) z najważniejszymi fragmentami kodu
- ☐ komunikat przy próbie odczytu prywatnej właściwości
- ☐ zrzut terminala z wynikiem testów i zrzut strony `sklep.php`
- ☐ diagram klas (prostokąty z nazwami klas i metod; strzałka „implementuje” od klas dostawy do interfejsu) — może być odręczny lub w draw.io
- ☐ archiwum `nazwisko_imie_zad15.zip` i odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Co daje hermetyzacja? Podaj przykład błędu, przed którym chroni prywatna `cenaNetto` z walidacją.
2. Dlaczego `Zamowienie` przyjmuje w konstruktorze typ `SposobDostawy`, a nie `Kurier`? Co trzeba zmienić w `Zamowienie`, gdy dochodzi nowy sposób dostawy?
3. Czym różni się właściwość statyczna `$licznik` od zwykłej właściwości obiektu?
4. Po co przestrzenie nazw i jak nazwa klasy `Sklep\Dostawa\Kurier` odpowiada ścieżce pliku?

Kryteria oceny

Kryterium	Punkty
Produkt: promocja właściwości, walidacja z wyjątkami, cena brutto (6 testów OK)	4
Koszyk: łączenie tych samych produktów, zmiana ilości, wartość (6 testów OK)	4
Interfejs i 3 implementacje + własna klasa <code>PocztaPolska</code> z testem	3
Zamowienie: walidacja, numeracja statyczna, readonly, podsumowanie (wszystkie testy OK)	4
Strona <code>sklep.php</code> , diagram klas, zrzuty, archiwum	3
Odpowiedzi na pytania kontrolne	2
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).