

ЛАБОРАТОРНА РОБОТА № 3

Основи об'єктно-орієнтованого програмування: сутності, атрибути, CRUD-операції в пам'яті

Мета роботи: Закріпити базові принципи об'єктно-орієнтованого програмування в PHP (класи, об'єкти, інкапсуляція, конструктори) і навчитися реалізовувати CRUD-операції (створення, читання, оновлення, видалення) над колекцією об'єктів, що зберігається виключно в оперативній пам'яті програми — без використання файлів, бази даних чи інших зовнішніх сховищ.

Теоретичні відомості

1. Класи, об'єкти та інкапсуляція

Клас (тема 4) — це шаблон, що описує майбутні об'єкти: які *властивості* (дані) вони матимуть і які *методи* (дії) над цими даними будуть доступні. *Об'єкт* — конкретний екземпляр класу, створений оператором `new`. У цій лабораторній роботі всі об'єкти створюються **безпосередньо в коді програми** — без зчитування з файлу, форми чи бази даних.

Інкапсуляція — принцип, за яким внутрішні дані об'єкта (властивості) приховані від прямого доступу ззовні (модифікатор `private`) і доступні лише через методи самого класу (гетери, сетери, інші публічні методи). Це захищає об'єкт від встановлення некоректного стану ззовні та дозволяє змінювати внутрішню реалізацію класу, не торкаючись коду, що ним користується.

2. Конструктор

Конструктор — спеціальний метод `__construct()`, що викликається автоматично під час створення об'єкта через `new`. Він гарантує, що об'єкт від самого моменту створення перебуває в коректному, повністю ініціалізованому стані — на відміну від послідовного встановлення властивостей окремими викликами після створення об'єкта, коли між ними об'єкт тимчасово перебуває в неповному стані.

```
<?php
class Category {
    public function __construct(
        private int $id,
        private string $name
    ) {}

    public function getId(): int { return $this->id; }
    public function getName(): string { return $this->name; }
}

$category = new Category(1, 'Туристичні'); // об'єкт одразу готовий до використання
```

Синтаксис оголошення параметрів конструктора з модифікатором доступу (`private int $id`) називається *promoted properties* (PHP 8+) — він одночасно оголошує властивість класу й параметр конструктора, замінюючи довгий запис із явним `$this->id = $id;` в тілі методу.

3. CRUD-операції над колекцією об'єктів у пам'яті

CRUD (Create, Read, Update, Delete) — чотири базові операції з даними, розглянуті раніше (тема 7) на прикладі бази даних. У цій лабораторній роботі ті самі операції реалізуються **без жодної бази даних**: колекцією об'єктів керує клас-репозиторій, що зберігає їх у звичайному масиві PHP, проіндексованому за ідентифікатором.

Операція CRUD	Реалізація в пам'яті (масив об'єктів)
Create	Створити новий об'єкт (<code>new</code>) і додати його в масив під унікальним ключем-ідентифікатором
Read	Повернути весь масив (усі записи) або один об'єкт за ключем-ідентифікатором
Update	Знайти об'єкт за ідентифікатором і викликати його сетер(и) для зміни властивостей
Delete	Видалити елемент масиву за ключем-ідентифікатором (<code>unset</code>)

Головна відмінність від теми 7: там CRUD-методи виконували SQL-запити (`INSERT`, `SELECT`, `UPDATE`, `DELETE`) до бази даних, а дані зберігалися постійно на диску. Тут аналогічні за назвою методи лише змінюють масив у оперативній пам'яті — після завершення роботи скрипта всі дані зникають. Це нормально й очікувано для цієї лабораторної роботи: мета — відпрацювати сам об'єктно-орієнтований підхід, а не постійне зберігання.

4. Зв'язки між об'єктами

Реальні предметні області рідко складаються з однієї ізольованої сутності. Один об'єкт часто *посилається* на інший (наприклад, товар належить до певної категорії) або *містить* список інших об'єктів (наприклад, товар має декілька характеристик). В ООП це виражається зовсім просто: властивість одного класу оголошується як тип іншого класу (чи масив об'єктів цього класу) — жодних зовнішніх ключів чи JOIN-ів, як у базі даних, не потрібно.

```

<?php
class Product {
    private array $tags = [];    // список пов'язаних об'єктів

    public function __construct(
        private int $id,
        private string $name,
        private Category $category    // посилання на інший об'єкт
    ) {}

    public function addTag(Tag $tag): void {
        $this->tags[] = $tag;
    }
}

```

Завдання на лабораторну роботу

За основу візьміть **власний варіант завдання курсової роботи** з дисципліни «Організація баз даних» — предметну область, для якої ви вже проєктували структуру бази даних (наприклад, облік товарів, бібліотека, готель, автопарк, кінотеатр тощо). У цій лабораторній роботі та сама предметна область реалізується суто засобами ООП, без бази даних.

1. Визначте основні сутності, з якими буде здійснюватися робота у вашій програмі. Оберіть щонайменше дві пов'язані сутності зі свого варіанта курсової роботи (наприклад, «Товар» і «Категорія», «Книга» і «Автор», «Автомобіль» і «Марка») — одна сутність повинна посилатися на іншу або містити список інших об'єктів (розділ 4 теоретичних відомостей).
2. Визначте атрибути кожної сутності (властивості класу) та методи, які будуть з нею працювати. Обов'язково передбачте: конструктор, що ініціалізує обов'язкові властивості; гетери для читання властивостей; сетери чи інші методи для зміни стану об'єкта (Update); метод (чи спеціальний рядковий метод) для зручного виведення інформації про об'єкт.
3. Реалізуйте код програми з використанням об'єктно-орієнтованого підходу. Створіть клас-репозиторій (менеджер), що керує колекцією об'єктів головної сутності в масиві та реалізує повний набір CRUD-методів: create() — додавання нового об'єкта; findAll() — отримання всіх об'єктів; findById() — пошук одного об'єкта за ідентифікатором; update() — зміна властивостей наявного об'єкта; delete() — видалення об'єкта з колекції.

Додаткові вимоги

- Жодні дані не зчитуються з файлів, форм, консолі чи бази даних — усі об'єкти-приклади створюються оператором new безпосередньо в коді програми (у демонстраційній частині скрипта).
- Створіть не менше 3-4 об'єктів головної сутності та не менше 2-3 об'єктів пов'язаної сутності.
- Продемонструйте роботу кожного CRUD-методу окремо, виводячи на екран результат до і після операції (особливо наочно це для update() і delete()).

- Усі властивості класів мають бути `private` (інкапсульовані); доступ — лише через методи.

Вимоги до звіту

Звіт про лабораторну роботу повинен містити: тему й мету роботи; стислий опис обраної предметної області (з посиланням на відповідний варіант курсової роботи з «Організації баз даних»); перелік сутностей з їхніми атрибутами та методами (можна у вигляді таблиці чи схеми, як у прикладі нижче); повний лістинг коду програми; знімок екрана з результатом виконання програми (демонстрація всіх CRUD-операцій); висновок.

Приклад виконання: «Спортивні рюкзаки»

Розглянемо повний приклад для предметної області «спортивні рюкзаки, їхні категорії та характеристики». Приклад демонструє весь очікуваний обсяг роботи — від визначення сутностей до готового програмного коду — і може слугувати орієнтиром щодо структури та рівня деталізації власного варіанта.

1. Визначення сутностей

Для цієї предметної області визначено три сутності: **Категорія** (тип рюкзака: туристичний, спортивний, тактичний), **Характеристика** (окрема пара «назва — значення», наприклад «Матеріал — Кордура») та **Рюкзак** — головна сутність, що посилається на одну категорію та містить список характеристик.

Сутність	Зв'язок з іншими сутностями
Категорія	Незалежна сутність; на неї посилається Рюкзак
Характеристика	Незалежна сутність; список таких об'єктів міститься в Рюкзаку
Рюкзак	Головна сутність: посилається на одну Категорію, містить масив Характеристик

2. Атрибути та методи

Клас	Атрибути / Методи
Category	id, name / <code>__construct()</code> , <code>getId()</code> , <code>getName()</code>
Characteristic	name, value / <code>__construct()</code> , <code>getName()</code> , <code>getValue()</code>
Backpack	id, name, category, price, volumeLiters, characteristics[] / <code>__construct()</code> , <code>getери</code> , <code>setName()</code> , <code>setCategory()</code> , <code>setPrice()</code> , <code>setVolumeLiters()</code> , <code>addCharacteristic()</code>
BackpackRepository	backpacks[], nextId / <code>create()</code> , <code>findAll()</code> , <code>findById()</code> , <code>update()</code> , <code>delete()</code>

BackpackRepository — це не сутність предметної області, а допоміжний клас-менеджер (патерн Repository, знайомий з теми 7), відповідальний виключно за керування колекцією об'єктів Backpack: створення, пошук, оновлення й видалення. Такий поділ (сутність окремо, керування колекцією окремо) робить код зрозумілішим і повторно використовуваним.

3. Код класів сутностей

```
<?php

class Category {
    public function __construct(
        private int $id,
        private string $name
    ) {}

    public function getId(): int { return $this->id; }
    public function getName(): string { return $this->name; }

    public function __toString(): string {
        return $this->name;
    }
}

class Characteristic {
    public function __construct(
        private string $name,
        private string $value
    ) {}

    public function getName(): string { return $this->name; }
    public function getValue(): string { return $this->value; }

    public function __toString(): string {
        return "{$this->name}: {$this->value}";
    }
}
```

```

<?php

class Backpack {
    private array $characteristics = [];

    public function __construct(
        private int $id,
        private string $name,
        private Category $category,
        private float $price,
        private float $volumeLiters
    ) {}

    // --- Гетери (Read) ---
    public function getId(): int { return $this->id; }
    public function getName(): string { return $this->name; }
    public function getCategory(): Category { return $this->category; }
    public function getPrice(): float { return $this->price; }
    public function getVolumeLiters(): float { return $this-
>volumeLiters; }
    public function getCharacteristics(): array { return $this-
>characteristics; }

    // --- Сетери (Update) ---
    public function setName(string $name): void { $this->name =
$name; }
    public function setCategory(Category $category): void { $this-
>category = $category; }
    public function setPrice(float $price): void { $this->price =
$price; }
    public function setVolumeLiters(float $v): void { $this-
>volumeLiters = $v; }

    public function addCharacteristic(Characteristic $characteristic):
void {
        $this->characteristics[] = $characteristic;
    }

    public function __toString(): string {
        $chars = implode(' ', array_map(
            fn(Characteristic $c) => (string) $c,
            $this->characteristics
        ));
        return sprintf(
            '%d %s [%s] - %.1f л, %.2f грн (%s)',
            $this->id, $this->name, $this->category,
            $this->volumeLiters, $this->price, $chars
        );
    }
}

```

4. Клас-репозиторій із CRUD-методами

```

<?php

class BackpackRepository {
    private array $backpacks = [];
    private int $nextId = 1;

    // Create
    public function create(
        string $name, Category $category, float $price, float
        $volumeLiters
    ): Backpack {
        $backpack = new Backpack($this->nextId++, $name, $category,
        $price, $volumeLiters);
        $this->backpacks[$backpack->getId()] = $backpack;
        return $backpack;
    }

    // Read (усі записи)
    public function findAll(): array {
        return $this->backpacks;
    }

    // Read (один запис)
    public function findById(int $id): ?Backpack {
        return $this->backpacks[$id] ?? null;
    }

    // Update
    public function update(int $id, array $data): bool {
        $backpack = $this->findById($id);
        if ($backpack === null) {
            return false;
        }
        if (isset($data['name'])) $backpack->setName($data['name']);
        if (isset($data['price'])) $backpack->setPrice($data['price']);
        if (isset($data['volumeLiters'])) $backpack-
        >setVolumeLiters($data['volumeLiters']);
        return true;
    }

    // Delete
    public function delete(int $id): bool {
        if (!isset($this->backpacks[$id])) {
            return false;
        }
        unset($this->backpacks[$id]);
        return true;
    }
}

```

Метод create() сам відповідає за генерацію унікального ідентифікатора (nextId++) — так само, як AUTO_INCREMENT робив би це в базі даних (тема 7). Методи update() і delete() повертають bool, що дозволяє коду, який їх викликає, дізнатися про успіх операції (наприклад, чи справді існував об'єкт із заданим id) без потреби додатково перевіряти масив.

5. Створення екземплярів і демонстрація роботи (без файлів)

Демонстраційна частина скрипта створює всі об'єкти **безпосередньо в коді** (жодного файлу, форми чи бази даних) і послідовно викликає кожен CRUD-метод, виводячи результат на екран.

```
<?php
// --- Категорії (створені напряму в коді) ---
$hiking = new Category(1, 'Туристичні');
$sport = new Category(2, 'Спортивні');
$tactical = new Category(3, 'Тактичні');

$repository = new BackpackRepository();

// --- Create: додавання рюкзаків ---
$b1 = $repository->create('TrailMaster 40', $hiking, 2400.00, 40);
$b1->addCharacteristic(new Characteristic('Матеріал', 'Нейлон Cordura'));
$b1->addCharacteristic(new Characteristic('Вага', '1.2 кг'));

$b2 = $repository->create('RunLight 15', $sport, 1200.00, 15);
$b2->addCharacteristic(new Characteristic('Матеріал', 'Поліестер'));
$b2->addCharacteristic(new Characteristic('Вага', '0.6 кг'));

$b3 = $repository->create('TacticalPro 30', $tactical, 3200.00, 30);
$b3->addCharacteristic(new Characteristic('Матеріал', 'Кордура 1000D'));
$b3->addCharacteristic(new Characteristic('MOLLE-система', 'Так'));

// --- Read: усі рюкзаки ---
echo "=== Усі рюкзаки ===\n";
foreach ($repository->findAll() as $backpack) {
    echo $backpack . "\n";
}

// --- Read: один рюкзак за id ---
echo "\n=== Пошук за id=2 ===\n";
echo $repository->findById(2) . "\n";

// --- Update: зміна ціни ---
echo "\n=== Оновлення ціни рюкзака #1 ===\n";
$repository->update(1, ['price' => 2200.00]);
echo $repository->findById(1) . "\n";

// --- Delete: видалення рюкзака ---
echo "\n=== Видалення рюкзака #3 ===\n";
$repository->delete(3);
foreach ($repository->findAll() as $backpack) {
    echo $backpack . "\n";
}
```

Очікуваний результат виконання

```
=== Усі рюкзаки ===
#1 TrailMaster 40 [Туристичні] – 40.0 л, 2400.00 грн (Матеріал: Нейлон Cordura, Вага: 1.2 кг)
#2 RunLight 15 [Спортивні] – 15.0 л, 1200.00 грн (Матеріал: Поліестер, Вага: 0.6 кг)
#3 TacticalPro 30 [Тактичні] – 30.0 л, 3200.00 грн (Матеріал: Кордура 1000D, MOLLE-система: Так)

=== Пошук за id=2 ===
#2 RunLight 15 [Спортивні] – 15.0 л, 1200.00 грн (Матеріал: Поліестер, Вага: 0.6 кг)

=== Оновлення ціни рюкзака #1 ===
#1 TrailMaster 40 [Туристичні] – 40.0 л, 2200.00 грн (Матеріал: Нейлон Cordura, Вага: 1.2 кг)

=== Видалення рюкзака #3 ===
#1 TrailMaster 40 [Туристичні] – 40.0 л, 2200.00 грн (Матеріал: Нейлон Cordura, Вага: 1.2 кг)
#2 RunLight 15 [Спортивні] – 15.0 л, 1200.00 грн (Матеріал: Поліестер, Вага: 0.6 кг)
```

Контрольні запитання

- Чим інкапсуляція відрізняється від простого приховування властивостей через private — навіщо потрібні саме публічні методи доступу?
- Навіщо потрібен конструктор, якщо властивості можна встановити окремими викликами сетерів одразу після створення об'єкта?
- Чому масив у класі-репозиторії зручніше індексувати за ідентифікатором об'єкта, а не використовувати простий список без ключів?
- Чим методи create()/update()/delete() у цій роботі (в оперативній пам'яті) відрізняються за призначенням від однойменних SQL-операцій із теми 7 — і чим схожі?
- Чому методи findById() і delete() повертають, відповідно, null і false у разі відсутності об'єкта, а не викликають помилку?
- Як саме один об'єкт (наприклад, Рюкзак) посилається на інший (Категорія) в ООП-підході — без зовнішніх ключів і JOIN-ів?
- Що станеться зі створеними об'єктами після завершення роботи скрипта і чому це очікувано для цієї лабораторної роботи?
- Яку схожість і яку принципову відмінність ви бачите між власною предметною областю (курсова з «Організації баз даних») і поданим прикладом рюкзаків?