МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Програмної інженерії

Звіт
з лабораторної роботи №2
з дисципліни: «Скриптові мови програмування»
з теми: «Продовольчий магазин «Весна»»

Виконав:
ст. гр. ПЗПІ-23-2
Ситник Є. С.

Перевірив:
Старший викладач кафедри ПІ
Сокорчук І. П.

Харків – 2025

2 ПРОДОВОЛЬЧИЙ МАГАЗИН «ВЕСНА»

2.1 Історія змін

№	Дата	Версія звіту	Опис змін та виправлень
1	03.03.2025	0.1	Створено звіт

2.2 Мета роботи

Лабораторна робота полягає у розробці консольного застосунку «Продовольчий магазин «Весна»» засобами мови програмування «PHP».

2.3 Хід роботи

2.3.1 Архітектура програми

Для реалізації консольного застосунку було обрано об’єктно-орієнтований підхід з використанням наступних компонентів:

2.3.1.1 Класи та їх призначення:

- **DB** – клас для роботи з базою даних «SQLite», що забезпечує збереження даних про товари, кошик користувача та налаштування профілю;
- **App** – основний клас застосунку, що реалізує логіку роботи програми та взаємодію з користувачем;
- **DbException** та **AppException** – класи винятків для обробки помилок.

2.3.1.2 Enum State:

Для управління станами програми використовується перелічення «State» з наступними значеннями:

- **Hello** – привітальний екран;
- **Menu** – головне меню;
- **Items** – вибір товарів;
- **Checkout** – формування рахунку;
- **Settings** – налаштування профілю;
- **Exit** – вихід з програми.

2.3.2 Структура бази даних

Програма використовує базу даних SQLite з трьома таблицями:

- **settings** – зберігає інформацію про користувача (ім’я та вік);
- **items** – містить каталог товарів з їх назвами та цінами;
- **cart** – зберігає товари, додані до кошика, з їх кількістю.

2.3.3 Основний функціонал

2.3.3.1 Головне меню:

Програма відображає заголовок магазину та пропонує користувачу чотири основні опції:

- вибір товарів для покупки;
- перегляд підсумкового рахунку;
- налаштування профілю користувача;
- вихід з програми.

2.3.3.2 Вибір товарів:

При виборі першого пункту меню користувач потрапляє в режим покупки товарів, де:

- відображається список доступних товарів з цінами;
- можна вибрати товар за номером та вказати кількість;
- товари додаються до кошика;
- при введенні кількості «0» товар видаляється з кошика;
- реалізована валідація введених даних.

2.3.3.3 Підсумковий рахунок:

Другий пункт меню формує детальний рахунок з інформацією про:

- номер, назву та ціну кожного товару;
- кількість товару в кошику;
- загальну вартість кожного товару;
- підсумкову суму до сплати.

2.3.3.4 Налаштування профілю:

Третій пункт дозволяє користувачу вказати своє ім’я та вік з валідацією:

- ім’я повинно містити хоча б одну літеру;
- вік повинен бути в діапазоні від 7 до 150 років.

2.3.4 Технічні деталі

Програма написана з дотриманням сучасних стандартів «PHP»:

- використання строгої типізації;
- документування методів за допомогою «PHPDoc»;
- дотримання принципів «SOLID»;
- розділення логіки на окремі методи для кращої читабельності.

2.4 Висновки

Під час даної лабораторної роботи я вивчив основні принципи роботи з «PHP» для написання консольних програм. Зокрема, було освоєно:

- створення об’єктно-орієнтованих консольних застосунків на «PHP»;
- роботу з базою даних «SQLite» через «PDO»;
- обробку користувацького вводу та валідацію даних;
- використання перелічень («enum») для управління станами програми;
- реалізацію системи обробки винятків;
- форматування виводу в консолі для створення зручного інтерфейсу.

Програмне забезпечення, реалізоване протягом даної лабораторної роботи, повністю відповідає поставленим вимогам та забезпечує всі необхідні функції для роботи з продовольчим магазином. Застосунок має зручний інтерфейс, надійну систему валідації даних та ефективно працює з базою даних для збереження інформації про товари та користувача.

ДОДАТОК А

Відеозапис

Відеозапис презентації результатів лабораторної роботи: https://youtu.be/CMGeL1OdyvI

Хронологічний опис відеозапису:

00:00 – Вступ та опис завдання

00:19 – Архітектура програми

00:39 – Робота з базою даних

01:43 – Реалізація станів програми

02:17 – Формування списку товарів

03:13 – Додавання товарів до кошика

04:15 – Робота з кошиком

05:10 – Налаштування профілю користувача

05:38 – Демонстрація роботи програми

ДОДАТОК Б

Програмний код

Б.1 Скрипт з реалізацією програми

GitHub репозиторій: https://github.com/NureSytnykYehor/smp-pzpi-23-2-sytnyk-yehor/blob/main/Lab2/smp-pzpi-23-2-sytnyk-yehor-lab2/smp-pzpi-23-2-sytnyk-yehor-lab2-code

```
#!/usr/bin/php

<?php

class DbException extends Exception {}
class AppException extends Exception {}

enum State
{
    case Hello;
    case Menu;
    case Items;
    case Checkout;
    case Settins;
    case Exit;
}

class DB
{
    private $pdo;

    /**
     * Initializes database
     *
     * @param string $db_path
     * @throws DbException If there's a database error.
     */
    public function __construct($db_path)
    {
        try {
            $this->pdo = new PDO("sqlite:" . $db_path);
            $this->pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            throw new DbException("Connection to DB failed.\nCaused
by: " . $e->getMessage());
        }

        try {
            $this->pdo->exec("
                CREATE TABLE IF NOT EXISTS settings (
                    name TEXT,
                    age TEXT
```

```
                );
            ");

            if ($this->pdo->query("SELECT COUNT(*) FROM settings;")->fetchColumn() == 0) {
                $this->pdo->exec("INSERT INTO settings (name, age) VALUES ('user', 0);");
            }
        } catch (PDOException $e) {
            throw new DbException("Error initialising settings table.\nCaused by: " . $e->getMessage());
        }

        try {
            $this->pdo->exec("
                CREATE TABLE IF NOT EXISTS items (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    name TEXT NOT NULL,
                    price REAL NOT NULL
                );
            ");

            if ($this->pdo->query("SELECT COUNT(id) FROM items;")->fetchColumn() == 0) {
                $this->pdo->exec("
                    INSERT INTO items (name, price) VALUES ('Молоко пастеризоване', 12);
                    INSERT INTO items (name, price) VALUES ('Хліб чорний', 9);
                    INSERT INTO items (name, price) VALUES ('Сир білий', 21);
                    INSERT INTO items (name, price) VALUES ('Сметана 20%', 25);
                    INSERT INTO items (name, price) VALUES ('Кефір 1%', 19);
                    INSERT INTO items (name, price) VALUES ('Вода газована', 18);
                    INSERT INTO items (name, price) VALUES ('Печиво \"Весна\"', 14);
                ");
            }
        } catch (PDOException $e) {
            throw new DbException("Error initialising items table.\nCaused by: " . $e->getMessage());
        }

        try {
            $this->pdo->exec("
                CREATE TABLE IF NOT EXISTS cart (
                    id INTEGER NOT NULL UNIQUE,
                    count INTEGER NOT NULL,
                    FOREIGN KEY(id) REFERENCES item(id)
                );
            ");
        } catch (PDOException $e) {
```

```
85              throw new DbException("Error initialising cart table.
    \nCaused by: " . $e->getMessage());
86          }
87      }
88
89      /**
90       * Updates user information in the database
91       *
92       * @param string $name
93       * @param int $age
94       * @return void
95       * @throws DbException If there's a database error.
96       */
97      public function update_user($name, $age): void
98      {
99          try {
100             $stmt = $this->pdo->prepare("UPDATE settings SET name
    = :name, age = :age;");
101
102             $stmt->bindParam(':name', $name, PDO::PARAM_STR);
103             $stmt->bindParam(':age', $age, PDO::PARAM_INT);
104
105             $stmt->execute();
106         } catch (PDOException $e) {
107             throw new DbException("Error updating user info.\nCaused
    by: " . $e->getMessage());
108         }
109     }
110
111     /**
112      * Fetches all items from the database.
113      *
114      * @return array[]
115      * @throws DbException If there's a database error.
116      */
117     public function get_items(): array
118     {
119         try {
120             $stmt = $this->pdo->query("SELECT id, name, price FROM
    items ORDER BY id;");
121             return $stmt->fetchAll(PDO::FETCH_ASSOC);
122         } catch (PDOException $e) {
123             throw new DbException("Error retrieving data from the
    items table.\nCaused by: " . $e->getMessage());
124         }
125     }
126
127     /**
128      * Fetches all items in the cart from the database without price
    info.
129      *
130      * @return array[]
131      * @throws DbException If there's a database error.
132      */
133     public function get_cart_no_price(): array
134     {
```

```
135         try {
136             $stmt = $this->pdo->query(
137                 "SELECT name, count FROM cart
138                 INNER JOIN items ON cart.id = items.id
139                 ORDER BY cart.id;"
140             );
141             return $stmt->fetchAll(PDO::FETCH_ASSOC);
142         } catch (PDOException $e) {
143             throw new DbException("Error inserting init data in the
    items table.\nCaused by: " . $e->getMessage());
144         }
145     }
146 
147     /**
148      * Fetches all items in the cart from the database.
149      *
150      * @return array[]
151      * @throws DbException If there's a database error.
152      */
153     public function get_cart(): array
154     {
155         try {
156             $stmt = $this->pdo->query(
157                 "SELECT
158                     cart.id, name, price, count, price*count as
    total_price
159                 FROM cart
160                 INNER JOIN items ON cart.id = items.id
161                 ORDER BY cart.id;"
162             );
163             return $stmt->fetchAll(PDO::FETCH_ASSOC);
164         } catch (PDOException $e) {
165             throw new DbException("Error inserting init data in the
    items table.\nCaused by: " . $e->getMessage());
166         }
167     }
168 
169     /**
170      * Add item to the cart
171      *
172      * @param int $id
173      * @param int $count
174      *
175      * @return bool
176      * @throws DbException If there's a database error.
177      */
178     public function add_to_cart($id, $count): bool
179     {
180         try {
181             $stmt = $this->pdo->prepare(
182                 "INSERT INTO cart
183                     (id, count)
184                 VALUES
185                     (:id, :count)
186                 ON CONFLICT(id)
187                     DO UPDATE SET
```

```
                        count = :count
                    WHERE id = :id;"
                );
                return $stmt->execute(['id' => $id, 'count' => $count]);
            } catch (PDOException $e) {
                throw new DbException("Error adding item to the cart.
\nCaused by: " . $e->getMessage());
            }
        }

        /**
         * Remove an item from the cart
         *
         * @param int $id
         *
         * @return bool
         * @throws DbException If there's a database error.
         */
        public function remove_from_cart($id): bool
        {
            try {
                $stmt = $this->pdo->prepare("DELETE FROM cart WHERE id
= :id");
                return $stmt->execute(['id' => $id]);
            } catch (PDOException $e) {
                throw new DbException("Error removimg item from the cart.
\nCaused by: " . $e->getMessage());
            }
        }
    }

    class App
    {
        private $db;
        private $state;

        private $menu_ops = <<<'END'
        1 Вибрати товари
        2 Отримати підсумковий рахунок
        3 Налаштувати свій профіль
        0 Вийти з програми
        END;
        private $hello = <<<'END'
        ################################
        # ПРОДОВОЛЬЧИЙ МАГАЗИН "ВЕСНА" #
        ################################
        END;


        /**
         * @param string $db_path
         */
        public function __construct($db_path)
        {
            try {
                $this->db = new DB($db_path);
```

```
241             $this->state = State::Hello;
242         } catch (DbException $e) {
243             throw new AppException("Error initializing app.\nCaused
    by: " . $e);
244         }
245     }
246 
247     public function poll(): void
248     {
249         while ($this->state != State::Exit) {
250             switch ($this->state) {
251                 case State::Hello:
252                     $this->hello();
253                     break;
254                 case State::Menu:
255                     $this->menu();
256                     break;
257                 case State::Items:
258                     $this->items();
259                     break;
260                 case State::Checkout:
261                     $this->checkout();
262                     break;
263                 case State::Settins:
264                     $this->settings();
265                     break;
266 
267                 default:
268                     break;
269             }
270         }
271     }
272 
273     private function menu(): void
274     {
275         echo "\n";
276         echo "$this->menu_ops\n";
277 
278         $op = readline('Введіть команду: ');
279         switch ($op) {
280             case '1':
281                 $this->state = State::Items;
282                 break;
283             case '2':
284                 $this->state = State::Checkout;
285                 break;
286             case '3':
287                 $this->state = State::Settins;
288                 break;
289             case '0':
290                 $this->state = State::Exit;
291                 break;
292 
293             default:
294                 echo "ПОМИЛКА! Введіть правильну команду\n";
295                 break;
```

```
296         }
297 
298         echo "\n";
299     }
300     private function hello(): void
301     {
302         echo "$this->hello\n";
303         $this->state = State::Menu;
304     }
305     private function items(): void
306     {
307         $items = $this->db->get_items();
308         array_unshift($items, ['id' => "№", 'name' => "НАЗВА", 'price'
    => "ЦІНА"]);
309         array_push($items, ['id' => " ", 'name' => "-----------",
    'price' => ""]);
310         array_push($items, ['id' => "0", 'name' => "ПОВЕРНУТИСЯ",
    'price' => ""]);
311         $columns = $this->count_columns($items);
312 
313         while (true) {
314             $this->print_lits($items, $columns);
315 
316             $id = readline("Виберіть товар: ");
317 
318             if ($id == '0') {
319                 break;
320             }
321 
322             echo "\n";
323 
324             $selected = null;
325             foreach ($items as $item) {
326                 if ($item['id'] === (int)$id)
327                     $selected = $item;
328             }
329 
330             if ($selected == null) {
331                 echo "ПОМИЛКА! ВКАЗАНО НЕПРАВИЛЬНИЙ НОМЕР ТОВАРУ\n";
332                 continue;
333             }
334 
335             echo "Вибрано: {$selected['name']}\n";
336 
337             $count = (int)readline("Введіть кількість, штук: ");
338 
339             if ($count > 100) {
340                 echo "ПОМИЛКА! Не можна додати більше 100 одиниць
    товару в кошик\n";
341                 continue;
342             }
343 
344             if ($count < 0) {
345                 echo "ПОМИЛКА! Кількість не може бути від'ємною\n";
346                 continue;
347             }
```

```
348
349             if ($count == 0) {
350                 echo "ВИДАЛЯЮ З КОШИКА\n";
351                 $this->db->remove_from_cart($id);
352             } else {
353                 $this->db->add_to_cart($id, $count);
354             }
355
356             $cart = $this->db->get_cart_no_price();
357             if (count($cart) == 0) {
358                 echo "КОШИК ПОРОЖНІЙ\n";
359             } else {
360                 echo "\nУ КОШИКУ:\n";
361                 array_unshift($cart, ['name' => "НАЗВА", 'count' =>
    "КІЛЬКІСТЬ"]);
362                 $cart_columns = $this->count_columns($cart);
363                 $this->print_lits($cart, $cart_columns);
364                 echo "\n";
365             }
366         }
367
368         $this->state = State::Menu;
369     }
370     private function checkout(): void
371     {
372         $cart = $this->db->get_cart();
373         if (count($cart) == 0) {
374             echo "КОШИК ПОРОЖНІЙ\n";
375             $this->state = State::Menu;
376             return;
377         } else {
378             echo "У КОШИКУ:\n";
379             array_unshift($cart, ['id' => "№", 'name' => "НАЗВА",
    'price' => "ЦІНА", 'count' => "КІЛЬКІСТЬ", 'total_price' =>
    "ВАРТІСТЬ"]);
380             $cart_columns = $this->count_columns($cart);
381             $this->print_lits($cart, $cart_columns);
382         }
383
384         $total_price = array_reduce($cart, function ($carry, $item) {
385             return $carry + (int)$item['total_price'];
386         }, 0);
387         echo "РАЗОМ ДО СПЛАТИ: {$total_price}\n";
388
389         $this->state = State::Menu;
390     }
391     private function settings(): void
392     {
393         while (true) {
394             $name = readline("Ваше ім'я: ");
395             if ($name !== "" && preg_match("/[a-zA-Z]+/", $name))
396                 break;
397         }
398
399         while (true) {
400             $age = readline("Ваш вік: ");
```

```
401
402             if (!filter_var($age, FILTER_VALIDATE_INT)) {
403                 echo "ПОМИЛКА! Вік користувача потрібно вказати
    числом\n\n";
404                 continue;
405             }
406
407             if ($age < 7 || $age > 150) {
408                 echo "ПОМИЛКА! Користувач повинен мати вік від 7 та до
    150 років\n\n";
409                 continue;
410             }
411
412             break;
413         }
414
415         echo "\n";
416
417         $this->db->update_user($name, $age);
418
419         $this->state = State::Menu;
420     }
421
422     /**
423      * @param array<array> $items
424      * @return array<int>
425      */
426     private function count_columns($items): array
427     {
428         $columns = [];
429         foreach ($items as $item) {
430             foreach ($item as $field => $value) {
431                 if (!key_exists($field, $columns))
432                     $columns[$field] = mb_strlen($value);
433                 else
434                     $columns[$field] = max(mb_strlen($value),
    $columns[$field]);
435             }
436         }
437
438         return $columns;
439     }
440     /**
441      * @param array $element
442      * @param array<int> $columns
443      */
444     private function pad_row($element, $columns): string
445     {
446         $result = [];
447         foreach ($element as $field => $value)
448             $result[] = mb_str_pad($value, $columns[$field], ' ',
    STR_PAD_RIGHT);
449
450         return implode("  ", $result);
451     }
452     /**
```

```
453      * @param array<array> $items
454      * @param array<int> $columns
455      */
456     private function print_lits($items, $columns): void
457     {
458         foreach ($items as $item)
459             echo $this->pad_row($item, $columns) . "\n";
460     }
461 }
462 
463 try {
464     $app = new App("db.sqlite");
465 } catch (AppException $e) {
466     echo $e;
467     exit(1);
468 }
469 
470 $app->poll();
```