Міністерство освіти і науки України

Економіко-технологічний інститут ім. Роберта Ельворті

Кафедра інформаційних технологій

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни «Об’єктно-орієнтоване програмування»

«Бібліотека класів для розробки

родового (Генеалогічного) дерева»

Керівник:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Комісія:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницький 2024

Економіко-Технологічний інститут

ім. Роберта Ельворті

Кафедра *інформаційних технологій*

Рівень вищої освіти *перший (бакалаврський)*

Дисципліна *Об’єктно-орієнтоване програмування*

Спеціальність *122 Комп’ютерні науки*

Освітня програма: *Комп’ютерні науки*

Курс *4*

Група *КН-20*

Семестр *8*

ЗАВДАННЯ на курсовий проект студента

Васильєвої Мар’яни Вікторівни

1 Тема проекту: Спроектувати та реалізувати бібліотеку класів для розробки родового (Генеалогічного) дерева. Паспортні дані членів деякого родового клану; посилання на дітей (або на батьків). Пошук всіх нащадків або всіх предків для вказаної особи. Ієрархічне відображення генеалогічного дерева обраної людини.

2 Термін здачі студентом закінченого проекту: “16” квітня 2024 р.

3 Вихідні дані до проекту: *Специфікація програми, методичні*

*вказівки до виконання курсової роботи*

4 Зміст розрахунково-пояснювальної записки:

*Вступ, специфікація програми, проектна специфікація, інструкція*

*користувача, висновки\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Перелік графічного матеріалу:

*Схема об’єктної моделі, алгоритми, приклади екранних форм, знімки екрану*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 44 с., 10 рис., 3 табл., 2

додатки, 7 джерел.

ЛОКАЛЬНИЙ СЕРВЕР РОЗРОБКИ, MVC, SQL, PHP, СУБД SQLITE, COMPOSER, HTML, CSS, ІНКАПСУЛЯЦІЯ, КЛАС, ОБ'ЄКТ, МЕТОД, ВЛАСТИВІСТЬ, КОНСТРУКТОР, SINGLETON, FACTORY.

Метою роботи є розробка програми «Family Tree» на засадах об'єктно-орієнтованого програмування. Методи розробки базуються на використанні локального середовища розробки PHP, мови програмування PHP та принципів ООП.

В результаті отриманий веб-додаток під назвою «Family Tree», що дозволяє зберігати, опрацьовувати інформацію про осіб та їх взаємозв’язки в контексті створення генеалогічного дерева.

Реалізовано можливості користувацького інтерфейсу:

- додавати, редагувати, видаляти осіб в таблицю “People”;

- додавати, редагувати, видаляти інформаційні записи на головну сторінку.

Реалізовано можливості вебдодатку:

- формувати родове дерево для обраної особи;

- виводити загальну інформацію по всім особам у вигляді таблиці.

Реалізовано додаткові серверні можливості

- консольні команди для демонстрації роботи проєкту.

ЗМІСТ

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

ВСТУП

У сучасному світі інформаціних технологій, зокрема в веб-розробці (програмування веб додатків), особливо значущим стає вміння застосувати об’єктно-орієнтовані принципи для створення масштабованих і легко модифікованих програмних рішень. Дана курсова робота, виконана в рамках дисципліни "Об’єктно-орієнтоване програмування", має на меті не лише закріпити базові поняття і принципи об’єктно-орієнтованого програмування, а й демонструє їх практичне застосування через розробку спеціалізованої бібліотеки класів.

Головна задача даної курсової роботи — спроектувати та реалізувати бібліотеку класів для розробки генеалогічного (родового) дерева, здатну зберігати паспортні дані членів деякого родового клану, включно з посиланнями на дітей та батьків. Окрім того, бібліотека дозволить здійснювати пошук всіх нащадків або всіх предків для вказаної особи та ієрархічно відображати генеалогічне дерево обраної людини.

Реалізація проекту здійснена з використанням мови програмування PHP, шаблонізатора Blade та системи управління базами даних SQLite, що забезпечує гнучкість та зручність управління даними. Застосування цих технологій дозволяє ефективно інтегрувати об’єктно-орієнтовані підходи у веб-розробку, створюючи міцну основу для подальших інноваційних розробок у цій галузі.

Таким чином, дана курсова робота відображає не тільки теоретичне осмислення об’єктно-орієнтованих принципів, а й їх практичне застосування в реалізації конкретного програмного продукту, що має значний потенціал застосування у сфері родових досліджень та управління генеалогічною інформацією.

РОЗДІЛ 1. Специфікація програми

**Локальний сервер розробки** - це середовище для розробки вебдодатків на локальному комп'ютері. Локальний сервер дозволяє запускати вебсервер, базу даних, та інші сервіси для розробки та тестування вебдодатків. Локальний сервер дозволяє розробникам працювати над вебдодатками без доступу до Інтернету та забезпечує швидкий доступ до ресурсів.

**Модель-контролер-представлення** (MVC) - це архітектурний шаблон для розробки вебдодатків, який розділяє логіку програми на три компоненти: модель, контролер, та представлення. Модель відповідає за роботу з даними, контролер відповідає за обробку запитів, та представлення відповідає за відображення даних. MVC дозволяє розділити логіку програми на окремі компоненти та забезпечує гнучкість та розширюваність програми.

**SQL** - це мова запитів для роботи з базами даних, яка використовується для зберігання та обробки даних. SQL дозволяє створювати таблиці, вставляти, оновлювати, видаляти дані, та виконувати запити до бази даних. SQL підтримує різні типи даних, операції, та функції для роботи з даними.

**Таблиця** - це структура для зберігання даних у базі даних, яка складається з рядків та стовпців. Кожен рядок таблиці представляє запис з даними, а кожен стовпець представляє властивість запису. Таблиці можуть бути пов'язані між собою за допомогою зовнішніх ключів, що дозволяє створювати зв'язки між даними.

**Запит** - це команда для виконання операцій з базою даних, яка використовується для отримання, вставки, оновлення, видалення даних. Запити можуть бути виконані за допомогою SQL-команд, які виконують операції з базою даних. Запити можуть бути використані для отримання даних з бази даних, вставки нових даних, оновлення існуючих даних, та видалення даних.

**Генеалогічне дерево** - це ієрархічна структура, що відображає зв'язки між членами сім'ї. Кожна особа може мати батьків, дітей, та інших родичів. Генеалогічне дерево може бути представлене у вигляді графа, де вершини - це особи, а ребра - це зв'язки між ними. Генеалогічне дерево може бути використане для відстеження родоводу, пошуку нащадків/предків, та відображення зв'язків між членами сім'ї.

**Мова програмування PHP** - це мова програмування загального призначення, яка використовується для розробки вебдодатків. PHP підтримує об'єктноорієнтоване програмування, що дозволяє створювати класи та об'єкти для роботи з даними. PHP також підтримує роботу з базами даних, що дозволяє зберігати та обробляти інформацію.

**СУБД SQLite** - це легка вбудовувана база даних, яка підтримує стандарт SQL. SQLite зберігає дані у вигляді файлів на диску, що дозволяє легко переносити та резервувати дані. SQLite підтримує транзакції, індекси, та інші функції для роботи з даними.

**Composer** - це пакетний менеджер для PHP, який дозволяє встановлювати та оновлювати залежності для проєктів. Composer дозволяє встановлювати пакети з Packagist, а також власні пакети. Composer автоматизує процес встановлення та оновлення залежностей, що дозволяє зосередитися на розробці програми.

Пакет "**nikic/fast-route**" - це бібліотека для маршрутизації HTTP-запитів у PHP. FastRoute дозволяє визначати маршрути для обробки запитів, що дозволяє створювати вебдодатки з різними сторінками та функціоналом. FastRoute підтримує шаблони маршрутів, що дозволяє визначати динамічні маршрути для обробки запитів.

**Blade** - це шаблонізатор для PHP, який дозволяє використовувати шаблони для відображення даних. Blade дозволяє використовувати директиви, які замінюються на значення змінних, що дозволяє створювати шаблони для відображення даних. Blade підтримує вкладені шаблони, що дозволяє створювати складні структури для відображення даних.

**HTML** - це мова розмітки для створення вебсторінок та додатків. HTML використовує теги для визначення структури сторінки, що дозволяє відображати текст, зображення, та інші елементи. HTML підтримує вкладені теги, атрибути, та інші функції для створення вебсторінок.

**CSS** - це мова стилів для оформлення вебсторінок та додатків. CSS використовує правила для визначення стилів елементів, що дозволяє змінювати кольори, шрифти, розміри, та інші властивості. CSS підтримує класи, ідентифікатори, та інші селектори для вибору елементів для оформлення.

**Bootstrap** - це фреймворк для розробки вебсторінок та додатків. Bootstrap містить готові компоненти, які дозволяють швидко створювати стильні та адаптивні вебсторінки. Bootstrap підтримує сітку, кнопки, форми, та інші елементи для розробки вебсторінок.

**Google Icons** - це набір іконок для вебсторінок та додатків. Google Icons містить різноманітні іконки, які можна використовувати для відображення різних елементів на сторінці. Google Icons підтримує різні розміри та стилі іконок, що дозволяє вибрати необхідний варіант для вебсторінки.

**Інкапсуляція** - це принцип ООП, який дозволяє об'єднувати дані та методи в класах. Інкапсуляція дозволяє приховувати деталі реалізації та забезпечувати доступ до даних через методи. Це дозволяє зменшити залежність між класами та забезпечити безпеку даних.

**Наслідування** - це принцип ООП, який дозволяє створювати нові класи на основі наявних класів. Наслідування дозволяє використовувати властивості та методи батьківського класу у дочірньому класі. Це дозволяє зменшити дублювання коду та забезпечити повторне використання коду.

**Поліморфізм** - це принцип ООП, який дозволяє використовувати об'єкти різних класів через спільний інтерфейс. Поліморфізм дозволяє використовувати об'єкти різних класів у ролі аргументів методів та повертати різні типи об'єктів. Це дозволяє зменшити залежність від конкретних класів та забезпечити гнучкість програми.

**Клас** - це шаблон для створення об'єктів, який містить властивості та методи для роботи з даними. Клас може мати конструктор для ініціалізації об'єкта, властивості для зберігання даних, та методи для роботи з даними. Клас може бути використаний для створення об'єктів, які представляють реальні об'єкти або концепції.

**Об'єкт** - це екземпляр класу, який містить дані та методи для роботи з даними. Об'єкт може мати властивості для зберігання даних та методи для роботи з даними. Об'єкт може бути створений за допомогою конструктора класу та використаний для виконання операцій над даними.

**Метод** - це функція, яка визначена в класі та використовується для роботи з даними. Метод може приймати аргументи, виконувати операції над даними, та повертати результати. Метод може бути викликаний для об'єкта класу та використаний для виконання певних операцій.

**Властивість** - це змінна, яка визначена в класі та використовується для зберігання даних. Властивість може бути публічною, приватною, або захищеною, що визначає доступ до даних ззовні класу. Властивість може бути використана для зберігання даних та передачі їх між методами класу.

**Конструктор** - це спеціальний метод класу, який викликається при створенні нового об'єкта. Конструктор може приймати аргументи для ініціалізації об'єкта та виконувати певні операції. Конструктор може бути використаний для ініціалізації властивостей об'єкта та виконання певних дій при створенні об'єкта.

**Статичний метод** - це метод класу, який може бути викликаний без створення об'єкта. Статичний метод може бути використаний для виконання операцій, які не потребують доступу до властивостей об'єкта. Статичний метод може бути викликаний через ім'я класу та використаний для виконання певних операцій.

**Статична властивість** - це властивість класу, яка може бути викликана без створення об'єкта. Статична властивість може бути використана для зберігання даних, які не залежать від конкретного об'єкта. Статична властивість може бути викликана через ім'я класу та використана для зберігання даних.

**Абстрактний клас** - це клас, який містить абстрактні методи та може бути використаний для створення інших класів. Абстрактний клас може містити абстрактні методи, які повинні бути реалізовані в дочірніх класах. Абстрактний клас може бути використаний для визначення загальних властивостей та методів для дочірніх класів.

**Інтерфейс** - це контракт, який визначає методи, які повинні бути реалізовані в класі. Інтерфейс може містити декларації методів без їх реалізації, що дозволяє визначити структуру класу. Інтерфейс може бути використаний для визначення методів, які повинні бути реалізовані в класі, що дозволяє створювати класи з однаковим інтерфейсом.

**Трейт** - це механізм для повторного використання коду в PHP, який дозволяє додавати методи та властивості до класу. Трейт може містити методи та властивості, які можуть бути використані в класі. Трейт може бути використаний для додавання функціональності до класу без наслідування.

**Константа** - це змінна, яка має постійне значення та не може бути змінена під час виконання програми. Константа може бути визначена в класі та використана для зберігання значень, які не змінюються. Константа може бути використана для зберігання значень, які використовуються в різних частинах програми.

**Простори імен** - це механізм для організації класів та функцій в PHP, який дозволяє уникнути конфліктів імен. Простір імен може містити класи, функції, та константи, які можуть бути використані в програмі. Простір імен може бути використаний для організації коду та уникнення конфліктів імен.

**Автозавантаження класів** - це механізм для автоматичного завантаження класів в PHP, який дозволяє використовувати класи без ручного підключення файлів. Автозавантаження класів може бути використане для автоматичного завантаження класів з файлів за допомогою функцій автозавантаження. Автозавантаження класів дозволяє зосередитися на розробці програми та уникнути дублювання коду.

**Патерни проєктування** - це шаблони для розробки програмного забезпечення, які дозволяють розв'язувати типові проблеми та завдання. Патерни проєктування дозволяють створювати програми, які є гнучкими, розширюваними, та підтримуваними. Патерни проєктування дозволяють використовувати best practices та забезпечують високу якість програмного забезпечення.

**Singleton** - це патерн проєктування, який використовується для створення єдиного екземпляра класу. Singleton дозволяє створити тільки один об'єкт класу та забезпечує глобальний доступ до нього. Singleton може бути використаний для створення єдиного об'єкта, який може бути використаний в різних частинах програми.

**Factory** - це патерн проєктування, який використовується для створення об'єктів без прив'язки до конкретних класів. Factory дозволяє створювати об'єкти за допомогою фабричних методів та фабричних класів. Factory може бути використаний для створення об'єктів залежно від умов та параметрів.

РОЗДІЛ 2. Проєктна специфікація

РОЗДІЛ 3. Інструкція користувача

ВИСНОВКИ

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

Додаток А