**Міністерство освіти і науки України**

**Національний авіаційний університет**

**Навчально-науковий інститут комп’ютерних**

**інформаційних технологій**



Лабораторна робота №3

з дисципліни «Об’єктно орієнтоване програмування»

«ДОСЛІДЖЕННЯ МЕХАНІЗМУ СЕРІАЛІЗАЦІЇ ТА ТИПОВОЇ ТРИРОВНЕВОЇ АРХІТЕКТУРИ»

Виконав студент:

групи ПІ-223Б

Карпов І.Б.

Перевірив викладач:

Дишлевий О.П.

Київ 2022

**Мета:** дослідити механізм серіалізації у мові програмування С# та навчитися розділяти функціонал застосування між рівнями архітектури.

**Завдання**

**Частина 1.** Дослідження механізму серіалізації. Усі завдання частини 1 допускається реалізовувати в одному проекті одного рішення

1. Описати клас, заданий варіантом (табл 1), та передбачити для нього можливість серіалізації.
2. Створити масив об’єктів класу, вказаного в п.1, та серіалізувати їх у файл.
3. Створити новий масив та відновити в ньому значення серіалізованих об’єктів.
4. Виконати серіалізацію та десеріалізацію об‘єктів будь-якої колекції. Порівняти з масивом.
5. Продемонструвати використання бінарної серіалізації, XML-серіалізації, JSON та користувацької серіалізації:

* для отримання оцінки «задовільно» реалізувати будь-яку серіалізацію
* для отримання оцінки «добре» реалізувати бінарну, користувацьку, XML та JSON серіалізацію.

**Частина 2.** Дослідження використання серіалізації в типовій трирівневій архітектурі застосування

1. Спроектувати застосування із розділенням рішення (solution) на 3 рівні, які реалізовуються 3- ма різними проекти: рівень доступу до даних (DAL), рівень бізнес-логіки (BLL), рівень представлення (PL).

В рішенні (solution) лабораторної роботи повинні бути:

* Проект з класом Program, в якому є метод Main(), за допомогою якого запускається застосування. В методі Main() повинен бути тільки виклик методу MainMenu() з класу Menu().
* Проект з класом EntityService. На цей проект повинен посилатися проект з класом Menu.
* Проект з класом EntityContext. На цей проект повинен посилатися проект з класом EntityService.

В кожному проекті можуть бути також допоміжні класи та інтерфейси. Так само, як і в кожному класі можуть бути також інші члени, окрім тих, що вказані в завданні.

Також можна додавати в рішення інші проекти, які будуть сприяти ізоляції рівнів.

1. Реалізувати завдання за варіантом з табл. 2. Передбачити взаємодію з кожною сутністю. А також збереження та читання з файлів за допомогою механізму серіалізації:

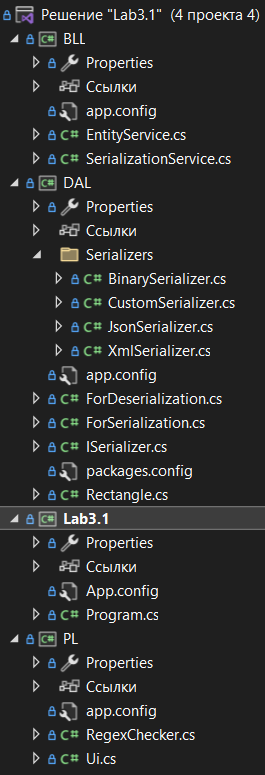
* для отримання оцінки «задовільно» дозволяється реалізувати тільки клас (-и) EntityContext на рівні DAL, EntityService на рівні BLL та Menu на рівні BLL (за необхідності – статичний). Клас Entity може використовуватися будь-де. Може бути реалізований тільки один спосіб серіалізації (як в ч. 1);
* для отримання оцінки «добре» необхідно реалізувати на рівні BLL власний клас (-и) Exception (відповідно предметній області). Обробка вийняткових ситуацій повинна бути не в момент їх виникнення/ генерації. Серіалізація повинна бути XML або JSON;
* для отримання оцінки «відмінно» клас(-и) Entity повинен бути тільки на рівні DAL. Для зменшення зв’язків між рівнями, в BLL та PL повинні бути реалізовані власні класи сутностей. Клас Menu повинен бути в окремому проекті. Крім імплементації бінарної, XML, JSON та користувацької серіалізації, користувач повинен мати можливість вибору типу та імені файлу, в який потрібно серіалізувати дані. Для цього назва файлу та розташування повинні передаватися з рівня представлення (PL) на нижчі рівні. А також - створити абстракцію DataProvider, яка відповідає за роботу з файлами відповідного типу (для кожного типу серіалізації – свій провайдер). Уникати повторення коду шляхом використання абстракцій та узагальнень.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Варіант** | **Дані** | | **Методи та властивості** | | **Кількість об’єктів** | |
| 5 | Прямокутник:  Колір заповнення, колір контуру, довжини сторін | | Обчислення площі;  Обчислення периметру;  Виведення інформації про фігуру | | 5 | |
| **Варіант** | **Елементи класу Student** | **Операції з студентами** | | **Додаткові сутності** | | **Додаткові вміння** |
| 5 | Прізвище, курс, Студентський квиток, Середній бал, Країна, номер залікової книжки | Обчислити кількість студентів 3-го курсу, які проживають в Україні. Отримати їх дані. | | McdonaldsWorker,  Manger | | Грати в шахи |

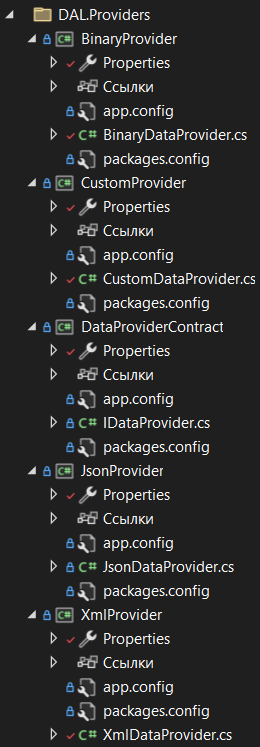
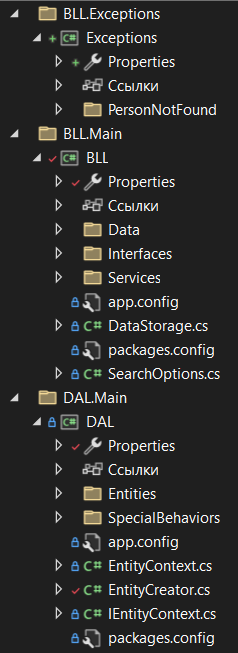
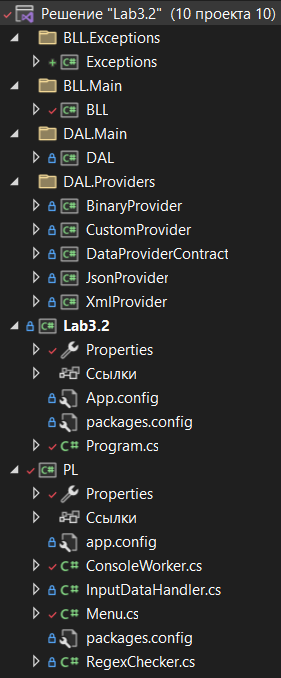
**Хід роботи**

Код програми та приклад роботи зберігається на GitHub за цим посиланням:  
<https://github.com/IvanKarpov-1/course2-term1-OOP-lab3>

Частина 1



Частина 2



**Відповіді на контрольні запитання**

1. **Розкрийте сутність та призначення механізму серіалізації.**

Серіалізація використовується для передавання об'єктів мережею й для збереження їх у файлах. Наприклад, потрібно створити розподілений застосунок, різні частини якого мають обмінюватися даними зі складною структурою. У такому випадку для типів даних, які передбачається передавати, пишеться код, який здійснює серіалізацію і десеріалізацію. Об'єкт заповнюється необхідними даними, потім викликається код серіалізації, в результаті виходить, наприклад, XML-документ. Результат серіалізації передається приймальній стороні, наприклад, електронною поштою або через HTTP. Додаток-одержувач створює об'єкт того ж типу і викликає код десеріалізації, у результаті отримуючи об'єкт із тими ж даними, які були в об'єкті програми-відправника. За такою схемою працює, наприклад, серіалізація об'єктів через SOAP в Microsoft.NET.

1. **Яка альтернатива серіалізації існує в мові С++?**

Будь-якій зі схем серіалізації властиво те, що кодування даних послідовно за визначенням, і вибірка будь-якої частини серіалізованої структури даних вимагає, щоб весь об'єкт був зчитаний від початку до кінця і був відновлений. У багатьох програмах така лінійність корисна, тому що дозволяє використовувати прості інтерфейси введення/виведення загального призначення для збереження і передачі стану об'єкта. У додатках, де важлива висока продуктивність, можливо буде доречніше використовувати складнішу, нелінійну організацію зберігання даних.

1. **Які типи серіалізації існують в С#?**

Єдина вимога основної серіалізації - це те, що до об'єкта застосовано атрибут SerializableAttribute . NonSerializedAttribute може використовуватися , щоб зберегти певні поля від сериализации. Під час використання базової серіалізації версії об'єктів можуть створювати проблеми. Ви користуєтеся спеціальною серіалізацією, коли важливі проблеми з версіями. Базова серіалізація - найпростіший спосіб здійснити серіалізацію, але вона не забезпечує особливого контролю над процесом.

1. **Опишіть механізм стандартної серіалізації.**

Об'єкт серіалізується в потік, який служить передачі даних. Потік також може містити відомості про тип об'єкта, у тому числі про його версію, мову та регіональні параметри, а також ім'я складання. У цьому форматі потоку об'єкт можна зберегти у базі даних, файлі чи пам'яті.

1. **Опишіть механізм бінарної серіалізації.**

Бінарна серіалізація використовує двійкове кодування для отримання компактної серіалізації для таких застосувань, як мережеві потоки на основі зберігання або на сокетах.

1. **Опишіть механізм користувацької серіалізації.**

Користувацька серіалізація - це особлива форма серіалізації, яка передбачає вид стійкості об'єкта, пов’язаний із інструментами розробки. Конструкторська серіалізація - це процес перетворення об’єктного графіка у вихідний файл, який згодом може бути використаний для відновлення об’єктного графіка. Вихідний файл може містити код, розмітку або навіть інформацію таблиці SQL.

1. **Опишіть механізм XML серіалізації.**

XML-серіалізація серіалізує публічні поля та властивості об’єкта або параметри та значення повернення методів у потік XML, який відповідає конкретному документу мови визначення схеми XML (XSD). Серіалізація XML призводить до сильно набраних класів із загальнодоступними властивостями та полів, які перетворюються на XML. System.Xml.Серіалізація містить класи, необхідні для серіалізації та десеріалізації XML. Ви застосовуєте атрибути до класів та членів класу, щоб контролювати те, як XmlSerializer серіалізує або десеріалізує екземпляр класу.

1. **Опишіть механізм JSON серіалізації.**

У користувацькій серіалізації ви можете вказати, які саме об'єкти будуть серіалізовані та як це буде зроблено. Клас повинен бути позначений SerializableAttribute та реалізувати інтерфейс ISerializable .Якщо ви хочете, щоб ваш об'єкт був десеріалізований і в користувальницькому порядку, ви повинні використовувати спеціальний конструктор.

1. **В чому полягає концепція багаторівневої архітектури застосування? Для чого вона потрібна?**

Серіалізація дозволяє розробнику зберігати стан об’єкта та відтворювати його за необхідності, забезпечуючи зберігання об’єктів, а також обмін даними. За допомогою серіалізації розробник може виконувати такі дії, як надсилання об’єкта у віддалений додаток за допомогою веб-сервісу, передача об'єкта з одного домену в інший, передача об'єкта через брандмауер у вигляді рядка XML або підтримка безпеки або специфіки для користувача інформація в додатках.

1. **Які існують рівні і як вони зв’язані між собою?**

Бінарна та XML серіалізація, двійкова серіалізація, XML-серіалізація, під час використання базової серіалізації версії об'єктів можуть створювати проблеми. Ви користуєтеся спеціальною серіалізацією, коли важливі проблеми з версіями. Базова серіалізація - найпростіший спосіб здійснити серіалізацію, але вона не забезпечує особливого контролю над процесом.

1. **Чому необхідно ізолювати рівні чи окремий функціонал?**

Щоб розділити права доступу між рівнями. Так дата рівень має дозвіл на роботу з файлами і рівень бізнес логіки, якщо захоче якимось чином змінити файл, повинен звернутися до дата рівня. Взагалі щоб зробити код більш гнучким і читабельним.

1. **Як розподіляються класи між окремими рівнями та проектами в рішенні?**

Класи розподіляються на три рівні:

1. DAL – рівень доступу до даних, нижчий рівень;
2. BLL – рівень бізнес логіки, проміжний середній рівень;
3. PL – рівень представлення, вищий рівень.

В рішенні може бути тільки три проекти - DAL, BLL та PL, але допускається створення більшої кількості проектів, для відображення цих трьох рівнів.

1. **Яким чином допомагає абстракція при проектуванні багаторівневої архітектури?**

Поділяючи додаток на рівні абстракції, розробники набувають можливість внесення змін в якийсь певний шар, замість того, щоб переробляти все додаток цілком.

1. **Чим можуть бути корисні узагальнення та колекції при серіалізації та проектування багаторівневої архітектури?**

Узагальнена та колекції можуть бути корисними, коли розробнику потрібно розробити одну функціональність, яка може працювати з багатьма типами. Узагальнення та колекції допомагають позбутися повторювального коду, за рахунок створення загального типу, виведення абстракцій з якими в подальшому буде відбуватися робота.

1. **Чому необхідно, щоб метод виконував тільки одну задачу?**

Щоб розділити права доступу між рівнями. Взагалі щоб зробити код більш гнучким і читабельним.