Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**Лабораторная работа**

по предмету **«Методы и механизмы проектирования программных продуктов»**

Тема: **Шаблоны проектирования — Порождающие (англ. Design Patterns – Creational)**

Студент: Никита Маршагин TI-212 FR

Преподаватель: Андрей Постару

Кишинёв 2025**Цель работы**

Изучить и реализовать три поведенческих шаблона проектирования:

**Singleton (Одиночка)**

**Builder (Строитель)**

**Prototype (Прототип)**

**Теоретическая часть**

**Singleton (Одиночка)**

Порождающий шаблон, гарантирующий, что у класса существует только один экземпляр, и предоставляющий к нему глобальную точку доступа.

Применяется, когда важно, чтобы объект был единым — например, логгер, конфигурация, подключение к базе данных. Singleton помогает централизовать управление ресурсом и не допускать его дублирования.

**Builder (Строитель)**

Шаблон, предназначенный для пошагового построения сложных объектов с различными параметрами и конфигурациями.

Позволяет разделить процесс создания объекта от его представления, что особенно полезно при создании объектов с множеством опциональных параметров. Пример — конструктор оружия, формы, интерфейса или конфигуратора автомобиля.

**Prototype (Прототип)**

Позволяет создавать новые объекты на основе уже существующих путём их клонирования, а не создания "с нуля".

Особенно полезен, когда создание объекта — это дорогой или ресурсоёмкий процесс. Использование прототипов упрощает повторное создание объектов с одинаковой структурой, но с разными значениями. Пример — клонирование персонажей, копии форм, шаблоны данных.

**Практическая часть**

**Singleton (Одиночка)**

**Описание:** реализация класса Singleton, гарантирующего создание только одного экземпляра. При повторном вызове конструктора возвращается уже существующий объект. Используется свойство instance для хранения ссылки. Примером выступает объект с полем time, фиксирующим момент создания.

<https://raw.githubusercontent.com/Niki-Mar/TMPP_Labs/main/Lab1/Singleton.js>  
  


Рисунок 1 Вывод работы паттерна Singletonr

**Builder (Строитель)**

**Описание:** реализован класс WeaponBuilder, предоставляющий пошаговую настройку объекта оружия. Методы setCategory() и setDamage() возвращают this, позволяя вызывать их в цепочке. Метод build() возвращает готовый объект. Таким образом создаются разные виды оружия с индивидуальными параметрами.

<https://raw.githubusercontent.com/Niki-Mar/TMPP_Labs/main/Lab1/Builder.js>



Рисунок 2 Вывод работы паттерна Builder

**Prototype (Прототип)**

**Описание:** реализован объект characterPrototype с методом clone(), основанным на Object.create(this). Клонированные объекты наследуют структуру прототипа и могут быть индивидуализированы. Используется для создания новых персонажей с базовыми свойствами, которые можно переопределить.

[https://raw.githubusercontent.com/Niki-Mar/TMPP\_Labs/main/Lab1/Prototype.js](https://raw.githubusercontent.com/Niki-Mar/TMPP_Labs/main/Lab1/Prototype.js%20)

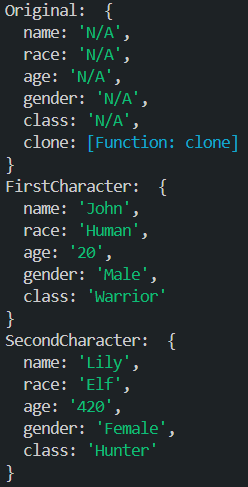
****

Рисунок 3 Вывод работы паттерна Prototype

**Выводы:**

* Реализованы 3 шаблона создания объектов.
* Получен опыт проектирования с разделением ответственности.
* Шаблоны повышают читаемость, гибкость и переиспользуемость кода.
* Все реализации протестированы в JavaScript-среде.