**🔑 Основные принципы ООП в ЛР5-8:**

**1. Абстракция**

Абстракция — выделение сущностей с важными характеристиками, скрывая детали реализации.

* **Интерфейс IHeater** определяет базовые функции (setName, getPrice, print и т.д.), которые должны реализовать все обогреватели. Это позволяет обращаться к объектам через общий интерфейс, не зная их внутренней реализации.

**2. Наследование**

Позволяет создавать иерархию классов и переиспользовать код.

* Heater наследует интерфейс IHeater.
* SealedHeater и UnsealedHeater — конкретные реализации, **унаследованные от Heater**.
* SpiralHeater — более специализированный класс, **унаследованный от UnsealedHeater**, тем самым показывая **многоуровневое наследование**.
* UnsealedHeater **множественно наследуется** от Heater и Material.

**3. Инкапсуляция**

Скрытие реализации от пользователя.

* В классе Heater поля name, price, power закрыты (private), доступ к ним осуществляется через **сеттеры/геттеры** (setName, getPower, и т.д.).
* В Material тоже закрытые поля material и durability.

**4. Полиморфизм**

Позволяет использовать объекты производных классов через интерфейсы или указатели на базовый класс.

* Благодаря интерфейсу IHeater, объекты разных классов (SealedHeater, SpiralHeater) можно хранить в одном контейнере (TemplateLister<IHeater>) и вызывать print(), что работает по принципу **виртуальных функций** (динамический полиморфизм).
* Метод print() виртуальный, а значит, будет вызываться корректная реализация для конкретного типа обогревателя.

**🧱 Структура проекта с точки зрения ООП:**

| **Класс** | **Назначение** | **Особенности** |
| --- | --- | --- |
| IHeater | Интерфейс обогревателя | Абстрактный класс |
| Heater | Базовый класс | Реализация общих свойств |
| SealedHeater | Запечатанный обогреватель | Наследует Heater |
| UnsealedHeater | Незапечатанный обогреватель | Наследует Heater, Material |
| SpiralHeater | Спиральный обогреватель | Наследует UnsealedHeater |
| Material | Материал | Используется в UnsealedHeater |
| Lister, TemplateLister | Связный список объектов | Шаблонный список, поддерживает разные типы |
| Statistics | Хранение и анализ данных | Использует map для хранения |

**📌 Особенности реализации:**

* **Шаблоны (TemplateLister<T>)** — пример обобщённого программирования (generic programming), где ты можешь создать список любых объектов, реализующих нужные методы.
* **Проверка исключений** — при установке некорректных значений (имя пустое, отрицательная цена/мощность) выбрасываются исключения (throw exception, out\_of\_range).
* **Операторные перегрузки** — сравнение по цене, вывод объектов (operator==, operator<, operator>), пусть и с недоработанным << (его нужно исправить).

**🔍 Итог:**

Твоя лабораторная демонстрирует хорошие знания базовых концепций ООП:

* Используется **интерфейс + наследование**.
* Есть **инкапсуляция** через геттеры/сеттеры.
* Есть **полиморфизм** с виртуальными методами.
* Используются **шаблоны и STL-контейнеры (map)**.
* Есть **обработка ошибок и исключений**.

Хочешь, чтобы я помог оформить текст для отчёта или сделать UML-диаграмму классов?