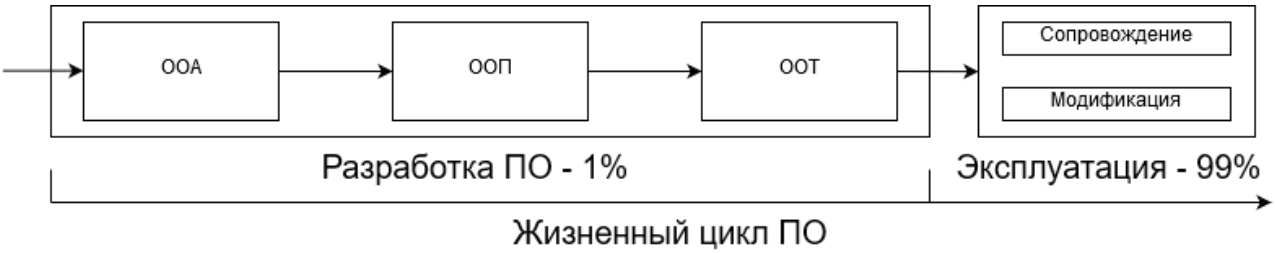


Объектно-ориентированный анализ это методология, при которой требования к системе воспринимаются с точки зрения класса и объектов, выявленных в предметной области.

Объектно-ориентированное проектирование - методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектирования системы.

Объектно-ориентированное программирование - это методология программирования, основанная на представлениии программы в виде совокупности объектов, каждый из которых может являться экземпляром определенного класса или типа, а каждый класс или тип образует иерархию наследования и взаимодействия.

Принципы объектно-ориентированного программирования

1. Абстрагирование
2. Ограничение доступа
3. Модульность
4. Иерархическая реализация (Дополнительные)
5. Параллелизм
6. Типизация
7. Устойчивость

Абстрагирование - процесс выделения абстракций. Абстракция это совокупность существенных характеристик объекта, которые отличают его от других видов, то есть, четко определяют данный объект с точки зрения дальнейшего рассмотрения и анализа. (Абстракция = класс).

Существует три классических способа классификации объектов.

1. Классическая категоризация. Пример: кот может бежать или не бежать. Тогда объект может быть бегущим или не бегущим.
2. Концептуальная классификация (кластеризация). Кластеризация определяет степень принадлежности к какому-то классу.
3. ?

```
class <ИмяКласса>{
    (по умолчанию private)
    private:
    protected: (необязательно)
    public:
    [<имя типа><переменная>];
    [<имя типа><сигнатура метода>];
}
```

Ограничение доступа - это сокрытие отдельных элементов абстракции, не затрагивающий ее существенных характеристик. Необходимость ограничения доступа предполагает выделение двух частей:

1. Интерфейс (совокупность доступных извне абстракции характеристик, состояний и поведения абстракции)
2. Совокупность недоступных извне элементов абстракции, включает внутреннюю организацию абстракции и ее реализацию.

Ограничение доступа позволяет разработчикам:

1. Выполнять конструирование программы по шагам, начиная с разработки интерфейса (проектировать сверху вниз)
2. Позволяет изменять один элемент абстракции, не затрагивая другие.

Инкапсуляция - это сочетание объединения всех свойств предметов, характеризующих его состояние и поведение в единой абстракции с ограничением доступа к ее реализации.

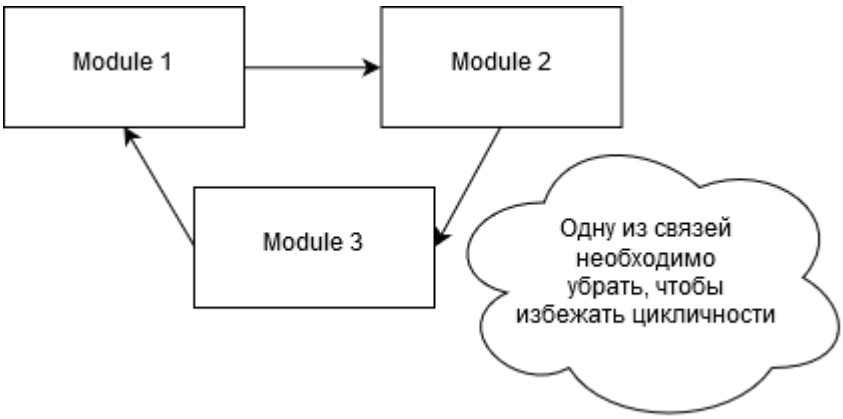
Пример

```
//Не универсально, так делать нехорошо
class Key{
    public:
    // В public секции открыта реализация
        void input(Door *d);
        void rotate();
        void take(Door *d);
}
//Гораздо универсальней
class Key{
    private:
    void input(Door *d);
    void take(Door *d);
    public:
    int openDoor(Door *d);
}
```

Разработка класса - это логическое проектирование систем, физическим проявлением является модульность.

Модульность - принцип разработки программной системы, которая предполагает ее реализацию в отдельных модулях. Модуль - это физический контейнер некоторого набора логических элементов. Следование этому принципу значительно упрощает разработку.

Циклическая зависимость модулей может вызвать проблемы при изменении любого из них. Модуль должен иметь минимум связей с внешним миром.



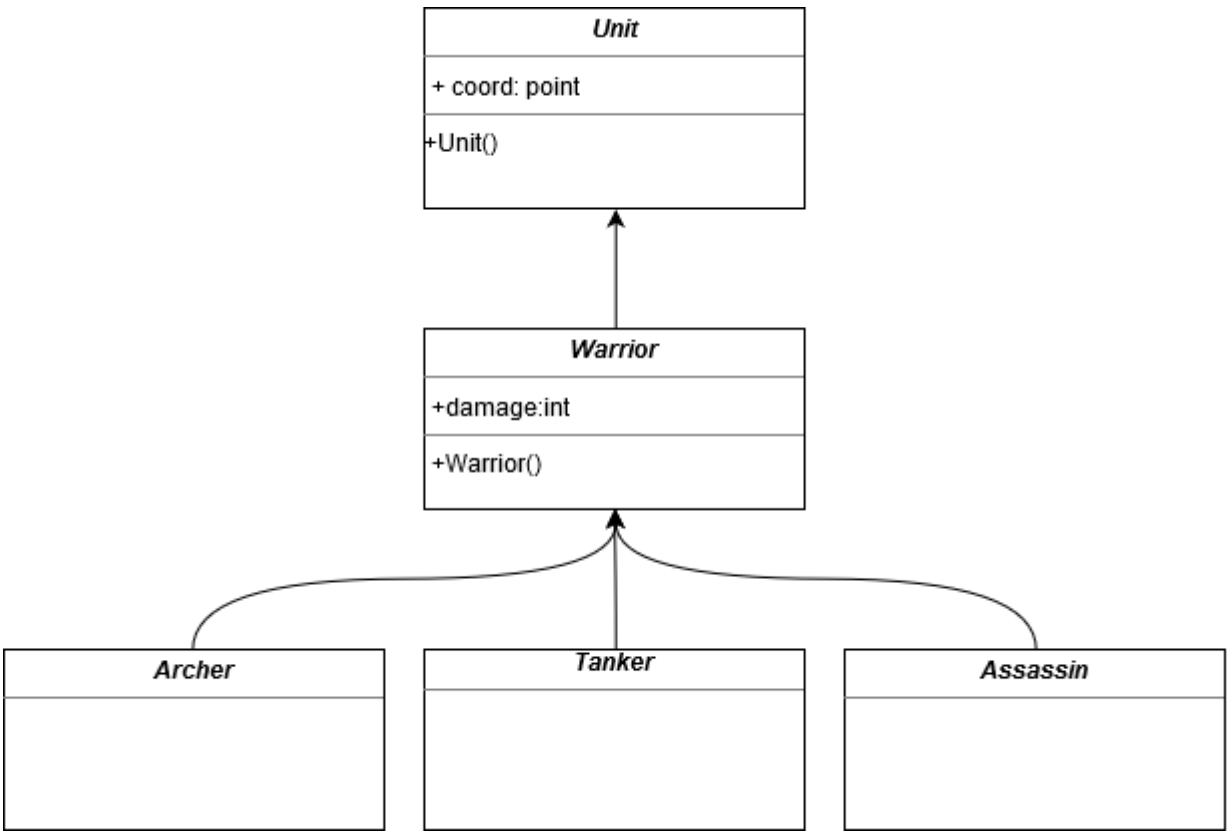
Модули в C++ могут храниться в файлах .h, .hpp, .e, .exx

Иерархическая организация предполагает использование иерархий при разработке программной системы. Иерархии упрощают систему абстракций. Иерархия - это упорядочивание абстракций, расположение их по уровню.

Существуют два вида иерархий:

1. *Часть - целое (Part-All).* Предполагает, что некоторая абстракция включена в другую. Данный вид иерархии используется в ранних этапах проектирования. На логическом уровне при разбиении предметной области на объекты, на физическом уровне - при разбиении систем на модули. (связь между объектами - агрегация)
2. *Общее - частное (Is - A).* Показывает, что некоторая абстракция является частным случаем другой. Используется в основном механизмее ООП - наследовании.

Наследование - это отношение между классами, когда один из них использует структурную или функциональную часть другого.



Типизация - ограничение, накладываемое на свойства объектов, препятствующее взаимозаменяемости абстракций различного типа. Использование принципа типизации помогает выполнить раннее обнаружение ошибок, упрощает комментирование, помогает генерировать более эффективный код

Параллелизм - свойство, которое позволяет нескольким абстракциям одновременно находиться в активном состоянии. Данный принцип реализует ОС.

Устойчивость (сохраняемость) - свойство абстракции существовать во времени, независимо от процесса, породившего ее. Различают следующие типы объектов:

1. Временные объекты
2. Локальные объекты
3. Глобальные объекты
4. Сохраняемые объекты (на внешних носителях)