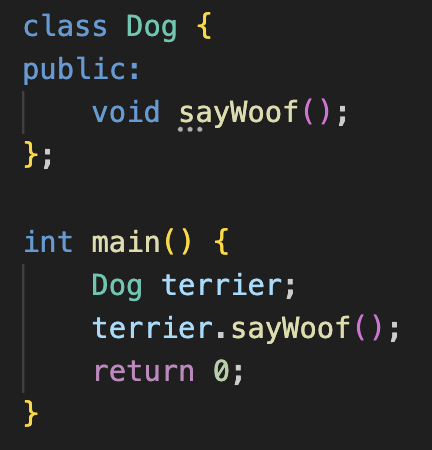
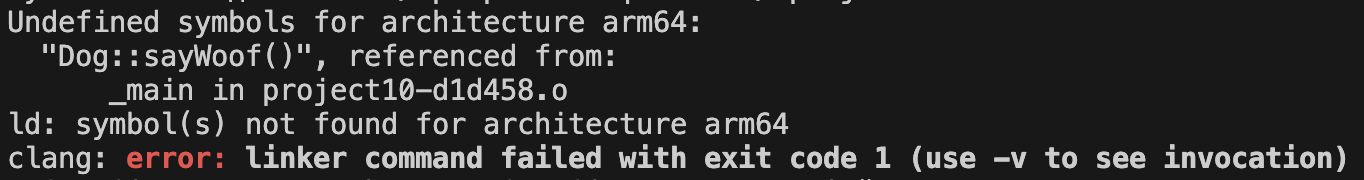
**Отчет курсанта Громова Григория Андреевича группы 22.Б05 о выполнении практического задания на тему «абстрактные классы» (десятое задание в третьем семестре)**

Абстрактные классы характеризуются наличием чисто виртуальных функций. Это означает, что в классе есть функции, у которых не написана реализация. То есть мы не можем создать экземпляр такого класса и обратиться к его методам – компилятор выдаст ошибку:



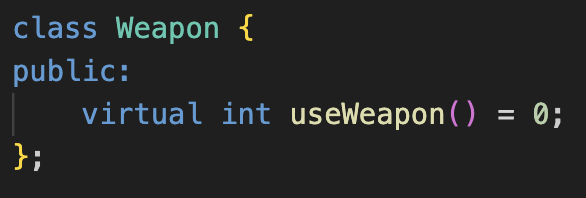


Зачем же нужны такие классы? Давайте рассмотрим пример:

Я предлагаю вспомнить пример из прошлого проекта (ДЗ №9) и немного его доработать. В прошлый раз, у нас был родительский класс Gun, у которого были наследники MachineGun и Rocket.

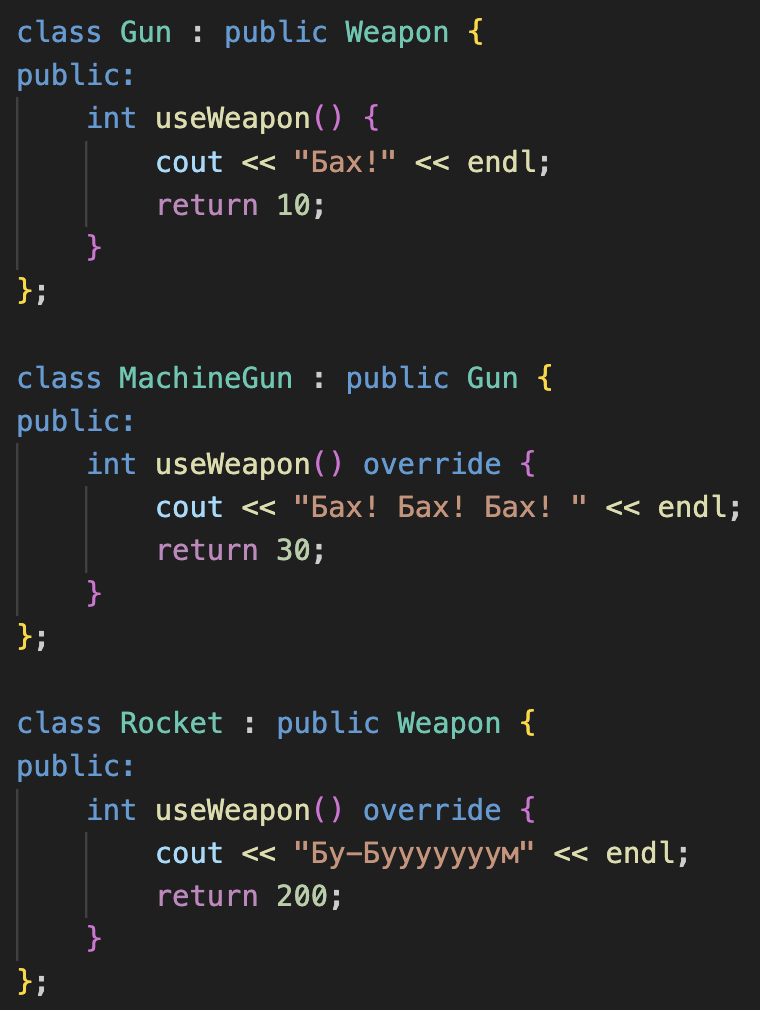
Уже в этот момент можно заметить некоторую нестройность: почему ракета является наследником пистолета? Ситуация усугубится, если мы захотим добавить в арсенал игрока лук или тем более нож. Они уж точно по смыслу не являются наследниками пистолета (а значит при расширении функциональности, например, при добавлении анимации возникнут новые проблемы).

Неплохим решением будет создать какой-то общий виртуальный класс *Oружие* и сделать имеющиеся классы его потомками. Создадим класс Weapon с чисто виртуальной функцией useWeapon():



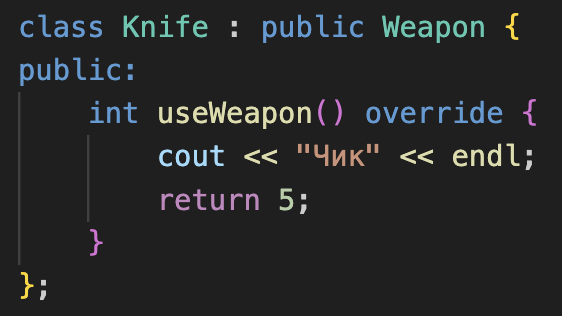
( **= 0** - мы указываем на то, что функция чисто виртуальная)

Теперь изменим старые классы оружия:

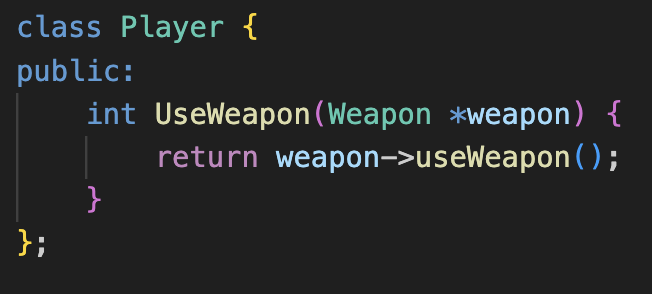


Обратите внимание, что мы по-прежнему оставили MachineGun наследником Gun, но при этом функция useWeapon в Gun перестала быть виртуальной. Дело в том, что MachineGun является «внуком» Weapon, а у него есть чисто виртуальная функция useWeapon.

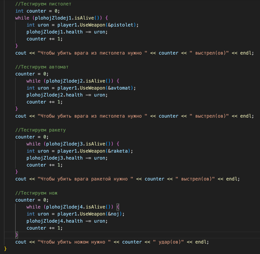
И теперь можно добавить нож:



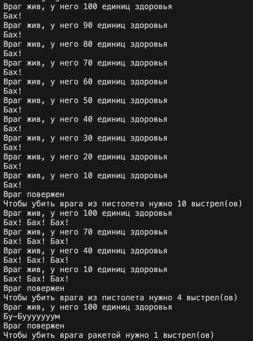
Немного изменим класс игрока (просто поменяем тип указателя с Gun на Weapon):



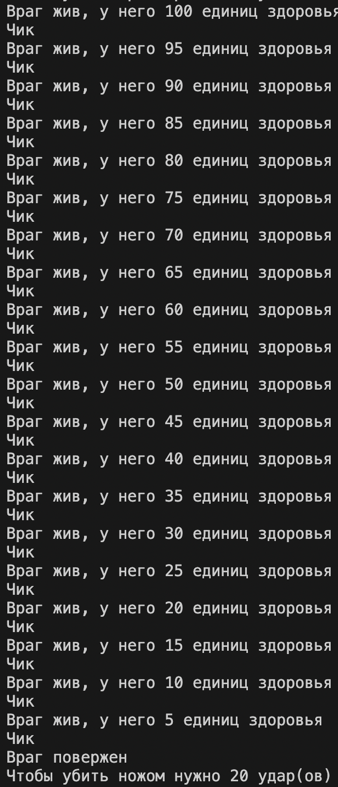
Добавим в main новое оружие нож и его тестирование:



При использовании старого оружия все осталось как было:



При использовании ножа всё тоже корректно работает:



Вывод: в принципе, мы можем обойтись и без абстрактных классов. Однако их использование помогает более логично организовать взаимодействие между объектами и сделать код более читабельным. Это будет особенно полезно при написании больших проектов несколькими программистами.