# Задание на применение технологии ООП

## Постановка задачи

Требуется разработать программы согласно указанным ниже заданиям (на выбор, но можно реализовать свою тему).

Реализация классов должна быть в отдельных модулях.

Клиентский код должен импортировать вашу библиотеку и посредством объектов воспользоваться реализуемой в них функциональностью.

В тексте использовать комментарии, поясняющие ваши действия и принятые решения.

Объем задания (конкретные указания, что надо выполнить) указан ниже.

Программное решение общего задания требуется представить в виде отдельных решений по каждому заданию, которые должны показать развитие общего проекта.

## Задание 1. Базовое решение

* Реализовать класс с конструктором. Определить значения по умолчанию для некоторых аргументов, чтобы их можно было не указывать в тех случаях, когда какие-то определенные значения недоступны или бессмысленны.
* Реализовать с помощью свойств инкапсуляцию.
* Добавить в класс методы, определяющие поведение.
* Реализовать производные классы (не менее двух) и методы, характерные для них.
* Реализовать отношение композиции.
* Реализовать для базового класса перегрузку двух любых стандартных операторов, например, сложения и вычитания. Методы перегрузки должны возвращать новый объект того же класса.

## Задание 2. Множественное наследование

* Реализовать некую сущность вашего решения (обосновать свой выбор) как абстрактный класс.
* Изменить реализацию наследования, добавив наследование от абстрактного класса.
* Применить в иерархии классов наследование от класса-миксина.
* Продумать и по возможности реализовать множественное наследование (в случае отказа от реализации множественного наследования обосновать свое решение).

## Задание 3. Расширение функциональности

* Реализуйте в своем проекте декораторы, которые

1) регистрирует вызовы важных сценариев.

Результат декорирования – информация о выполнении сценария (например, факт отработки, время начала и окончания, длительность и т.д.).

2) устанавливает/проверяет пре- и постусловия функций.

3) что-либо на ваше усмотрение.

* Реализуйте в своем проекте метаклассы для расширения классов дополнительными возможностями (на ваше усмотрение).

## Задание 4. Исключения

* Реализуйте в своем проекте поддержку исключений.
* Создайте в своем проекте классы исключений и покажите их применение.
* Предложите в вашей процедуре (любой на ваш выбор) реализацию возбуждения исключения в случае нахождения некого соответствия вместо того, чтобы возвращать флаг состояния, который должен интерпретироваться вызывающей программой, т.е. с помощью исключения обеспечить способ подачи сигнала, не возвращая значение.

## Задание 5. Многопоточность и асинхронность

* Добавьте в свой проект реализацию вторичного потока для любого алгоритма.
* Подберите наиболее наглядный пример, обоснуйте свой выбор.
* Добавьте в свой проект реализацию асинхронного вызова для любого алгоритма.
* Подберите наиболее наглядный пример, обоснуйте свой выбор.

## Задание 6. Разработка графического интерфейса пользователя

Реализуйте модуль ввода данных GUI с помощью библиотеки tkinter (в ООП стиле) для любого алгоритма (сценария) вашей задачи.

**Отчетность**

Проект в виде файлов исходного кода. Отчетом о выполненной работе являются комментарии в этих файлах – рекомендуется сделать их как можно подробными и понятными.

# Варианты заданий

На примере первого задания показан примерный перечень действий по проектированию системы.

## Задание. Пиццерия.

**Постановка задачи.**

Пиццерия предлагает клиентам три вида пиццы: *Пепперони*, *Барбекю* и *Дары Моря*, каждая из которых определяется тестом, соусом и начинкой.

**Требуется:**

* спроектировать и реализовать приложение для терминала, позволяющее обеспечить обслуживание посетителей.

**Рекомендация по проектированию**

1. По результатам анализа предметной области (бизнес-процесса работы пиццерии) в контексте задачи можно выделить 3 сущности (объекта):

* Терминал: отвечает за взаимодействие с пользователем:
  + вывод меню на экран;
  + прием команд от пользователя (выбор пиццы, подтверждение заказа, оплата и др.);
* Заказ: содержит список заказанных пицц, умеет подсчитывать свою стоимость;
* Пицца: содержит заявленные характеристики пиццы, а также умеет себя подготовить (замесить тесто, собрать ингредиенты и т.д.), испечь, порезать и упаковать.

Поскольку пиццерия реализует несколько видов пиццы, которые различаются характеристиками, логично будет сделать общий класс ***Пицца***, а в дочерних классах (например, классе ***ПиццаБарбекю***) уточнить характеристики конкретной пиццы.

1. Рекомендуется разработать для указанных классов диаграмму классов в нотации UML.
2. Разработать алгоритм работы (сценарий) пользователя с терминалом (основным классом, отвечающим за интерфейс).

## Задание. Банковские вклады

Банк предлагает ряд вкладов для физических лиц:

* Срочный вклад: расчет прибыли осуществляется по формуле простых процентов;
* Бонусный вклад: бонус начисляется в конце периода как % от прибыли, если вклад больше определенной суммы;
* Вклад с капитализацией процентов.

Требуется реализовать приложение, которое бы позволило подобрать клиенту вклад по заданным параметрам.

## Задание. Поединок

Дан класс "Боец" – базовый класс.

Требуется реализовать имитацию поединка двух соперников.

Реализовать производные классы (не менее двух) – бойцов определённого единоборства.

От них создаются два экземпляра-юнита.

Примерная последовательность действий:

* Перед началом поединка каждому устанавливается здоровье в 100 очков.
* В ходе поединка противники в случайном порядке бьют друг друга. Тот, кто бьет, здоровья не теряет. У того, кого бьют, оно уменьшается на 20 очков от одного удара.
* После каждого удара надо выводить сообщение, какой юнит атаковал, и сколько у противника осталось здоровья.
* Как только у кого-то заканчивается ресурс здоровья, программа завершается сообщением о том, кто одержал победу.

## Задание. Шахматы

Требуется реализовать объектную модель *Шахматные фигуры*.

У всех шахматных фигур есть общий функционал, например - возможность фигуры ходить и быть отображенной на доске.

Исходя из этого, необходимо создать абстрактный класс *Фигура*, определить в нем абстрактный метод (например, - ход, поскольку каждая фигура ходит по-своему) и реализовать общий функционал (например, отрисовка на доске).

Создать класс конкретной фигуры, например, ферзя, в котором реализовать нужный функционал.