Федеральное государственное автономное образовательное учреждение высшего образования

«ОМСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Прикладная математика и фундаментальная информатика»

Практическое занятие №11

по дисциплине «Практикум по программированию»

на тему: «Проектирование и разработка классов в Python»

Вариант №14

Выполнил

Студент гр. **ФИТ-212** **Курпенов К.И.**

группа Фамилия И,О. студента подпись

Принял:

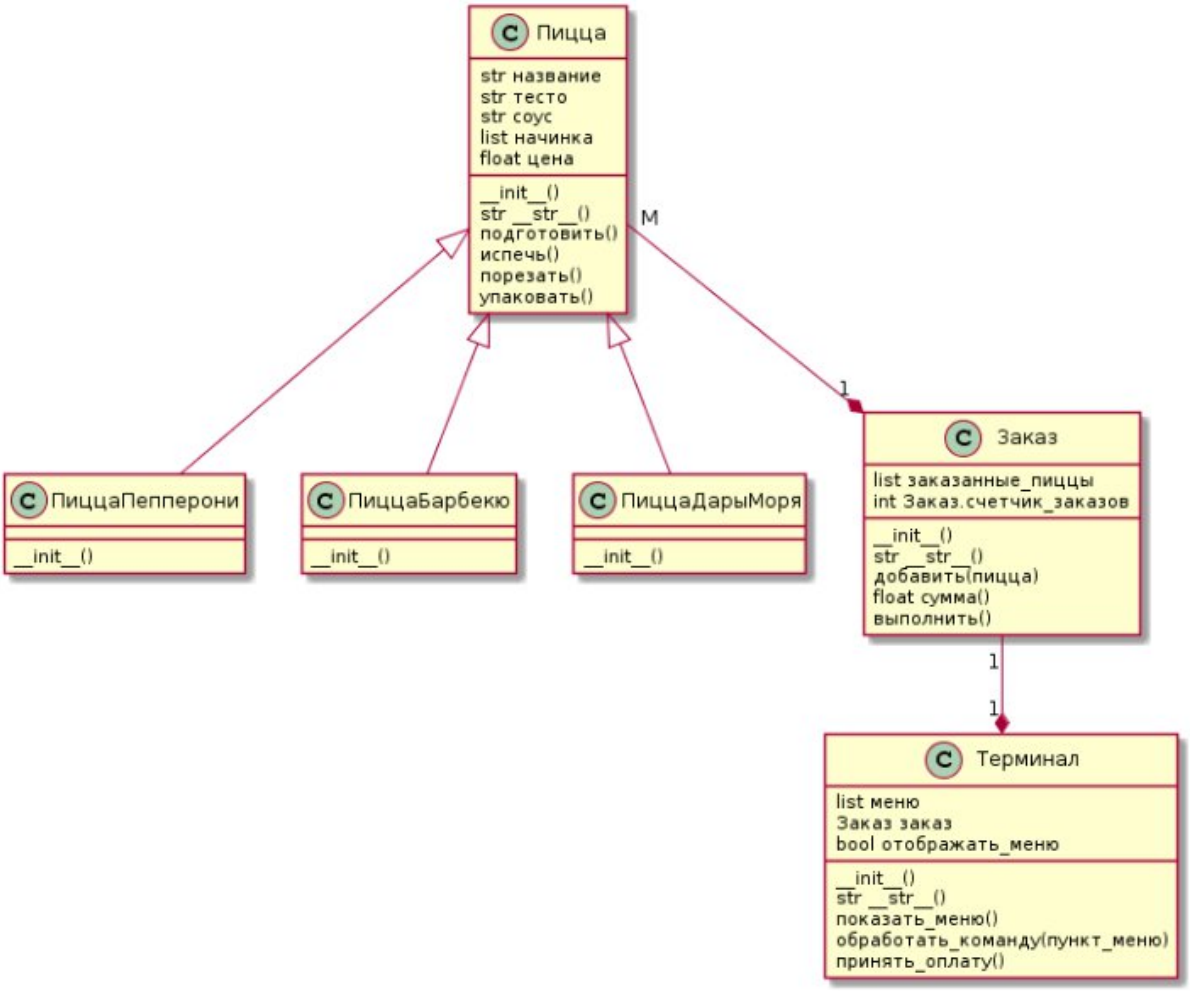
Преподаватель **Моисеева Н.А.**

Фамилия И.О, преподавателя дата, подпись

Омск 2022

Задача 1.

Условие:



Решение:

```python
class Pizza:
    def __init__(self, name: str, dough: str, sauce: str, filling: list, cost: float) -> None:
        self.name = name
        self.dough = dough
        self.sauce = sauce
        self.filling = filling
        self.cost = cost

    def __str__(self) -> str:
        return f"{self.name} from {self.dough} with {self.sauce} and {self.filling}: {self.cost}"

    def prepare(self) -> str:
        return f"[+] Preparing"

    def bake(self) -> str:
        return f"[+] Baking"

    def cut(self) -> str:
        return f"[+] Cutting"

    def pack(self) -> str:
        return f"[+] Packing"


class Pepperoni(Pizza):
    def __init__(self, name: str, dough: str, sauce: str, filling: list, cost: float) -> None:
        super().__init__(name, dough, sauce, filling, cost)


class Barbecue(Pizza):
    def __init__(self, name: str, dough: str, sauce: str, filling: list, cost: float) -> None:
        super().__init__(name, dough, sauce, filling, cost)


class SeaGifts(Pizza):
    def __init__(self, name: str, dough: str, sauce: str, filling: list, cost: float) -> None:
        super().__init__(name, dough, sauce, filling, cost)

```

```python
class Order:
    def __init__(self, orders: list) -> None:
        self.orders = orders
        self.count = len(orders)

    def __str__(self) -> str:
        return f"[+] Orders: {self.orders}"

    def add(self, pizza: Pizza) -> None:
        self.orders.append(pizza)
        self.count += 1

    def sum(self) -> float:
        result = 0

        for order in self.orders:
            result += order.cost

        return result

    def perform(self) -> None:
        pass


class Terminal:
    def __init__(self, menu: list, order: Order, show: bool) -> None:
        self.menu = menu
        self.order = order
        self.show = show

    def __str__(self) -> str:
        return f"[+] Order: {self.order}"

    def showMenu(self) -> str:
        result = ""

        for element in self.menu:
            result += f"{element}\n"

        return result

    def command(self) -> str:
        return f"[+] Command is done"

    def acceptPayment(self) -> str:
        return f"[+] Payment is accepted"
```

Задача 2.

Условие:

Банк предлагает ряд вкладов для физических лиц:

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

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

Решение:

```python
class Contribution:
    def __init__(self, money: int, persent: float, years: int) -> None:
        self.money = money
        self.persent = persent
        self.years = years


class FastContribution(Contribution):
    def __init__(self, money: int, persent: float, years: int) -> None:
        super().__init__(money, persent, years)

    def profit(self) -> float:
        s = self.money

        for _ in range(self.years):
            s += self.money * self.persent

        return s


class BonusContribution(Contribution):
    def __init__(self, money: int, persent: float, years: int) -> None:
        super().__init__(money, persent, years)

    def profit(self) -> float:
        for _ in range(self.years):
            self.money *= (1 + self.persent)

        return self.money


class CapitalPersentsContribution(Contribution):
    def __init__(self, money: int, persent: float, years: int) -> None:
        super().__init__(money, persent, years)

    def profit(self) -> float:
        return 0

```