Лабораторные работы 5

Присылать на почту daria.zubkova@spbpu.com

Дедлайн: 20.04.2025 до 23:59

Формат (выбирайте один из):

1. архив с задачами в формате *.py/*.ipynb, архив подписывать своей фамилией и именем
2. один файл *.txt, задачи с пометкой #задача 1, #задача 2 и т.д., файл подписывать своей фамилией и именем

Важно: входные данные в задачах запрашиваем у пользователя через input() или через файл (если это указано в задании).

Задача 1:

Создайте класс Coffee, который принимает 1 необязательный аргумент при инициализации, который отвечает за вид молока.

В этом классе реализуйте метод get_coffee(), выводящий на печать «Кофе с {x} молоком» в случае наличия аргумента x. Иначе выводить «Кофе на обычным молоке».

Входные данные:

```
кокосовым
```

Выходные данные:

```
Кофе с кокосовым молоком
```

Входные данные:

-

Выходные данные:

```
Кофе на обычном молоке
```

Задача 2:

Напишите программу, которая содержит несколько классов фигур: прямоугольник, круг, треугольник, параллелограмм, ромб, трапеция и добавьте методы для расчета площади каждой фигуры.

Пример:

```
rect = Rectangle(5, 10)
circle = Circle(3)
```

```
print(rect.area())       # 50
print(circle.area())  # ~28.27
```

Задача 3:

Создайте класс FibonacciGenerator, который генерирует последовательность чисел Фибоначчи. Используйте метод next() для получения следующего числа последовательности.

Пример:

```
fib_gen = FibonacciGenerator()
for i in range(10):
    print(fib_gen.next())
```

Задача 4:

Напишите программу, которая моделирует работу банковского счета с различными проверками. Должны быть реализованы классы и методы:

- Account — основной класс, представляющий банковский счет (по умолчанию начальная сумма = 0):
    - Методы:
    - deposit – пополнение счета на сумму. Сумма должна быть больше 0.
    - withdraw – снятие наличных. Сумма должна быть больше 0 и средств на счете должно быть достаточно.
    - transfer_c2a – перевод средств со счета. Сумма должна быть больше 0 и средств на счете должно быть достаточно.
    - transfer_a2c - перевод средств на счет. Сумма должна быть больше 0.
    - get_balance – возвращает текущий баланс счета.
    - show_transactions – возвращает список совершенных транзакций по счету.
- Transaction — хранит записи транзакций с суммой и типом (deposit, withdraw, transfer_c2a, transfer_a2c).

Пример:

```
account = Account(1000)  # Счет с ненулевой начальной суммой
account.deposit(500)      # Пополнение счета на 500
account.withdraw(200)     # Снятие 200
print(account.get_balance())    # 1300
```

Задача 5:

Напишите программу, которая моделирует бронирование билетов на мероприятия. Пользователь может добавлять событие, бронировать и отменять билеты, а также узнавать актуальную информацию по событиям и забронированным билетам. Должны быть реализованы классы и методы:

- Event — хранит информацию о мероприятии, такую как название, цена одного билета, место проведения и количество доступных билетов:
  - Методы:
  - get_event_info – возвращает информацию о событии
- Ticket — хранит информацию о том, на какое событие билет, о стоимости билета, его владельце и идентификационный номер билета:
  - Методы:
  - get_ticket_info – возвращает информацию о билете
- BookingSystem — класс для управления процессом бронирования:
  - Методы:
  - add_event – добавляет событие в систему
  - get_available_events – возвращает информацию о доступных событиях
  - check_availability – возращает доступное количество билетов на переданное событие
  - book_ticket – бронирует указанное количество билетов на нужное мероприятие на имя определенного человека
  - cancel_booking – отменяет бронь билета по его идентификационному номеру
  - get_booked_tickets – возвращает информацию о забронированных билетах

Пример:

```
booking_system = BookingSystem()

event = Event("Концерт", 500, "Городской парк", 100) # Добавили
новое событие

booking_system.book_ticket("Концерт", 2, "Иван Иванов")  #
Забронировали 2 билета на концерт

print(booking_system. check_availability("Концерт"))  # Оставшиеся
билеты = 98

print(booking_system.get_ booked_tickets())

# {

#           "Event Name": "Концерт",
```

```
#           "Owner": "Иван Иванов",
#           "Price": 500,
#           "ID": 0,
# },
# {
#           "Event Name": "Концерт",
#           "Owner": "Иван Иванов",
#           "Price": 500,
#           "ID": 1,
# }
```