

Санкт-Петербургский Национальный
Исследовательский Университет
Информационных технологий, механики и оптики

Контрольная работа №1
ООП

Выполнил: Фоминцев
Денис Русланович
Группа № 3123
Проверила: Казанова
Полина Петровна

Санкт-Петербург
2021

Цель работы:

Требуется разработать программы согласно указанным ниже заданиям. Реализация классов должна быть в отдельных модулях. Клиентский код должен импортировать нашу библиотеку и посредством объектов воспользоваться реализуемой в них функциональностью.

Ход работы:

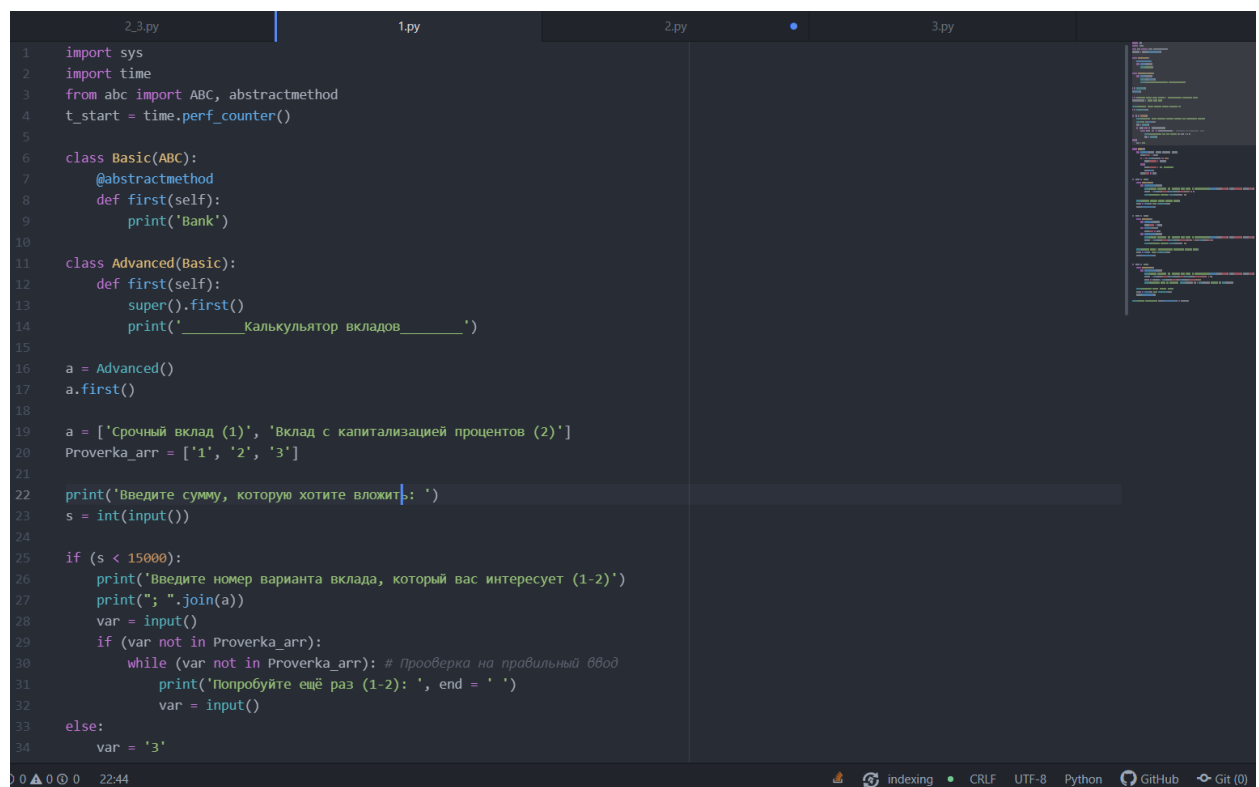
Задание №1

Постановка задачи:

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.



```
1 import sys
2 import time
3 from abc import ABC, abstractmethod
4 t_start = time.perf_counter()
5
6 class Basic(ABC):
7     @abstractmethod
8     def first(self):
9         print('Bank')
10
11 class Advanced(Basic):
12     def first(self):
13         super().first()
14         print('_____ Калькулятор вкладов _____')
15
16 a = Advanced()
17 a.first()
18
19 a = ['Срочный вклад (1)', 'Вклад с капитализацией процентов (2)']
20 Proverka_arr = ['1', '2', '3']
21
22 print('Введите сумму, которую хотите вложить: ')
23 s = int(input())
24
25 if (s < 15000):
26     print('Введите номер варианта вклада, который вас интересует (1-2)')
27     print(";", ".join(a))
28     var = input()
29     if (var not in Proverka_arr):
30         while (var not in Proverka_arr): # Проверка на правильный ввод
31             print('Попробуйте ещё раз (1-2): ', end = ' ')
32             var = input()
33     else:
34         var = '3'
```

Рисунок 1

```
2_3.py 1.py 2.py 3.py
33 else:
34     var = '3'
35
36 class bank():
37     def __init__(self, summa, percent, time):
38         self.summa = summa
39         if 1 <= int(percent) <= 100:
40             self.percent = percent
41         else:
42             self.percent = 'Не определен'
43             sys.exit()
44         self.time = time
45     def __private(self):
46         print("Это приватный метод! Деньги не вернем.")
47
48 if (var == '1'):
49     class easy(bank):
50         def display_wow(self):
51             print('Вклад размером: {}. Процент: {}%. Время {} месяц(-а/-ев)'.format(self.summa, self.percent, self.time))
52             answer = int(self.summa)*(int(self.time)*0.05 + 1)
53             print('Итоговая сумма:', round(answer, 2))
54
55         print('Выбран срочный вклад, введите срок')
56         res_1 = easy(s, '5', str(input()))
57         res_1.display_wow()
58
59
60 if (var == '2'):
61     class cap(bank):
62         def setSumma(self):
63             self.summa = summa
64         def setTime(self):
65             self.time = time
66         def display_wow(self):
```

Рисунок 2

```
2_3.py 1.py 2.py 3.py
54 res_1 = easy(s, '5', str(input()))
55 res_1.display_wow()
56
57
58 if (var == '2'):
59     class cap(bank):
60         def setSumma(self):
61             self.summa = summa
62         def setTime(self):
63             self.time = time
64         def display_wow(self):
65             print('Вклад размером: {}. Процент: {}%. Время {} месяц(-а/-ев)'.format(self.summa, self.percent, self.time))
66             answer = int(self.summa)*(int(self.percent)/100 + 1)**int(self.time)
67             print('Итоговая сумма:', round(answer, 2))
68
69         print('Выбран вклад с капитализацией процентов, введите срок')
70         res_1 = cap(s, '12', str(input()))
71         res_1.display_wow()
72
73
74 if (var == '3'):
75     class bonus(bank):
76         def display_wow(self):
77             print('Вклад размером: {}. Процент: {}%. Время {} месяц(-а/-ев)'.format(self.summa, self.percent, self.time))
78             answer = int(self.summa)*(int(self.time)*int(self.percent)/100 + 1)
79             bonus = (answer - int(self.summa))*int(self.percent)/100
80             print('Итоговая сумма (+ бонус):', round(answer, 2) + int(bonus), 'Бонус: ', int(bonus))
81
82         print('Бонусный вклад. Введите срок')
83         res_1 = bonus(s, '7', str(input()))
84         res_1.display_wow()
85
86         print('Время окончания:', time.perf_counter() - t_start)
87
```

Рисунок 3

В данном решении я изначально создаю основной класс bank() (см. рисунок 1) и три зависимых от него класса, отвечающих за три различных варианта банковского вклада (см. рисунок 2 и рисунок 3). В самом начале данного кода я создаю абстрактный класс и его наследника для представления пользователю основной информации о программе.

Результат работы программы:

1.

```
Bank
    Калькулятор вкладов
Введите сумму, которую хотите вложить:
15000
Бонусный вклад. Введите срок
45
Вклад размером: 15000. Процент: 7%. Время 45 месяц(-а/-ев)
Итоговая сумма (+ бонус): 65557.0 Бонус: 3307
Время окончания: 8.3865821
```

2.

```
Bank
    Калькулятор вкладов
Введите сумму, которую хотите вложить:
10000
Введите номер варианта вклада, который вас интересует (1-2)
Срочный вклад (1); Вклад с капитализацией процентов (2)
1
Выбран срочный вклад, введите срок
6
Вклад размером: 10000. Процент: 5%. Время 6 месяц(-а/-ев)
Итоговая сумма: 13000.0
Время окончания: 9.6338869
```

3.

```
Bank
    Калькулятор вкладов
Введите сумму, которую хотите вложить:
1000
Введите номер варианта вклада, который вас интересует (1-2)
Срочный вклад (1); Вклад с капитализацией процентов (2)
daw
Попробуйте ещё раз (1-2): awdaw
Попробуйте ещё раз (1-2): dadwa
Попробуйте ещё раз (1-2): 0
Попробуйте ещё раз (1-2): 2
Выбран вклад с капитализацией процентов, введите срок
7
Вклад размером: 1000. Процент: 12%. Время 7 месяц(-а/-ев)
Итоговая сумма: 2210.68
Время окончания: 14.2542311
```

Задание №2

Постановка задачи:

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.

```
2_3.py 1.py 2.py 3.py
1 import sys
2 import time
3 import random
4 from abc import ABC, abstractmethod
5 from random import randint
6 t_start = time.perf_counter()
7
8
9 print('Бой борцов')
10 print()
11 health = 100
12 class Basic(ABC):
13     @abstractmethod
14     def first(self):
15         print("Начало боя.")
16
17 class Advanced(Basic):
18     def first(self):
19         super().first()
20         print("Здоровье каждого бойца: 100")
21
22 a = Advanced()
23 a.first()
24 health_1 = health_2 = 100
25 class fight():
26     def __init__(self, health_1, health_2):
27         self.health_1 = health_1
28         self.health_2 = health_2
29
30
31
32 class Fighter_1(fight):
33     def display_wow(self):
34         if (int(self.health_1) >= 0):
```

Рисунок 4

```
2_3.py 1.py 2.py 3.py
32 class Fighter_1(fight):
33     def display_wow(self):
34         if (int(self.health_1) >= 0):
35             print('А. Атаковал второй боец. Осталось здоровья у первого бойца: {}'.format(self.health_1))
36         else:
37             print('Победил второй боец!')
38             print()
39             print('Время окончания:', time.perf_counter() - t_start)
40             sys.exit()
41
42
43 class Fighter_2(fight):
44     def display_wow(self):
45         if (int(self.health_2) >= 0):
46             print('В. Атаковал первый боец. Осталось здоровья у второго бойца: {}'.format(self.health_2))
47         else:
48             print('Победил первый боец!')
49             print()
50             print('Время окончания:', time.perf_counter() - t_start)
51             sys.exit()
52
53
54 a = ['A', 'B']
55 while (health_1 != 0) or (health_2 != 0):
56     res = random.choice(a)
57     if (res == 'A'):
58         health_1 -= 20
59         res_1 = Fighter_1(health_1, health_2)
60         res_1.display_wow()
61     if (res == 'B'):
62         health_2 -= 20
63         res_2 = Fighter_2(health_1, health_2)
64         res_2.display_wow()
65
```

Рисунок 5

В самом начале создаём абстрактный класс и его наследника для представления пользователю основной информации о программе (см. рисунок 4). Далее следует класс родитель и два наследника (см. рисунок 4 и рисунок 5). Бой борцов происходит случайно, благодаря одноименной функции. Весь бой происходит в цикле while (см. рисунок 5).

Результат работы программы:

```
Бой борцов
Начало боя.
Здоровье каждого бойца: 100
В. Атаковал первый боец. Осталось здоровья у второго бойца: 80
А. Атаковал второй боец. Осталось здоровья у первого бойца: 80
А. Атаковал второй боец. Осталось здоровья у первого бойца: 60
В. Атаковал первый боец. Осталось здоровья у второго бойца: 60
А. Атаковал второй боец. Осталось здоровья у первого бойца: 40
В. Атаковал первый боец. Осталось здоровья у второго бойца: 40
А. Атаковал второй боец. Осталось здоровья у первого бойца: 20
В. Атаковал первый боец. Осталось здоровья у второго бойца: 20
А. Атаковал второй боец. Осталось здоровья у первого бойца: 0
Победил второй боец!
1. Время окончания: 0.003902700000000002
```

Задание №3

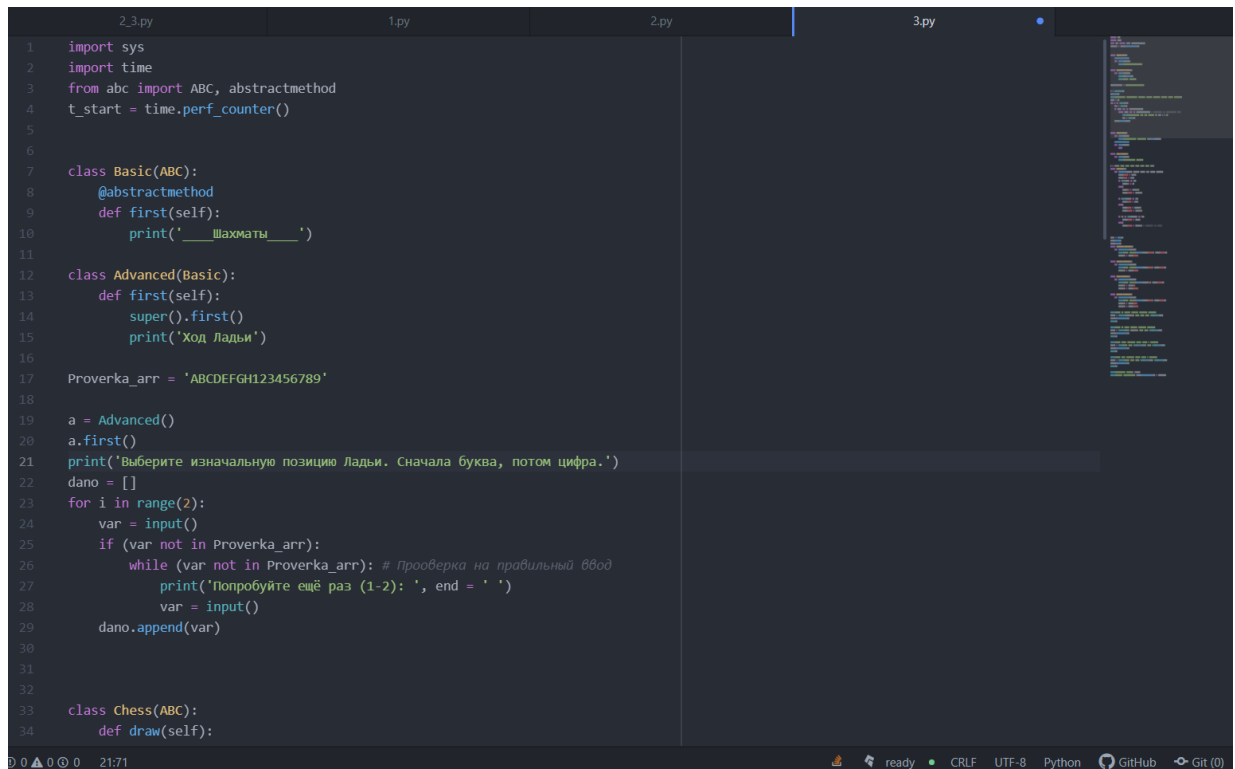
Постановка задачи:

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.



```
2_3.py 1.py 2.py 3.py
1 import sys
2 import time
3 from abc import ABC, abstractmethod
4 t_start = time.perf_counter()
5
6
7 class Basic(ABC):
8     @abstractmethod
9     def first(self):
10         print('Шахматы')
11
12 class Advanced(Basic):
13     def first(self):
14         super().first()
15         print('Ход Ладьи')
16
17 Proverka_arr = 'ABCDEFGH123456789'
18
19 a = Advanced()
20 a.first()
21 print('Выберите начальную позицию Ладьи. Сначала буква, потом цифра.')
22 dano = []
23 for i in range(2):
24     var = input()
25     if (var not in Proverka_arr):
26         while (var not in Proverka_arr): # Проверка на правильный ввод
27             print('Попробуйте ещё раз (1-2): ', end = ' ')
28             var = input()
29     dano.append(var)
30
31
32
33 class Chess(ABC):
34     def draw(self):
```

Рисунок 6

```
2,3.py 1.py 2.py 3.py
32
33 class Chess(ABC):
34     def draw(self):
35         print("Изначальная позиция", "".join(dano))
36     @abstractmethod
37     def move(self):
38         pass
39
40 class rook(Chess):
41     def move(self):
42         print("Функционал Ладьи")
43
44 a = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
45 class St_rook():
46     def __init__(self, right, left, up, down, count):
47         self.right = right
48         self.left = left
49         if (str(up) in a):
50             self.up = up
51         else:
52             self.up = dano[0]
53             self.count = dano[1]
54
55         if (str(down) in a):
56             self.down = down
57         else:
58             self.down = dano[0]
59             self.count = dano[1]
60
61         if (1 <= int(count) <= 8):
62             self.count = count
63         else:
64             self.count = dano[1] # Остается на месте
65
66
0 0 ▲ 0 0 21:71 ready CRLF UTF-8 Python GitHub Git (0)
```

Рисунок 7

```
2,3.py 1.py 2.py 3.py
67
68 Step = rook()
69 Step.draw()
70 Step.move()
71 class R_right(St_rook):
72     def display_wow(self):
73         print('Ход {}'.format(self.right, self.count))
74         dano[1] = self.count
75
76 class R_left(St_rook):
77     def display_wow(self):
78         print('Ход {}'.format(self.left, self.count))
79         dano[1] = self.count
80
81 class R_up(St_rook):
82     def display_wow(self):
83         print('Ход {}'.format(self.up, self.count))
84         dano[0] = self.up
85         dano[1] = self.count
86
87 class R_down(St_rook):
88     def display_wow(self):
89         print('Ход {}'.format(self.down, self.count))
90         dano[0] = self.down
91         dano[1] = self.count
92
93 print('Ход по правой стороне (введите число)')
94 res_1 = R_right(dano[0], '0', '0', '0', str(input()))
95 res_1.display_wow()
96 print()
97
98 print('Ход по левой стороне (введите число)')
99 res_1 = R_left('0', dano[0], '0', '0', str(input()))
100 res_1.display_wow()
0 0 ▲ 0 0 21:71 ready CRLF UTF-8 Python GitHub Git (0)
```

Рисунок 8

```
2.3.py 1.py 2.py 3.py
82 def display_wow(self):
83     print('Ход {}{}.'.format(self.up, self.count))
84     dano[0] = self.up
85     dano[1] = self.count
86
87 class R_down(St_rook):
88     def display_wow(self):
89         print('Ход {}{}.'.format(self.down, self.count))
90         dano[0] = self.down
91         dano[1] = self.count
92
93 print('Ход по правой стороне (введите число)')
94 res_1 = R_right(dano[0], '0', '0', '0', str(input()))
95 res_1.display_wow()
96 print()
97
98 print('Ход по левой стороне (введите число)')
99 res_1 = R_left('0', dano[0], '0', '0', str(input()))
100 res_1.display_wow()
101 print()
102
103 print('Ход вверх (введите букву (А-Н) и число)')
104 res_1 = R_up('0', '0', str(input()), '0', str(input()))
105 res_1.display_wow()
106 print()
107
108 print('Ход вниз (введите букву (А-Н) и число)')
109 res_1 = R_down('0', '0', '0', str(input()), str(input()))
110 res_1.display_wow()
111 print()
112
113 print('Конечная точка', "dano")
114 print('Время окончания:', time.perf_counter() - t_start)
115
```

Рисунок 9

В самом начале данного кода я создаю абстрактный класс и его наследника для представления пользователю основной информации о программе (см. рисунок 6). Далее создаем основной класс (St_rook()), который будет отвечать за функционал Ладьи, в нем прописываем условия, по которым будет проводиться ввод данных. Если данные будут введены неверно, то позиция ладьи, которая была зафиксирована в самом начале останется для следующего хода. (см. рисунок 7). Создаем 4 класса наследника (см. рисунок 8 и рисунок 9) для основной работы Ладьи и смены хода. Вывод данных см. на рисунке 9.

Результат работы программы:

```
Шахматы
Ход Ладьи
Выберите изначальную позицию Ладьи. Сначала буква, потом цифра.
G
7
Изначальная позиция G7
Функционал Ладьи
Ход по правой стороне (введите число)
8
Ход G8.

Ход по левой стороне (введите число)
5
Ход G5.

Ход вверх (введите букву (А-Н) и число)
R
9
Ход G5.

Ход вниз (введите букву (А-Н) и число)
D
4
Ход D4.

Конечная точка D 4
Время окончания: 38.3362788
```


Вывод:

В данной работе мы создавали программы по заданным параметрам, используя основные принципы работы ООП.