

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Информационный технологии»
Тема: Парадигмы программирования

Студент гр. 3344		Тукалкин. В.А.
Преподаватель		Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Ознакомиться с парадигмами программирования в языке программирования Python.

Задание.

Вариант 2.

Базовый класс - персонаж Character:

class Character:

Поля объекта класс Character:

- 1) Пол (значение может быть одной из строк: 'm', 'w')
- 2) Возраст (целое положительное число)
- 3) Рост (в сантиметрах, целое положительное число)
- 4) Вес (в кг, целое положительное число)

При создании экземпляра класса Character необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

Воин - Warrior:

class Warrior: #Наследуется от класса Character

Поля объекта класс Warrior:

- 1) Пол (значение может быть одной из строк: 'm', 'w')
- 2) Возраст (целое положительное число)
- 3) Рост (в сантиметрах, целое положительное число)
- 4) Вес (в кг, целое положительное число)
- 5) Запас сил (целое положительное число)
- 6) Физический урон (целое положительное число)
- 7) Количество брони (неотрицательное число)
- 8) При создании экземпляра класса Warrior необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

В данном классе необходимо реализовать следующие методы:

Метод __str__():

Преобразование к строке вида: Warrior: Пол <пол>, возраст <возраст>, рост <рост>, вес <вес>, запас сил <запас сил>, физический урон <физический урон>, броня <количество брони>.

Метод `__eq__()`:

Метод возвращает True, если два объекта класса равны и False иначе. Два объекта типа Warrior равны, если равны их урон, запас сил и броня.

Маг - Magician:

`class Magician: #Наследуется от класса Character`

Поля объекта класс Magician:

- 1) Пол (значение может быть одной из строк: 'm', 'w')
- 2) Возраст (целое положительное число)
- 3) Рост (в сантиметрах, целое положительное число)
- 4) Вес (в кг, целое положительное число)
- 5) Запас маны (целое положительное число)
- 6) Магический урон (целое положительное число)

При создании экземпляра класса Magician необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

В данном классе необходимо реализовать следующие методы:

Метод `__str__()`:

Преобразование к строке вида: Magician: Пол <пол>, возраст <возраст>, рост <рост>, вес <вес>, запас маны <запас маны>, магический урон <магический урон>.

Метод `__damage__()`:

Метод возвращает значение магического урона, который может нанести маг, если потратит сразу весь запас маны (умножение магического урона на запас маны).

Лучник - Archer:

`class Archer: #Наследуется от класса Character`

Поля объекта класс Archer:

- 1) Пол (значение может быть одной из строк: m (man), w(woman))
- 2) Возраст (целое положительное число)
- 3) Рост (в сантиметрах, целое положительное число)

- 4) Вес (в кг, целое положительное число)
- 5) Запас сил (целое положительное число)
- 6) Физический урон (целое положительное число)
- 7) Дальность атаки (целое положительное число)

При создании экземпляра класса Archer необходимо убедиться, что переданные в конструктор параметры удовлетворяют требованиям, иначе выбросить исключение ValueError с текстом 'Invalid value'.

В данном классе необходимо реализовать следующие методы:

Метод `__str__()`:

Преобразование к строке вида: Archer: Пол <пол>, возраст <возраст>, рост <рост>, вес <вес>, запас сил <запас сил>, физический урон <физический урон>, дальность атаки <дальность атаки>.

Метод `__eq__()`:

Метод возвращает True, если два объекта класса равны и False иначе. Два объекта типа Archer равны, если равны их урон, запас сил и дальность атаки.

Необходимо определить список list для работы с персонажами:

Воины:

class WarriorList – список воинов - наследуется от класса list.

Конструктор:

- 1) Вызвать конструктор базового класса.
- 2) Передать в конструктор строку name и присвоить её полю name созданного объекта.

Необходимо реализовать следующие методы:

Метод `append(p_object)`: Переопределение метода `append()` списка. В случае, если `p_object` - Warrior, элемент добавляется в список, иначе выбрасывается исключение TypeError с текстом: Invalid type <тип_объекта p_object>.

Метод `print_count()`: Вывести количество воинов.

Маги:

class MagicianList – список магов - наследуется от класса list.

Конструктор:

- 1) Вызвать конструктор базового класса.
- 2) Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

Метод `extend(iterable)`: Переопределение метода `extend()` списка. В случае, если элемент `iterable` - объект класса `Magician`, этот элемент добавляется в список, иначе не добавляется.

Метод `print_damage()`: Вывести общий урон всех магов.

Лучники:

`class ArcherList` – список лучников - наследуется от класса `list`.

Конструктор:

- 1) Вызвать конструктор базового класса.
- 2) Передать в конструктор строку name и присвоить её полю name созданного объекта

Необходимо реализовать следующие методы:

Метод `append(p_object)`: Переопределение метода `append()` списка. В случае, если `p_object` - `Archer`, элемент добавляется в список, иначе выбрасывается исключение `TypeError` с текстом: `Invalid type <тип_объекта p_object>`

Метод `print_count()`: Вывести количество лучников мужского пола.

Выполнение работы

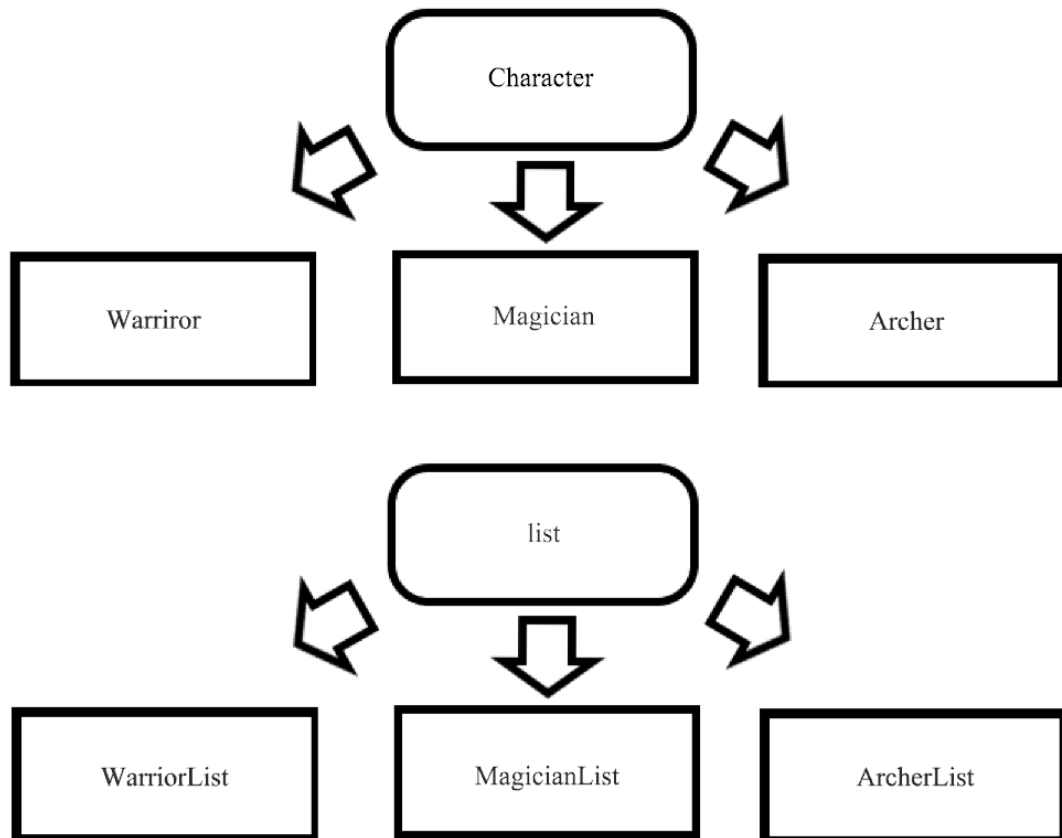


Рисунок 1 – Изображение иерархии классов

- 1) Методы классов, унаследованных у Character:
 - `__init__()` – Принимает параметры и проверяет их корректность
 - `__str__()` – Преобразовывает данный в строку и возвращает её
 - `__eq__()` – Сравнивает два объекта
 - `__damage__()` – Возвращает урон мага
- 2) Методы классов, унаследованных у list:
 - `__init__()` – Принимает параметры и проверяет их корректность
 - `append()` – Добавляет элемент в конец списка
 - `print_count()` – Возвращает количество воинов/лучников
 - `extend()` – Добавляет элемент в конец списка после проверки
 - `print_damage()` – Выводит сумму урона всех магов в списке
- 3) Метод `__str__()` будет применяться при вызове `str`.
Метод `print_damage()` будет применяться, когда вызван `Magicianlist`.

- 4) Методы будут работать, потому что они являются теми же методами, что и в родительских классах, отличие только в добавлении проверки, например: `append()` в классе `Warriorlist` отличается только проверкой класса `Warrior`.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	<pre> character=Character('m', 20, 180, 70) #персонаж print(character.gender, character.age, character.height, character.weight) warrior1 = Warrior('m', 20, 180, 70, 50, 100, 30) #воин warrior2 = Warrior('m', 20, 180, 70, 50, 100, 30) print(warrior1.gender, warrior1.age, warrior1.height, warrior1.weight, warrior1.forces, warrior1.physical_dama ge, warrior1.armor) print(warrior1.__str__()) print(warrior1.__eq__(w arrior2)) mag1 = Magician('m', 20, 180, 70, 60, 110) #маг </pre>	<pre> m 20 180 70 m 20 180 70 50 100 30 Warrior: Пол m, возраст 20, рост 180, вес 70, запас сил 50, физический урон 100, броня 30. True m 20 180 70 60 110 Magician: Пол m, возраст 20, рост 180, вес 70, запас маны 60, магический урон 110. 6600 m 20 180 70 60 95 50 Archer: Пол m, возраст 20, рост 180, вес 70, запас сил 60, физический урон 95, дальность атаки 50. True 2 220 2 </pre>	Верный ответ

```

mag2 = Magician('m', 20,
180, 70, 60, 110)
print(mag1.gender,
mag1.age, mag1.height,
mag1.weight,
mag1.mana,
mag1.magic_damage)
print(mag1.__str__())
print(mag1.__damage__(
))
archer1 = Archer('m', 20,
180, 70, 60, 95, 50)
#лучник
archer2 = Archer('m', 20,
180, 70, 60, 95, 50)
print(archer1.gender,
archer1.age,
archer1.height,
archer1.weight,
archer1.forces,
archer1.physical_damage, archer1.attack_range)
print(archer1.__str__())
print(archer1.__eq__(archer2))
warrior_list =
WarriorList(Warrior)
#СПИСОК ВОИНОВ

```

```
warrior_list.append(warrior1)
warrior_list.append(warrior2)
warrior_list.print_count()
mag_list = MagicianList(Magician)
#список магов
mag_list.extend([mag1, mag2])
mag_list.print_damage()
archer_list = ArcherList(Archer)
#список лучников
archer_list.append(archer1)
archer_list.append(archer2)
archer_list.print_count()
```

Выводы

Была изучена императивная парадигма (ООП), на примере программы, выполняющей с операции с классами в Python.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class Character:
    def __init__(self,gender,age,height,weight):
        if (gender in ["m","w"]) and isinstance(age,int) and age>0
and height>0 and isinstance(height,int) and weight>0 and
isinstance(weight,int):
            self.gender=gender #Пол (значение может быть одной из
строк: 'm', 'w')
            self.age=age #Возраст (целое положительное число)
            self.height=height #Рост (в сантиметрах, целое
положительное число)
            self.weight=weight #Вес (в кг, целое положительное
число)
        else: raise ValueError("Invalid value")

class Warrior(Character):
    def
__init__(self,gender,age,height,weight,forces,physical_damage,armor):
        super().__init__(gender,age,height,weight)
        if isinstance(forces,int) and forces>0 and
physical_damage>0 and isinstance(physical_damage,int) and
isinstance(armor,int) and armor>0:
            self.forces=forces #Запас сил (целое
положительное число)
            self.physical_damage=physical_damage #Физический урон
(целое положительное число)
            self.armor=armor #Количество брони
(неотрицательное число)
        else: raise ValueError("Invalid value")

    def __str__(self):
        return f"Warrior: Пол {self.gender}, возраст {self.age},
рост {self.height}, вес {self.weight}, запас сил {self.forces},
физический урон {self.physical_damage}, броня {self.armor}."

    def __eq__(self,other):
        if isinstance(self,Warrior) and isinstance(other,Warrior)
and self.physical_damage==other.physical_damage and
self.armor==other.armor and self.forces==other.forces:
            return True
        else: return False

class Magician(Character):
    def
__init__(self,gender,age,height,weight,mana,magic_damage):
        super().__init__(gender,age,height,weight)
        if magic_damage>0 and isinstance(magic_damage,int) and
isinstance(mana,int) and mana>0:
            self.magic_damage=magic_damage #Магический урон (целое
положительное число)
```

```

        self.mana=mana                                #Запас маны (целое
положительное число)
        else: raise ValueError("Invalid value")

    def __str__(self):
        return f"Magician: Пол {self.gender}, возраст {self.age},
рост {self.height}, вес {self.weight}, запас маны {self.mana},
магический урон {self.magic_damage}."

    def __damage__(self):
        return self.mana*self.magic_damage

class Archer(Character):
    def
__init__(self,gender,age,height,weight,forces,physical_damage,attack_r
ange):
        super().__init__(gender,age,height,weight)
        if isinstance(forces,int) and forces>0 and
physical_damage>0 and isinstance(physical_damage,int) and
isinstance(attack_range,int) and attack_range>0:
            self.forces=forces                    #Запас сил (целое
положительное число)
            self.physical_damage=physical_damage #Физический урон
(целое положительное число)
            self.attack_range=attack_range        #Дальность атаки
(целое положительное число)
        else: raise ValueError("Invalid value")

    def __str__(self):
        return f"Archer: Пол {self.gender}, возраст {self.age},
рост {self.height}, вес {self.weight}, запас сил {self.forces},
физический урон {self.physical_damage}, дальность атаки
{self.attack_range}."

    def __eq__(self, other):
        if isinstance(self, Archer) and isinstance(other, Archer)
and self.physical_damage == other.physical_damage and self.attack_range
== other.attack_range and self.forces == other.forces:
            return True
        else: return False

class WarriorList(list):
    '''Конструктор:
        1. Вызвать конструктор базового класса.
        2. Передать в конструктор строку name и присвоить её полю
name созданного объекта'''
    def __init__(self,name):
        super().__init__()
        self.name=name

    def append(self,p_object):
        if isinstance(p_object,Warrior): super().append(p_object)
        else: raise TypeError(f"Invalid type {type(p_object)}")

    def print_count(self):

```

```

        print(self.count(warrior1))

class MagicianList(list):
    def __init__(self, name):
        super().__init__()
        self.name = name

    def extend(self, iterable):
        for i in iterable:
            if isinstance(i, Magician):
                super().append(i)
            else:
                continue

    def print_damage(self):
        count = 0
        for i in self:
            count += i.magic_damage
        print(count)

class ArcherList(list):
    def __init__(self, name):
        super().__init__()
        self.name = name

    def append(self, p_object):
        if isinstance(p_object, Archer):
            super().append(p_object)
        else:
            raise TypeError(f"Invalid type {type(p_object)}")

    def print_count(self):
        count = 0
        for i in self:
            if i.gender == "m":
                count += 1
        print(count)

```