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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Информатика»
Тема: Парадигмы программирования

Студент гр. 3342 _____ Песчатский С. Д.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург
2024

Цель работы

Изучить основы ООП, работу с классами и исключениями и их реализацию на языке Python. С их помощью написать программу, создающие объекты этих классов.

Задание

Базовый класс - персонаж Character:

class Character:

- Поля объекта класс Character:
- Пол (значение может быть одной из строк: 'm', 'w')
- Возраст (целое положительное число)
- Рост (в сантиметрах, целое положительное число)
- 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:

- Пол (значение может быть одной из строк: 'm', 'w')
- Возраст (целое положительное число)
- Рост (в сантиметрах, целое положительное число)
- Вес (в кг, целое положительное число)
- Запас сил (целое положительное число)
- Физический урон (целое положительное число)
- Количество брони (неотрицательное число)

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:

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

- Пол (значение может быть одной из строк: m (man), w(woman))
- Возраст (целое положительное число)
- Рост (в сантиметрах, целое положительное число)
- Вес (в кг, целое положительное число)
- Запас сил (целое положительное число)
- Физический урон (целое положительное число)
- 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:

Вызвать конструктор базового класса.

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:

Вызвать конструктор базового класса.

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:

Вызвать конструктор базового класса.

Передать в конструктор строку name и присвоить её полю name созданного объекта

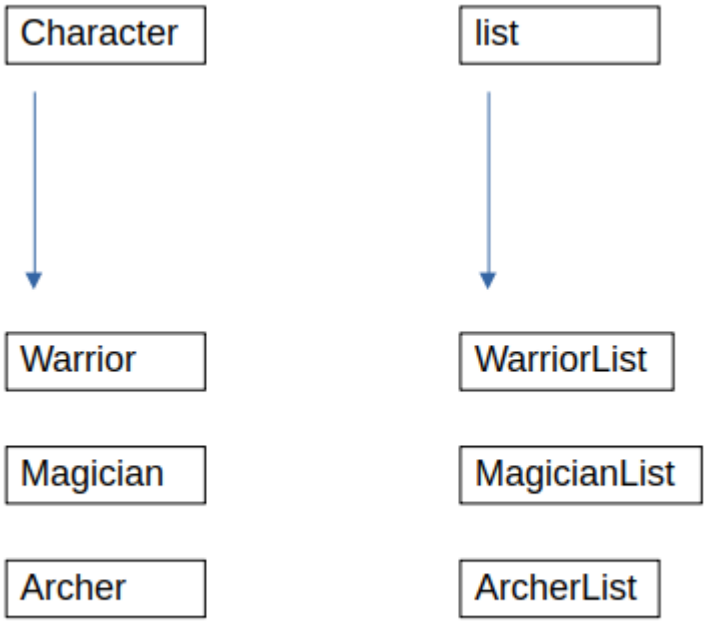
Необходимо реализовать следующие методы:

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

Покажем наследование классов.



Сначала создаётся класс Character с базовыми полями. Класс Warrior наследуется от Character и определяет методы __str__(), выводящий класс в строковом виде, и __eq__(), сравнивающий два объекта класса между собой.

Класс Magician наследуется от базового класса Character и определяет методы __str__(), выводящий класс в строковом виде, и __damage__(), возвращающий урон, который может нанести маг, если использует сразу всю ману.

Класс Archer наследуется от класса Character и определяет методы __str__(), выводящий класс в строковом виде, и __eq__(), сравнивающий два объекта класса между собой.

Класс WarriorList наследуется от базового класса list и переопределяет метод append(), позволяющий добавить в список объект класса Warrior. Также класс определяет метод print_count(), выводящий количество элементов класса Warrior

Класс MagicianList наследуется от базового класса list и переопределяет метод extend(), позволяющий добавить в список объект класса Magician. Также определяет метод print_damage(), выводящий общий урон всех магов в списке.

Класс ArcherList наследуется от базового класса list и переопределяет метод append(), позволяющий добавить в список объект класса Archer. Также класс определяет метод print_count(), выводящий количество элементов класса Archer, у которых поле gender имеет значение “m”

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные

1.	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_damage, warrior1.armor)
	print(warrior1.__str__())
	print(warrior1.__eq__(warrior2))
	
	mag1 = Magician('m', 20, 180, 70, 60, 110) #маг
	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)
		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

	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 разработана программа, использующая ООП, переопределены некоторые методы у дочерних классов, а так же была реализована обработка исключений.

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):  # Наследуется от класса Character
    def __init__(self, gender, age, height, weight, forces, physical_damage, armor):
        Character.__init__(self, gender, age, height, weight)
        if isinstance(forces, int) and isinstance(physical_damage, int) and isinstance(armor, int) and forces > 0 and physical_damage > 0 and armor > 0:
            self.forces = forces
            self.physical_damage = physical_damage
            self.armor = armor
        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 self.physical_damage == other.physical_damage and self.forces == other.forces and self.armor == other.armor:
            return True
        else:
            return False


class Magician(Character):  # Наследуется от класса Character
```

```
        def __init__(self, gender, age, height, weight, mana,
magic_damage):
            Character.__init__(self, gender, age, height, weight)
            if isinstance(mana, int) and isinstance(magic_damage, int)
and mana > 0 and magic_damage > 0:
                self.mana = mana
                self.magic_damage = magic_damage
            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):  # Наследуется от класса
Character
        def __init__(self, gender, age, height, weight, forces,
physical_damage, attack_range):
            Character.__init__(self, gender, age, height, weight)
            if isinstance(forces, int) and isinstance(physical_damage,
int) and isinstance(attack_range,int) and forces > 0 and physical_damage >
0 and attack_range > 0:
                self.forces = forces
                self.physical_damage = physical_damage
                self.attack_range = attack_range
            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 self.physical_damage == other.physical_damage and
self.forces == other.forces and self.attack_range == other.attack_range:
                return True
            else:
                return False  # Наследуется от класса
Character


    class WarriorList(list):
        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(len(self))


    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):
            damage = 0
            for i in self:
                damage = i.magic_damage + damage
            print(damage)


    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):
            amount = 0
            for i in self:
                if i.gender == "m":
                    amount = amount + 1
            print(amount)
```