**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Информатика»**

Тема: Парадигмы программирования

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Романов Е.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Изучение исключений и классов в языке программирования Python. Реализация программы, создающей экземпляры описанных 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

Работа состоит из описания семи классов.

Первый класс Character является родительским классом для классов Warrior, Magician и Archer. Класс имеет одну функцию \_\_init\_\_, которая добавляет в экземпляр объекта соответствующие поля и проверяет их на соответствие требованиям задания.

Второй класс Warrior имеет 3 метода:

* \_\_init\_\_ добавляет новые поля в экземпляр класса, проверяет их на корректность
* \_\_str\_\_ представляет экземпляр класса в строковом виде
* \_\_eq\_\_ сравнивает два экземпляра класса по полям physical\_damage, forces, armor

Третий класс Magician имеет 3 метода:

* \_\_init\_\_ добавляет новые поля в экземпляр класса, проверяет их на корректность
* \_\_str\_\_ представляет экземпляр в строковом виде
* \_\_damage\_\_ возвращает значение магического урона экземпляра

Четвёртый класс Archer имеет 3 метода:

* \_\_init\_\_ добавляет новые поля в экземпляр класса, проверяет их на корректность
* \_\_str\_\_ представляет экземпляр в строковом виде
* \_\_eq\_\_ сравнивает два экземпляра класса по полям physical\_damage, forces, attack\_range

Пятый класс WarriorList – наследник класса list, имеет 3 метода:

* \_\_init\_\_ добавляет поле name экземпляру класса
* append переопределённый метод родительского класса, добавляющий в объект только элементы типа Warrior, проверяет соответствие добавляемого элемента типа Warrior
* print\_count выводит длину списка

Шестой класс MagicianList – наследник класса list, имеет 3 метода:

* \_\_init\_\_ добавляет поле name экземпляру класса
* extend переопределение метода extend родительского класса list, добавляет только элементы соответствующие типу Magician
* print\_damage выводит сумму значений магического урона всех элементов списка

Седьмой класс ArcherList – наследник класса list, имеет 3 метода:

* \_\_init\_\_ добавляет поле name экземпляру класса
* append переопределение метода родительского класса, добавляет в список только элементы, соответствующие типу Archer
* print\_count вывод количества элементов со значением “m” поля gender

Наследование классов представлено на рисунке 1.

list

Character

MagicianList

WarriorList

ArcherList

Warrior

Archer

Magician

Рисунок 1 – Схема наследования классов в программе

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Комментарий |
| 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)  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() | 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 | Вывод верный |

## Выводы

В ходе работы была написана программа, содержащая описание классов и их методов. Было добавлено наследования классов, и исключения.

# **Приложение А Исходный код программы**

Название файла: main.py

class Character:

def \_\_init\_\_(self, gender, age, height, weight):

if (gender in 'mw' and all(x > 0 for x in [age, height, weight]) and all(isinstance(x, int) for x in [age, weight, height])):

self.gender = gender

self.age = age

self.height = height

self.weight = weight

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 (all(x>0 for x in [forces, physical\_damage, armor]) and all(isinstance(x, int) for x in [forces, physical\_damage, armor])):

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):

return all(getattr(self, field) == getattr(other, field) for field in ['physical\_damage', 'forces', 'armor'])

class Magician(Character):

def \_\_init\_\_(self, gender, age, height, weight, mana, magic\_damage):

super().\_\_init\_\_(gender, age, height, weight)

if (all(x>0 for x in [mana, magic\_damage])) and all(isinstance(x, int) for x in [mana, magic\_damage]):

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):

def \_\_init\_\_(self, gender, age, height, weight, forces, physical\_damage, attack\_range):

super().\_\_init\_\_(gender, age, height, weight)

if (all(x>0 for x in [forces, physical\_damage, attack\_range]) and all(isinstance(x, int) for x in [forces, physical\_damage, attack\_range])):

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):

return all(getattr(self, field) == getattr(other, field) for field in ['physical\_damage', 'forces', 'attack\_range'])

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 x in iterable:

if isinstance(x, Magician):

super().append(x)

def print\_damage(self):

print(sum(x.magic\_damage for x in self))

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 x in self:

if x.gender == 'm': count+=1

print(count)