**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Информационные технологии»**

Тема: Парадигмы программирования

|  |  |  |
| --- | --- | --- |
| Студент гр. 3344 |  | Пачев Д.К. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

**Цель работы**

Написать программу на языке Python, которая определяет набор классов для представления различных типов персонажей. Развить навыки написания кода с применением ООП.

**Задание**

**Вариант 2.**

**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:

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**

1. Иерархия классов:

* PositiveNumber
* StringValue
* Character
  + Warrior
  + Magician
  + Archer
* list
  + WarriorList
  + MagicianList
  + ArcherList

PositiveNumber – класс, содержащий статический метод для проверки числовых значений на положительность и целочисленность. Реализует дескриптор для установки и получения целочисленных значений.

StringValue – класс, содержащий статический метод для проверки строковых значений на соответствие допустимым значениям. Реализует дескриптор для установки и получения строковых значений.

Character – базовый класс для персонажей, содержит атрибуты пола, возраста, роста, веса и атрибуты родителя. Использует дескрипторы для установки и получения значений.

Warrior – класс-наследник от Character, представляющий воина. Содержит атрибуты силы, физического урона, брони и атрибуты родителя. Реализует методы для сравнения и вывода информации о воине.

Magician – класс-наследник от Character, представляющий мага. Содержит атрибуты маны, магического урона и атрибуты родителя. Реализует метод для расчета урона.

Archer – Класс-наследник от Character, представляющий лучника. Содержит атрибуты силы, физического урона, дальности атаки и атрибуты родителя. Реализует методы для сравнения и вывода информации о лучнике.

WarriorList, MagicianList, ArcherList – списочные классы, предназначенные для хранения объектов соответствующих классов, наследуются от класса list. Реализуют методы для добавления элементов, вывода информации.

2) Методы класса object, которые переопределены:

* + \_\_init\_\_
  + \_\_str\_\_
  + \_\_eq\_\_

Дополнительные методы, которые переопределены:

* + append в классе WarriorList и ArcherList
  + extend в классе MagicianList

3) Метод \_\_str\_\_() будет использоваться при представлении объекта в виде строки, например при применения к объекту методов str() и print().

4) Переопределенные методы класса list для созданных списков будут работать, так как это такие же методы, только с дополнительной проверкой. Например, метод append класса ArcherList делает то же самое, что и стандартный метод append, просто проверяет перед этим, принадлежит ли объект классу Archer.

**Тестирование**

Результаты тестирования представлены в Таблице 1

Таблица 1 - Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | character = Character('m', 20, 180, 70)  print(character.gender, character.age, character.height, character.weight)  warrior1 = Warrior('m', 20, 180, 70, 50, 100, 30)  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)) | 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 | верно |
| 2. | warrior1 = Warrior('m', 20, 180, 70, 50, 100, 30)  warrior2 = Warrior('m', 20, 180, 70, 50, 100, 30)  warrior\_list = WarriorList(Warrior)  warrior\_list.append(warrior1)  warrior\_list.append(warrior2)  warrior\_list.print\_count() | 2 | верно |

**Выводы**

В ходе выполнения лабораторной работы была разработана программа на языке Python для представления различных типов персонажей в виде классов, и развиты навыки написания кода, используя принципы ООП.

# ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

*class* PositiveNumber:  
 @staticmethod  
 *def* check\_number(number):  
 *if not* (*isinstance*(number, *int*) *and* number > 0):  
 *raise ValueError*('Invalid value')  
  
 *def \_\_set\_name\_\_*(*self*, owner, name):  
 *self*.name = '\_' + name  
  
 *def \_\_get\_\_*(*self*, instance, owner):  
 *return getattr*(instance, *self*.name)  
  
 *def \_\_set\_\_*(*self*, instance, value):  
 *self*.check\_number(value)  
 *setattr*(instance, *self*.name, value)  
  
  
*class* StringValue:  
 @staticmethod  
 *def* check\_string(string):  
 *if not* (*isinstance*(string, *str*) *and* string *in* 'mw'):  
 *raise ValueError*('Invalid value')  
  
 *def \_\_set\_name\_\_*(*self*, owner, name):  
 *self*.name = '\_' + name  
  
 *def \_\_get\_\_*(*self*, instance, owner):  
 *return getattr*(instance, *self*.name)  
  
 *def \_\_set\_\_*(*self*, instance, value):  
 *self*.check\_string(value)  
 *setattr*(instance, *self*.name, value)  
  
  
*class* Character:  
 gender = StringValue()  
 age = PositiveNumber()  
 height = PositiveNumber()  
 weight = PositiveNumber()  
  
 *def \_\_init\_\_*(*self*, gender, age, height, weight):  
 *self*.gender = gender  
 *self*.age = age  
 *self*.height = height  
 *self*.weight = weight  
  
  
*class* Warrior(Character):  
 forces = PositiveNumber()  
 physical\_damage = PositiveNumber()  
 armor = PositiveNumber()  
  
 *def \_\_init\_\_*(*self*, gender, age, height, weight, forces, physical\_damage, armor):  
 *super*().\_\_init\_\_(gender, age, height, weight)  
 *self*.forces = forces  
 *self*.physical\_damage = physical\_damage  
 *self*.armor = armor  
  
 *def \_\_str\_\_*(*self*):  
 *return* (f'Warrior: Пол {*self*.gender}, возраст {*self*.age}, рост {*self*.height}, вес {*self*.weight}, запас сил '  
 f'{*self*.forces}, физический урон {*self*.physical\_damage}, броня {*self*.armor}.')  
  
 *def \_\_eq\_\_*(*self*, other):  
 *return self*.physical\_damage == other.physical\_damage *and self*.forces == other.forces *and self*.armor == other.armor  
  
  
*class* Magician(Character):  
 mana = PositiveNumber()  
 magic\_damage = PositiveNumber()  
  
 *def \_\_init\_\_*(*self*, gender, age, height, weight, mana, magic\_damage):  
 *super*().\_\_init\_\_(gender, age, height, weight)  
 *self*.mana = mana  
 *self*.magic\_damage = magic\_damage  
  
 *def \_\_str\_\_*(*self*):  
 *return* (f'Magician: Пол {*self*.gender}, возраст {*self*.age}, рост {*self*.height}, вес {*self*.weight}, запас маны '  
 f'{*self*.mana}, '  
 f'магический урон {*self*.magic\_damage}.')  
  
 *def* \_\_damage\_\_(*self*):  
 *return self*.magic\_damage \* *self*.mana  
  
  
*class* Archer(Character):  
 forces = PositiveNumber()  
 physical\_damage = PositiveNumber()  
 attack\_range = PositiveNumber()  
  
 *def \_\_init\_\_*(*self*, gender, age, height, weight, forces, physical\_damage, attack\_range):  
 *super*().*\_\_init\_\_*(gender, age, height, weight)  
 *self*.forces = forces  
 *self*.physical\_damage = physical\_damage  
 *self*.attack\_range = attack\_range  
  
 *def \_\_str\_\_*(*self*):  
 *return* (  
 f'Archer: Пол {*self*.gender}, возраст {*self*.age}, рост {*self*.height}, вес {*self*.weight}, запас сил {*self*.forces}, '  
 f'физический урон {*self*.physical\_damage}, дальность атаки {*self*.attack\_range}.')  
  
 *def \_\_eq\_\_*(*self*, other):  
 *return* (*self*.physical\_damage == other.physical\_damage *and self*.forces == other.forces *and self*.attack\_range  
 == other.attack\_range)  
  
  
*class* WarriorList(*list*):  
 *def \_\_init\_\_*(*self*, name):  
 *super*().*\_\_init\_\_*(*self*)  
 *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*)  
 *self*.name = name  
  
 *def* extend(*self*, iterable):  
 *if all*(*isinstance*(i, Magician) *for* i *in* iterable):  
 *super*().extend(iterable)  
  
 *def* print\_damage(*self*):  
 damage = 0  
 *for* i *in range*(*len*(*self*)):  
 damage += *self*[i].magic\_damage  
 *print*(damage)  
  
  
*class* ArcherList(*list*):  
 *def \_\_init\_\_*(*self*, name):  
 *super*().\_\_init\_\_(*self*)  
 *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 range*(*len*(*self*)):  
 *if* (*self*[i].gender == 'm'):  
 count += 1  
 *print*(count)