**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 |  | Львов А.В. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Ознакомление с принципами ООП, их реализацией на языке Python.

## 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:

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

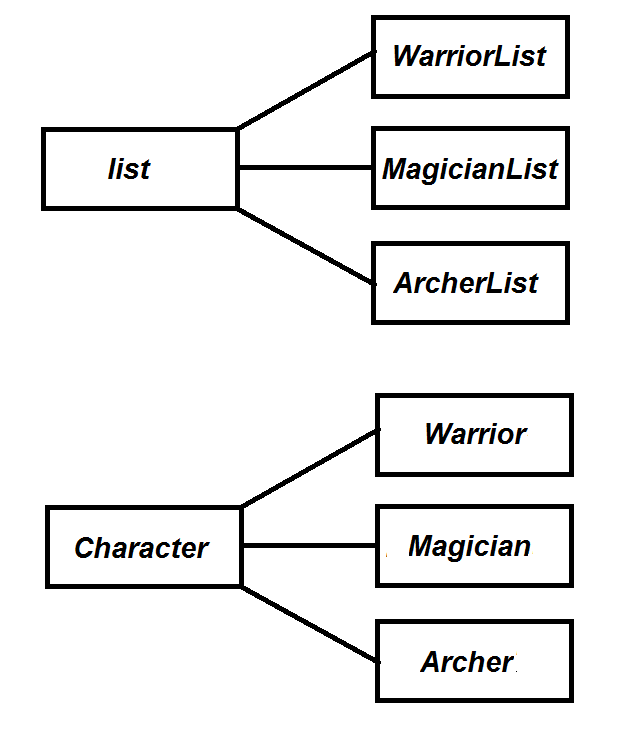


Рисунок 1 - Наследование классов

В начале работы создается базовый класс Character с требуемыми в задании полями.

Класс Warrior наследуется от базового класса Character и переопределяет методы \_\_str\_\_(), позволяющий получить строковое представление класса и \_\_eq\_\_(), позволяющий сравнивать объекты класса между собой.

Класс Magician наследуется от базового класса Character и переопределяет метод \_\_str\_\_(), позволяющий получить строковое представление класса. Также определяется метод \_\_damage\_\_(), возвращающий значение магического урона, который может нанести маг, если потратит сразу весь запас маны.

Класс Archer наследуется от базового класса Character и переопределяет методы \_\_str\_\_(), позволяющий получить строковое представление класса и \_\_eq\_\_(), позволяющий сравнивать объекты класса между собой.

Класс WarriorList наследуется от базового класса list и переопределяет метод append(), позволяя добавлять в список объекты класса Warrior. Также класс определяет метод print\_count(), позволяющий получить количество воинов.

Класс MagicianList наследуется от базового класса list и переопределяет метод extend(), позволяя добавлять в список объекты класса Magician. Также класс определяет метод print\_damage(), позволяющий получить общий урон всех магов.

Класс ArcherList наследуется от базового класса list и переопределяет метод append(), позволяя добавлять в список объекты класса Archer. Также класс определяет метод print\_count(), позволяющий получить количество лучников мужского пола.

## 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)  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 |

## Выводы

Было проведено ознакомление с основными принципами ООП, была разработана программа, реализующая систему классов на языке Python.

# Приложение А Исходный код программы

Название файла: main.py

class Character:  
 def \_\_init\_\_(self, gender, age, height, weight):  
 if self.check\_values(gender, age, height, weight):  
 self.gender = gender  
 self.age = age  
 self.height = height  
 self.weight = weight  
 else:  
 raise ValueError('Invalid value')  
  
 def check\_values(self, gender, age, height, weight):  
 if not (isinstance(gender, str) and gender in 'mw'):  
 return False  
 if not (self.check\_int\_positive(age, height, weight)):  
 return False  
 return True  
  
 def check\_int\_positive(self, \*args):  
 for num in args:  
 if not (isinstance(num, int) and num > 0):  
 return False  
 return True  
  
  
class Warrior(Character):  
 def \_\_init\_\_(self, gender, age, height, weight, forces, physical\_damage, armor):  
 if self.\_check\_values(gender, age, height, weight, forces, physical\_damage, armor):  
 self.gender = gender  
 self.age = age  
 self.height = height  
 self.weight = weight  
 self.forces = forces  
 self.physical\_damage = physical\_damage  
 self.armor = armor  
 else:  
 raise ValueError('Invalid value')  
  
 def \_check\_values(self, gender, age, height, weight, forces, physical\_damage, armor):  
 return super().check\_values(gender, age, height, weight) and super().check\_int\_positive(forces, physical\_damage,  
 armor)  
  
 def \_\_str\_\_(self):  
 return (f'Warrior: Пол {self.gender}, '  
 f'возраст {self.age}, '  
 f'рост {self.height}, '  
 f'вес {self.weight}, '  
 f'запас сил {self.forces}, '  
 f'физический урон {self.physical\_damage}, '  
 f'броня {self.armor}.')  
  
 def \_\_eq\_\_(self, other):  
 return all(  
 [self.physical\_damage == other.physical\_damage, self.forces == other.forces, self.armor == other.armor])  
  
  
class Magician(Character):  
 def \_\_init\_\_(self, gender, age, height, weight, mana, magic\_damage):  
 if self.\_check\_values(gender, age, height, weight, mana, magic\_damage):  
 self.gender = gender  
 self.age = age  
 self.height = height  
 self.weight = weight  
 self.mana = mana  
 self.magic\_damage = magic\_damage  
 else:  
 raise ValueError('Invalid value')  
  
 def \_check\_values(self, gender, age, height, weight, mana, magic\_damage):  
 return super().check\_values(gender, age, height, weight) and super().check\_int\_positive(mana, magic\_damage)  
  
 def \_\_str\_\_(self):  
 return (f'Magician: Пол {self.gender}, '  
 f'возраст {self.age}, '  
 f'рост {self.height}, '  
 f'вес {self.weight}, '  
 f'запас маны {self.mana}, '  
 f'магический урон {self.magic\_damage}.')  
  
 def \_\_damage\_\_(self):  
 return self.magic\_damage \* self.mana  
  
  
class Archer(Character): # Наследуется от класса Character  
 def \_\_init\_\_(self, gender, age, height, weight, forces, physical\_damage, attack\_range):  
 if self.\_check\_values(gender, age, height, weight, forces, physical\_damage, attack\_range):  
 self.gender = gender  
 self.age = age  
 self.height = height  
 self.weight = weight  
 self.forces = forces  
 self.physical\_damage = physical\_damage  
 self.attack\_range = attack\_range  
 else:  
 raise ValueError('Invalid value')  
  
 def \_check\_values(self, gender, age, height, weight, forces, physical\_damage, attack\_range):  
 return super().check\_values(gender, age, height, weight) and super().check\_int\_positive(forces, physical\_damage,  
 attack\_range)  
  
 def \_\_str\_\_(self):  
 return (f'Archer: Пол {self.gender}, '  
 f'возраст {self.age}, '  
 f'рост {self.height}, '  
 f'вес {self.weight}, '  
 f'запас сил {self.forces}, '  
 f'физический урон {self.physical\_damage}, '  
 f'дальность атаки {self.attack\_range}.')  
  
 def \_\_eq\_\_(self, other):  
 return all([self.physical\_damage == other.physical\_damage, self.forces == other.forces,  
 self.attack\_range == other.attack\_range])  
  
  
class WarriorList(list):  
 def \_\_init\_\_(self, name):  
 super().\_\_init\_\_()  
 self.name = name  
  
 def append(self, \_\_object):  
 if isinstance(\_\_object, Warrior):  
 super().append(\_\_object)  
 else:  
 raise TypeError(f'Invalid type {type(\_\_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 elem in \_\_iterable:  
 if isinstance(elem, Magician):  
 super().append(elem)  
  
 def print\_damage(self):  
 print(sum([i.magic\_damage for i in self]))  
  
  
class ArcherList(list):  
 def \_\_init\_\_(self, name):  
 super().\_\_init\_\_()  
 self.name = name  
  
 def append(self, \_\_object):  
 if isinstance(\_\_object, Archer):  
 super().append(\_\_object)  
 else:  
 raise TypeError(f'Invalid type {type(\_\_object)}')  
  
 def print\_count(self):  
 print(len([i for i in self if i.gender == 'm']))