**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.Вариант 1

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Атоян М |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Научится работать с классами, создавать методы и функции для классов, понять принцип наследования и переопределения, понять, как работает super().

Необходимо создать программу, которая может на основе различных классов создавать объекты фигур и работать с ними. Также программа должна уметь определять верный тип данных, а также уметь добавлять в определённую группу объектов.

## Задание

**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.

В отчете укажите:

1. Изображение иерархии описанных вами классов.

2. Методы, которые вы переопределили (в том числе методы класса object).

3. В каких случаях будут использованы методы \_\_str\_\_() и \_\_print\_damage\_\_().

4. Будут ли работать переопределенные методы класса list для созданных списков? Объясните почему и приведите примеры.

**Выполнение работы**

В лабораторной работе необходимо создать классы с определёнными методами, которые представляют собой фигуры с определёнными параметрами и списки для хранения этих фигур.

Класс Character будет родительским для классов Warrior, Magician и Archer и содержать в себе пол, рост, возраст и вес. При создании класса происходит проверка на тип входных данных, а также на их корректность.

Метод \_\_str\_\_() отвечает за строковое представление дочерних классов Warrior, Magician и Archer, т.е. когда происходит преобразование объекта класса в тип str.

Метод \_\_eq\_\_() отвечает за проверку на равенство дочерних классов Warrior и Archer.

Класс WarriorList, MagicianList и ArcherList наследуется от класса list. Был переопределён append для WarriorList и ArcherList, который работает точно так же, как и с любым другим списком, ведь мы наследуем WarriorList, MagicianList и ArcherList от list, однако при выполнении осуществляется проверка на принадлежность объекта к соответствующему классу, иначе выбрасываем TypeError.

Был переопределён метод extend для MagicianList. Если объект в итерируемом объекте, является магом, то он добавляется в конец MagicianList, иначе игнорируется.

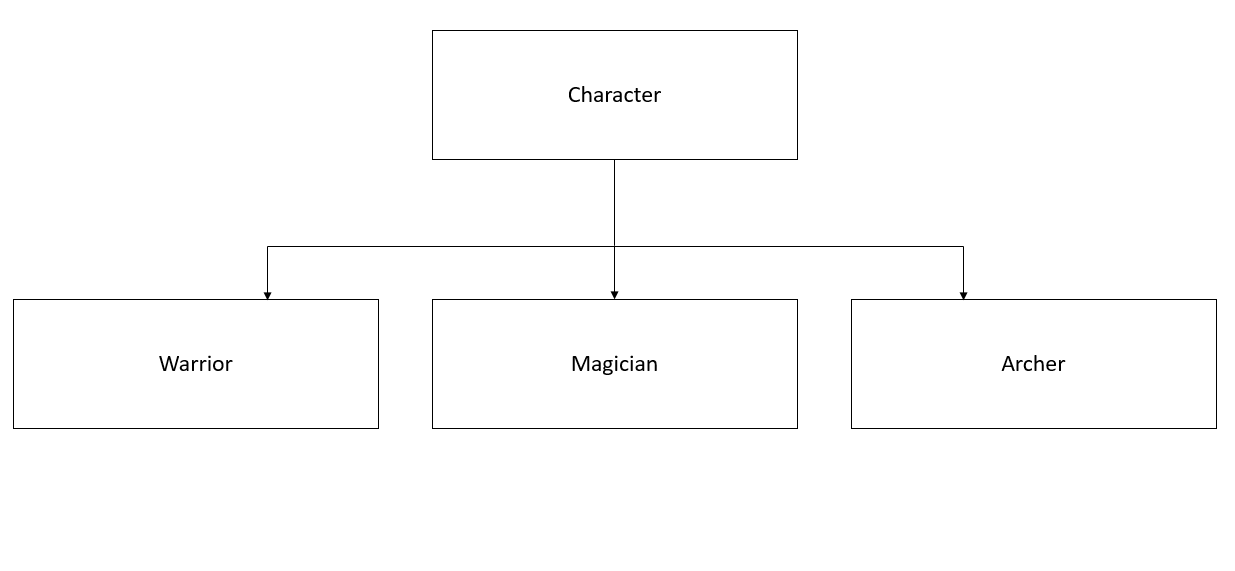


Рисунок 1 – Иерархия классов юнитов

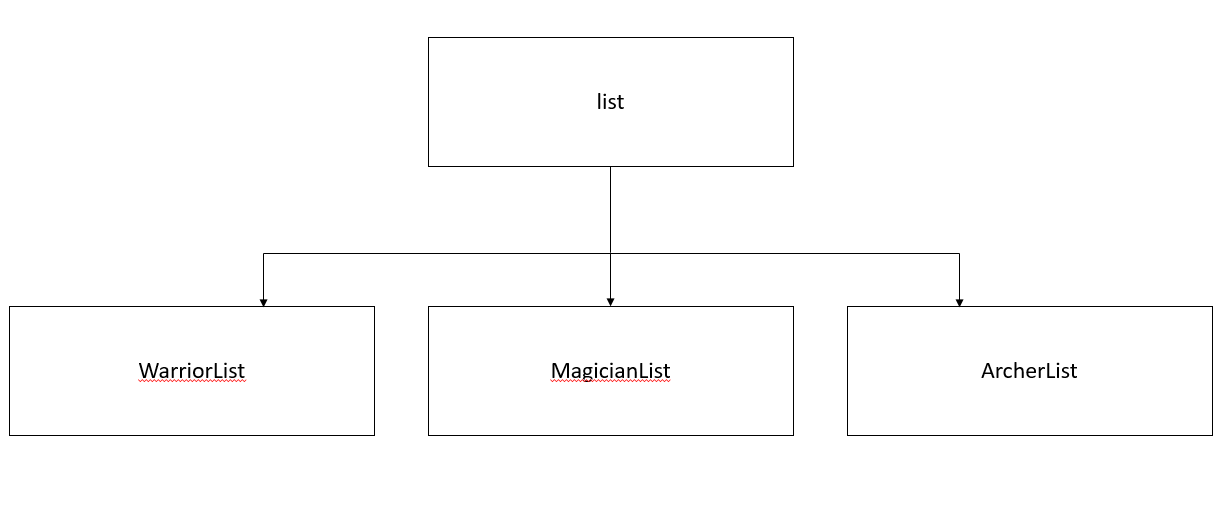


Рисунок 2 – Иерархия классов списков юнитов

## Выводы

Были изучены способы наследования от различных классов, переопределение методов, а также возможность использовать функции super() для того, чтобы использовать методы родительского класса.

Была реализована программа, которая создаёт объекты классов различных фигур, может добавлять их в определённые группы и работать с ними.

# Приложение А Исходный код программы

Название файла: main.py

class Character:

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

if gender not in ['m', 'w']:

raise ValueError('Invalid value')

if not isinstance(age, int) or age <= 0:

raise ValueError('Invalid value')

if not isinstance(height, int) or height <= 0:

raise ValueError('Invalid value')

if not isinstance(weight, int) or weight <= 0:

raise ValueError('Invalid value')

self.gender = gender

self.age = age

self.height = height

self.weight = weight

class Warrior(Character):

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

super().\_\_init\_\_(gender, age, height, weight) # Вызываем конструктор родительского класса

if not isinstance(forces, int) or forces <= 0:

raise ValueError('Invalid value')

if not isinstance(physical\_damage, int) or physical\_damage <= 0:

raise ValueError('Invalid value')

if not isinstance(armor, int) or armor <= 0:

raise ValueError('Invalid value')

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 {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):

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

super().\_\_init\_\_(gender, age, height, weight) # Вызываем конструктор родительского класса

if not isinstance(mana, int) or mana <= 0:

raise ValueError('Invalid value')

if not isinstance(magic\_damage, int) or magic\_damage <= 0:

raise ValueError('Invalid value')

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 {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) # Вызываем конструктор родительского класса

if not isinstance(forces, int) or forces <= 0:

raise ValueError('Invalid value')

if not isinstance(physical\_damage, int) or physical\_damage <= 0:

raise ValueError('Invalid value')

if not isinstance(attack\_range, int) or attack\_range <= 0:

raise ValueError('Invalid value')

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}, 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.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).\_\_name\_\_}')

def print\_count(self):

print(len(self))

class MagicianList(list):

def \_\_init\_\_(self, name):

super().\_\_init\_\_()

self.name = name

def extend(self, iterable):

for item in iterable:

if isinstance(item, Magician):

self.append(item)

def print\_damage(self):

total\_damage = 0

for magician in self:

total\_damage += magician.magic\_damage

print(total\_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).\_\_name\_\_}')

def print\_count(self):

male\_archers\_count = sum(1 for archer in self if archer.gender == 'm')

print(male\_archers\_count)