Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

ОТЧЕТ

к лабораторной работе 1
по дисциплине «Проектирование программного обеспечения интеллектуальных систем»

Выполнил: И.Е.Рыбакова
Проверил: С. В. Бутрин

Минск 2023

Цель: изучить основные возможности языка Python для разработки программных систем командной строки (CLI)
Вариант: модель животного мира

Для создания интерфейса была использована библиотека click. Пользователь может вводить следующие команды:

- –add - добавляет новое существо. При этом можно дополнительно указать некоторые свойство существа (координаты, количество жизней, пол, количество дней без еды).
- –next - делает один шаг в игровом мире
- –auto - делает количество шагов, введенное пользователем
- –info - выводит информацию о мире (список всех существ и их аттрибуты)

```
@click.command()
@click.option("--add", type=str)
@click.option("-x", type=int)
@click.option("-y", type=int)
@click.option("-l", "--lives", type=str)
@click.option("-sd", "--starve_days", type=str)
@click.option("-s", "--sex", type=str)
@click.option("--auto", type=int, help="N days pass")
@click.option("--next", "n", is_flag=True, help="One day passes")
@click.option("--info", "inf", is_flag=True, help="Prints iformation about the world")
def live(add, x, y, lives, starve_days, sex, auto, n, inf):
```

Информация о мире сохраняется в текстовом файле, первая строчка содержит информацию о размерах мира, остальные соответствуют существам, седержащимся в мире. Первое слово в строке - вид существа, далее в виде чисел перечисляются координаты, количество жизней, а также количество количество дней без еды и пол, если существо - животное. При изменениях в мире информация в файле обновляется.

Основным классом является класс world, который инициализируется, считывая информацию из файла. Среди аттрибутов класса есть field, которое представляет собой двумерный массив объектов класса cell, аттрибутом которого является массив объектов класса living. Также аттрибутами являются размеры данного поля. Среди методов данного класса есть методы add (добовляет новое существо), find (ищет и возращает существо с заданными параметрами на заданной клетке поля), next (делает один шаг).

Класс living описывает живых существ. От данного класса наследуются классы plant (описывает растения) и animal (описывает животных). От

класса animal наследуются классы predator (хищники) и herbivore (травоядные), от которых наследуются классы, описывающие конкретные виды животных (fox, wolf, hare, moose). В данных классах содержатся методы, реализующие передвижение, размножение, питание и смерть.

D: > 2_sem > ppois > plaba1 > info.txt

```
10 10
plant 8 9 4
plant 2 3 7
plant 4 6 8
plant 9 8 8
plant 5 6 3
fox 9 8 6 0 m
fox 7 4 6 0 f
wolf 8 5 7 0 f
wolf 3 7 7 0 m
hare 0 6 5 0 m
hare 0 1 5 0 m
moose 2 9 11 0 m
moose 4 6 11 0 f
moose 6 4 11 0 f
fox 1 6 6 0 f
```

```
class world:
    def __init__(self, file):
        f = open(file)
        size = f.readline().split()
        self.n = int(size[0])
        self.m = int(size[1])
        self.field = [[cell() for j in range(self.m)] for i in range(self.n)]
        self.everyone = []

        while True:
            inf = f.readline().split()
            if inf[0] == '.':
                break
            self.field[int(inf[1])][int(inf[2])].livings.append(eval(inf[0])(self, *inf))
            self.everyone.append(self.field[int(inf[1])][int(inf[2])].livings[-1])
        f.close()
```

```
class animal(living):

    def give_birth(self):
        inf = [self.who(), str(self.x), str(self.y), self.max_lives, '0']
        if randint(0,2) == 0:
            sex = 'm'
        else:
            sex = 'f'
        inf.append(sex)
        self.home.field[self.x][self.y].livings.append(eval(self.who())(self.home, *inf))
        self.home.everyone.append(self.home.field[self.x][self.y].livings[-1])
        plus_one = self.id() + self.home.field[self.x][self.y].livings[-1].id()
        self.change(plus_one)

    def where(self, old_pos, distance):
        if old_pos < distance:
            new_pos = old_pos + distance
        elif self.home.n-old_pos <= distance:
            new_pos = old_pos - distance
        else:
            if randint(0,2) == 0:
```

```
class predator(animal):
    def type(self):
        return 'predator'

    def eat(self, other):
        other.die()
        info = self.id().split()
        info[4] = '0'
        not_hungry = ' '.join(info) + '\n'
        self.change(not_hungry)
        self.starve_days = 0

    def next(self):
        if self.sex == 'f' and self.home.find(self.x, self.x, self.who(), s = 'm'):
            self.give_birth()
            print('on cell  ' + str(self.x) + ' ' + str(self.y) + ' new ' + self.who() + ' was bor
        if self.find_prey() != None:
            print(self.who() + ' eats ' + self.find_prey().who() + ' on cell ' + str(self.x) + ' '
```

```
class wolf(predator):
    size = 2
    max_lives = '7'
    max_distance = 3
    max_starve = 4

class fox(predator):
    size = 1
    max_lives = '6'
    max_distance = 2
    max_starve = 5
```