**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»**

Факультет физико-математических и естественных наук

Кафедра информационных технологий

**ОТЧЕТ  
по лабораторной работе №1**  
по дисциплине «Python и его приложения»

Выполнили:

Студенты группы НПИбд-01-22

Ситникова Диана Александровна

Москва 2023

**Задание – реализовать следующую программу:**

Создать класс Поле, хранящий двумерный массив символов – игровое поле. Реализовать **случайное заполнение** поля различными символами, например ‘\_’ для пустой клетки, ‘0’ для препятствия, ‘\*’ ‘!’ для специальных плиток.

Создать класс Юнит, хранящий координаты юнита и текст. Сделать класс Юнит **абстрактным**, объявить в нём абстрактный метод движения, после **унаследовать** от него два подкласса и в них определить конкретные реализации этого метода. Например, класс Unit и наследующие ему Rook и Bishop, передвигающиеся как соответствующие шахматные фигуры.

Альтернативно можете сделать абстрактный класс поле с разными наследниками – прямоугольны и круглым полем.

Также реализуйте взаимодействие с разными клетками поля. Например, при перемещении на клетку ‘\*’ у юнита отнимаются жизни, а на клетке ‘$’ добавляется счёт. Можете придумать свою вариацию.

Перемещение юнита должно реализовываться вводом команд с клавиатуры. Каждый ход нужно перерисовывать поле, со специальным символом на месте юнита. **Придумайте свои вариации наследных классов.**

**Ход работы:**

В ходе работы мы реализовали небольшую терминальную игру, на начало которой юниту дается 3 единицы здоровья и единица бонуса. Бонусы можно тратить в магазине на опции для следующего хода юнита. При этом одна опция имеет стоимость одного бонуса ($).

В терминале поле рандомно заполняется следующими символами:

& - иконка положения юнита на поле;

\* - препятствие, отнимающее единицу здоровья;

0 - стена (непроходимое препятствие);

\_ - пустая клетка;

$ - единица бонуса;

Перемещаясь по полю при помощи символов w (вверх), a (влево), s (вниз), d (вправо), юнит собирает бонусы или сталкивается с препятствиями.

Магазин имеет следующие варианты опций:

1. Разрушить ближайшую стену (выбор: w, a, s, d);
2. Совершить ход как шахматная ладья (по горизонтали или по вертикали на любую клетку, либо до столкновения со стеной, встав на ее место) – для этого необходимо указать координаты желаемой точки перемещения (y, x);
3. Совершить ход как шахматный слон (по диагоналям на любую клетку, либо до столкновения со стеной, встав на ее место) – для этого необходимо указать координаты желаемой точки перемещения (y, x).

Тело магазина реализовано в классе Unit, а варианты опций в соответствующих подклассах, наследуемых от Unit – Crush, Rook, Bishop. Если баланс бонусов будет 0, вход в магазин не будет доступен.

В подклассе Crush реализована проверка на наличие вблизи «стен» и обход случая, если была выбрана пустая клетка. В случае отсутствия «стен» поблизости, происходит выход без потери бонуса, иначе, в выбранной клетке со стеной появляется символ пустой клетки, и игра продолжается. В случае выбора клетки без «стены» предлагается перевыбрать клетку.

В подклассе Rook реализована проверка на соответствие координат условиям хода шахматной ладьи (одна из указанных координата совпадает с координатой исходного положения юнита, но не обе сразу; координаты не выходят за пределы поля). В случае ввода несоответствующих координат, будет запрошен повторный их ввод. Иначе, юнит будет перемещаться до нужной координаты, собирая и опустошая клетки с бонусами и препятствиями, отнимающими здоровье. Если же юнит столкнется со «стеной», по правилам шахмат он встанет на место препятствия (снесет стену) и не сможет пройти дальше. Бонус снимается после завершения хода.

В подклассе Bishop реализована проверка на соответствие координат условиям хода шахматной ладьи (ни одна указанная координата не совпадает с координатой исходного положения юнита; координаты равноудалены от координат положения юнита, для соблюдения диагонали; координаты не выходят за пределы поля). В случае ввода несоответствующих координат, будет запрошен повторный их ввод. Иначе, юнит будет перемещаться до нужной координаты, собирая и опустошая клетки с бонусами и препятствиями, отнимающими здоровье. Если же юнит столкнется со «стеной», по правилам шахмат он встанет на место препятствия (снесет стену) и не сможет пройти дальше. Бонус снимается после завершения хода.

Исходный код файл Field.py:

from random import randrange as rand

from unit import Unit

class Field:

def \_\_init\_\_(self, w, h):

self.w = w

self.h = h

self.matrix = []

self.generate()

def is\_obstacle(self, x, y):

if x < 0 or x >= self.w or y < 0 or y >= self.h:

return True

return self.matrix[y][x] == '0'

def generate(self):

for i in range(self.h):

self.matrix.append([])

for j in range(self.w):

rng = rand(100)

if rng < 50 :

self.matrix[i].append('\_') # пустая клетка

elif rng < 80:

self.matrix[i].append('0') # препятствие

elif rng < 95:

self.matrix[i].append('$') # +1 бонус

else:

self.matrix[i].append('\*') # -1 hp

def output(self):

for row in self.matrix:

#print(row)

line =''

for cell in row:

line += cell + ' '

print(line)

f = Field(5, 7)

u = Unit('&', 0, 0, f)

print("You have", u.hp, "HP now and", u.scor, "bonuses.\nTo end the game, press E or die!\nYou can also buy a special move in the store.\nPress B to open the stor.")

f.output()

key = ' '

while key != 'e' and u.hp > 0: # e - стопер игры

key = input("Your move: ")

u.move(key)

f.output()

print(" HP:", u.hp, "\n Bonus:", u.scor, "\n The end.") # итожик игры

Исходный код файл unit.py:

class Unit:

def \_\_init\_\_(self, icon, x, y, field):

self.icon = icon

self.field = field

self.x = x #width

self.y = y #height

self.scor = 1 #счетчик бонусов

self.hp = 3 #счетчик здоровья

self.field.matrix[y][x] = icon # иконка текущего положения

# меню магазина

def store(self, y, x):

choice = ' '

print("Welcome to the store!\nYou can buy one action for one bonus:\n1. Crush the wall\n2. Rook move\n3. Bishop move\n0. Exit the store\n")

choice = input("Your choice: ")

if choice == '1':

Crush.wall(self, y, x)

elif choice == '2':

Rook.move(self, y, x)

elif choice == '3':

Bishop.move(self, y, x)

else:

return

#ф-ция передвижения

def move(self, key):

x = self.x

y = self.y

if key == 'w':

y -= 1

elif key == 's':

y += 1

elif key == 'a':

x -= 1

elif key == 'd':

x += 1

elif key == 'b':

if self.scor > 0:

Unit.store(self, y, x)

else:

print("Not enough bonuses!")

return

if self.field.is\_obstacle(x, y):

print('Can not go')

return

self.field.matrix[self.y][self.x] = '\_'

self.x = x

self.y = y

if self.field.matrix[y][x] == '$':

self.scor += 1

if self.field.matrix[y][x] == '\*':

self.hp -= 1

self.field.matrix[y][x] = self.icon

class Crush(Unit):

# уничтожение 1 стены без перемещения

def wall(self, y, x):

if self.field.matrix[y+1][x] == '0' or self.field.matrix[y-1][x] == '0' or self.field.matrix[y][x+1] == '0' or self.field.matrix[y][x-1] == '0':

choice = input("Choose the wall: ")

wall = True

while wall:

if choice == 'w' and self.field.matrix[y-1][x] == '0':

self.field.matrix[y-1][x] = '\_'

break

elif choice == 's' and self.field.matrix[y+1][x] == '0':

self.field.matrix[y+1][x] = '\_'

break

elif choice == 'a' and self.field.matrix[y][x-1] == '0':

self.field.matrix[y][x-1] = '\_'

break

elif choice == 'd' and self.field.matrix[y][x+1] == '0':

self.field.matrix[y][x+1] = '\_'

break

else:

choice = input("Choose another side: ")

self.scor -= 1

else:

print("There are no walls here")

return

class Rook(Unit):

def fun(self, y, x):

if self.field.matrix[y][x] == '0': return True

if self.field.matrix[y][x] == '$': self.scor += 1

if self.field.matrix[y][x] == '\*': self.hp -= 1

return False

# перемещение как ладья в шахматах (до первого препятствия по вертикали/горизонтали)

def move(self, y, x):

print(y, x)

k = int(input("Choose the height: "))

l = int(input("Choose the width: "))

self.field.matrix[y][x] = '\_'

while (y != k and x != l) or (y == k and x == l) or k >= self.field.h or k < 0 or l >= self.field.w or l < 0:

k = int(input("Choose the height: "))

l = int(input("Choose the width: "))

if k > y:

while y != k:

y += 1

if Rook.fun(self, y, x):

break

elif k < y:

while y != k:

y -= 1

if Rook.fun(self, y, x):

break

elif l > x:

while x != l:

x += 1

if Rook.fun(self, y, x):

break

elif l < x:

while x != l:

x -= 1

if Rook.fun(self, y, x):

break

self.field.matrix[y][x] = self.icon

self.scor -= 1

self.x = x

self.y = y

class Bishop(Unit):

# перемещение как слон в шахматах (до первого препятствия по диагоналям)

def move(self, y, x):

print(y, x)

k = int(input("Choose the height: "))

l = int(input("Choose the width: "))

self.field.matrix[y][x] = '\_'

while x == l or y == k or (y == k and x == l) or k >= self.field.h or k < 0 or l >= self.field.w or l < 0 or abs(l - x) != abs(k - y):

k = int(input("Choose the height: "))

l = int(input("Choose the width: "))

if k > y and l > x: # ход в правую нижнюю диагональ

while y != k and x != l:

y += 1

x += 1

if Rook.fun(self, y, x):

break

elif k < y and l > x: # ход в правую верхнюю диагональ

while y != k and x != l:

y -= 1

x += 1

if Rook.fun(self, y, x):

break

elif k > y and l < x: # ход в левую нижнюю диагональ

while y != k and x != l:

y += 1

x -= 1

if Rook.fun(self, y, x):

break

elif k < y and l < x: # ход в левую верхнюю диагональ

while y != k and x != l:

y -= 1

x -= 1

if Rook.fun(self, y, x):

break

self.field.matrix[y][x] = self.icon

self.scor -= 1

self.x = x

self.y = y