IPC1

PRACTICA 3

MANUAL TÉCNICO

PY MAN 770C

DAMIÁN IGNACIO PEÑA AFRE
202110568

Contenido

Clase Usuario

```python
class Usuario:
    nombre = ''
    punteo = 0
    vidas = 1

    def __init__(self, nombre):
        self.nombre = nombre

    def aumentarPuntaje(self):
        self.punteo += 10

    def disminuirVidas(self):
        self.vidas -= 1
```

Propiedades principales del usuario/jugador

Constructor, recibe como parámetro el nombre

Método para aumentar el puntaje

Método para disminuir el numero de vidas

Clase Tablero

```python
import random
from usuario import Usuario

FILAS = 5
COLUMNAS = 6

# Items
FANTASMA = "@"
PREMIO = "O"
BLOQUE = "X"
PERSONAJE = "<"


class Tablero:
    tablero = []
    cantidadPremios = 1
    cantidadParedes = 1
    cantidadFantasmas = 1
    elementosRegidos = 0
    jugador = None
    juegoTerminado = False

    posxPacman = 0
    poyPacman = 0

    def __init__(self, jugador):
        self.jugador = jugador
        self.inicializarTablero()

    def inicializarTablero(self):
        self.tablero = []
        for i in range(FILAS):
            fil = []
            for j in range(COLUMNAS):
                fil.append(None)

            self.tablero.append(fil)

    def setPremios(self, c):
        self.cantidadPremios = c

    def setParedes(self, c):
        self.cantidadParedes = c

    def setFantasmas(self, c):
        self.cantidadFantasmas = c
```

Importa el modulo random y la clase usuario

Define el numero de filas y columnas del tablero

Caracteres de los elementos del juego

Propiedades del juego

Empieza un nuevo juego, pasando en el constructor al jugador y creando un nuevo tablero

Genera una lista de la forma

[[None, None, ...],

[[None, None, ...]]

De manera que sea de 5 x 6

Setters de las cantidades de premios, paredes y fantasmas

```python
def movimiento(self, tecla):
    msg = ''

    self.tablero[self.posxPacman][self.posyPacman] = None

    posxAntigua = self.posxPacman
    posyAntigua = self.posyPacman

    if(tecla == 'W'):
        self.posxPacman -= 1
    elif(tecla == 'S'):
        self.posxPacman += 1
    elif(tecla == 'A'):
        self.posyPacman -= 1
    elif(tecla == 'D'):
        self.posyPacman += 1
    elif(tecla == 'F'):
        self.juegoTerminado = True
        msg = 'Juego terminado'

    if(self.posxPacman < 0
    or self.posxPacman >= FILAS
    or self.posyPacman < 0
    or self.posyPacman >= COLUMNAS):
        self.posxPacman = posxAntigua
        self.posyPacman = posyAntigua
        msg = 'No es posible atravesar los extremos'

    # Fantasma
    if(self.tablero[self.posxPacman][self.posyPacman] == FANTASMA):
        self.tablero[self.posxPacman][self.posyPacman] = PERSONAJE
        self.jugador.disminuirVidas()

    if(self.tablero[self.posxPacman][self.posyPacman] == PREMIO):
        self.jugador.aumentarPuntaje()
        self.elementosRegidos += 1

    if(self.tablero[self.posxPacman][self.posyPacman] == BLOQUE):
        self.posxPacman = posxAntigua
        self.posyPacman = posyAntigua
        msg = 'No es posible atravesar una pared'

    self.tablero[self.posxPacman][self.posyPacman] = PERSONAJE
```

Recibe por parámetro la tecla presionada así como un posible mensaje de respuesta

Borra al Personaje de la posición inicial

Guarda la posición en la que empezó el personaje

Opciones de movimiento que alteran la posición y opción de finalizar la partida alterando la propiedad juego terminado

Validación de movimientos inválidos

Reestablece la posición anterior

Interacción con un fantasma (disminuye vidas)

Interacción con un premio (aumenta puntaje)

Reestablece la posición inicial al pasar por un Bloque

Mueve al pacman a la posición determinada

```
        if(self.jugador.vidas == 0):
            self.juegoTerminado = True
            msg = 'Moriste'

        if(self.elementosRegidos == self.cantidadPremios):
            self.juegoTerminado = True
            msg = 'Haz ganado'

        return msg

    def posicionInicialPacman(self, x, y):

        if(x > -1 and x < FILAS and y > -1 and y < COLUMNAS):
            if(self.tablero[x][y] == None):
                self.tablero[x][y] = PERSONAJE
                self.posxPacman = x
                self.posyPacman = y
            else:
                raise Exception("Posicion ya ocupada")
        else:
            raise Exception("Fuera de los limites")

    def mostrarTablero(self):
        print("Usuario: ", self.jugador.nombre)
        print("Punteo: ", self.jugador.punteo)

        print("\n--------------------")
        for fila in self.tablero:
            cadena = '|'
            for elemento in fila:
                if(elemento == None):
                    cadena += '   '
                else:
                    car = ' ' + elemento + ' '
                    cadena += car
            print(cadena+"|")
        print("--------------------\n")

    def generarEstadoInicial(self):
        self.generarElemento(self.cantidadPremios, PREMIO)
        self.generarElemento(self.cantidadParedes, BLOQUE)
        self.generarElemento(self.cantidadFantasmas, FANTASMA)
```

Validación de fin de juego, por premios recogidos o por número de vidas

Recibe como parámetro la posición inicial del pacman, verificando si es valida

Muestra los datos del usuario

Imprime lo contenido en la lista Tablero

Establece los elementos que se generarán

```python
def generarElemento(self, cantidad, simbolo):

    for i in range(cantidad):
        creado = False

        while(not creado):
            posy = random.randint(0, COLUMNAS-1)
            posx = random.randint(0, FILAS-1)

            if(self.tablero[posx][posy] == None):
                self.tablero[posx][posy] = simbolo
                creado = True
```

Ingresa la cantidad de items especificadas dentro de la lista Tablero

Clase Pacman (Principal)

```python
import random
from tablero import Tablero
from usuario import Usuario

# tablero
global objTablero
jugadores = []


def inicio():
    print('=== Menu de inicio ===')
    print('1. Iniciar Juego')
    print('2. Salir')
    try:
        resp = int(input('\nSelecciona una opcion : '))

        if (resp == 1):
            infoUsuario()
        elif(resp == 2):
            exit()
        else:
            raise ValueError("Fuera del rango")
    except ValueError:
        print('\nValor incorrecto\n')
        inicio()


# def usuario():


def infoUsuario():
    try:
        nombre = str(input("Nombre de usuario : "))
        global jugador
        jugador = Usuario(nombre)
        jugadores.append(jugador)
        dimensiones()
    except ValueError:
        infoUsuario()
```

Importa el modulo random y las clases Usuario y Tablero

Objeto Tablero donde se desarrolla la partida

Impresión de las opciones

Entrada y validación de la misma

Generación del usuario a partir del nombre introducido

```
def dimensiones():
    global objTablero
    objTablero = Tablero(jugador)
    print("=== Generando tablero y posicion inicial ===")
    print("Tablero")

    cantidadPremios()


def cantidadPremios():
    try:
        cantidad = random.randint(3, 6)
        if(cantidad > 0 and cantidad < 13):
            objTablero.setPremios(cantidad)
            cantidadParedes()
        else:
            raise ValueError("Fuera del rango")

    except ValueError:
        # print('\nValor incorrecto\n')
        cantidadPremios()


def cantidadParedes():
    try:
        cantidad = random.randint(5, 12)
        if(cantidad > 0 and cantidad < 7):
            objTablero.setParedes(cantidad)
            cantidadFantasmas()
        else:
            raise ValueError("Fuera del rango")

    except ValueError:
        # print('\nValor incorrecto\n')
        cantidadPremios()


def cantidadFantasmas():
    try:
        cantidad = random.randint(1, 6)
        if(cantidad > 0 and cantidad < 7):
            objTablero.setFantasmas(cantidad)
            estadoInicial()
        else:
            raise ValueError("Fuera del rango")

    except ValueError:
        # print('\nValor incorrecto\n')
        cantidadPremios()
```

Inicializa al tablero

Establece la cantidad de premios

Establece la cantidad de paredes

Establece la cantidad de fantasmas

```
def estadoInicial():
    objTablero.generarEstadoInicial()
    objTablero.mostrarTablero()
    posicionInicial()


def posicionInicial():
    try:
        fil = random.randint(1, 5)
        col = random.randint(1, 6)

        objTablero.posicionInicialPacman(fil-1, col-1)

        juego()

    except ValueError:
        # print("Valor incorrecto")
        posicionInicial()
    except Exception:
        # print("Posicion invalida")
        posicionInicial()


def juego():
    while(not objTablero.juegoTerminado):
        objTablero.mostrarTablero()
        print("W: Arriba |
       S: Abajo |
       D: Derecha |
       A: Izquierda |
       F: finalizar Partida")
        try:
            tecla = str(input()).upper()
            if(tecla == 'W'
      or tecla == 'S'
      or tecla == 'D'
      or tecla == 'A'
      or tecla == 'F'):
                mensaje = objTablero.movimiento(tecla)
                print(mensaje)
            else:
                raise ValueError("Tecla invalida")

        except ValueError:
            print("valor invalido")

    print(" Juego Finalizado ")
    inicio()
inicio()
```

Genera el estado inicial del tablero con sus items

Establece la posición inicial del personaje

Ejecuta el bucle principal del juego, mostrando las opciones

Validacion de las teclas presionadas

Empieza el juego