Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela Ciencias y sistemas

MANUAL TÉCNICO

Introducción a la programación y computación 2

Rubén Alejandro Ralda Mejia

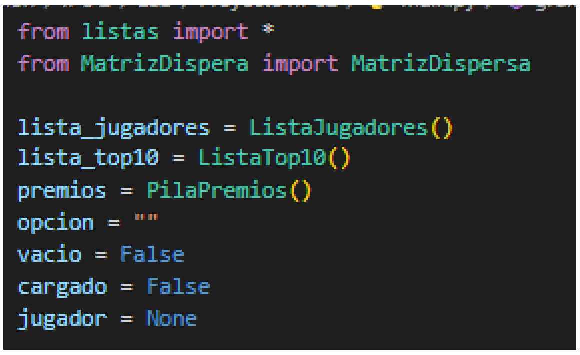
202111835

Guatemala 20 de diciembre del 2022

Main menu

It consists of a file where there is a cycle to display the menu and you can choose various options that you need depending on the value that the variable saves, a match case is used for each option. The first one loads the file calling the class “PlayersList” and empties the list of Top 10 players, for option 2 removes a player from the list and adds him to the Top 10, option 3 does the same, but with a loop to empty the entire list and add them to the top 10, the fourth option shows the figures from the input files of the next player in the queue, as long as they have data. The fifth option shows the top 10 on console, as well as the seventh option with the prizes, the sixth option adds these prizes and the eighth removes the last prize and the last player in the top 10 to simulate a delivery, the last one is the exit.

Imports and important variables:

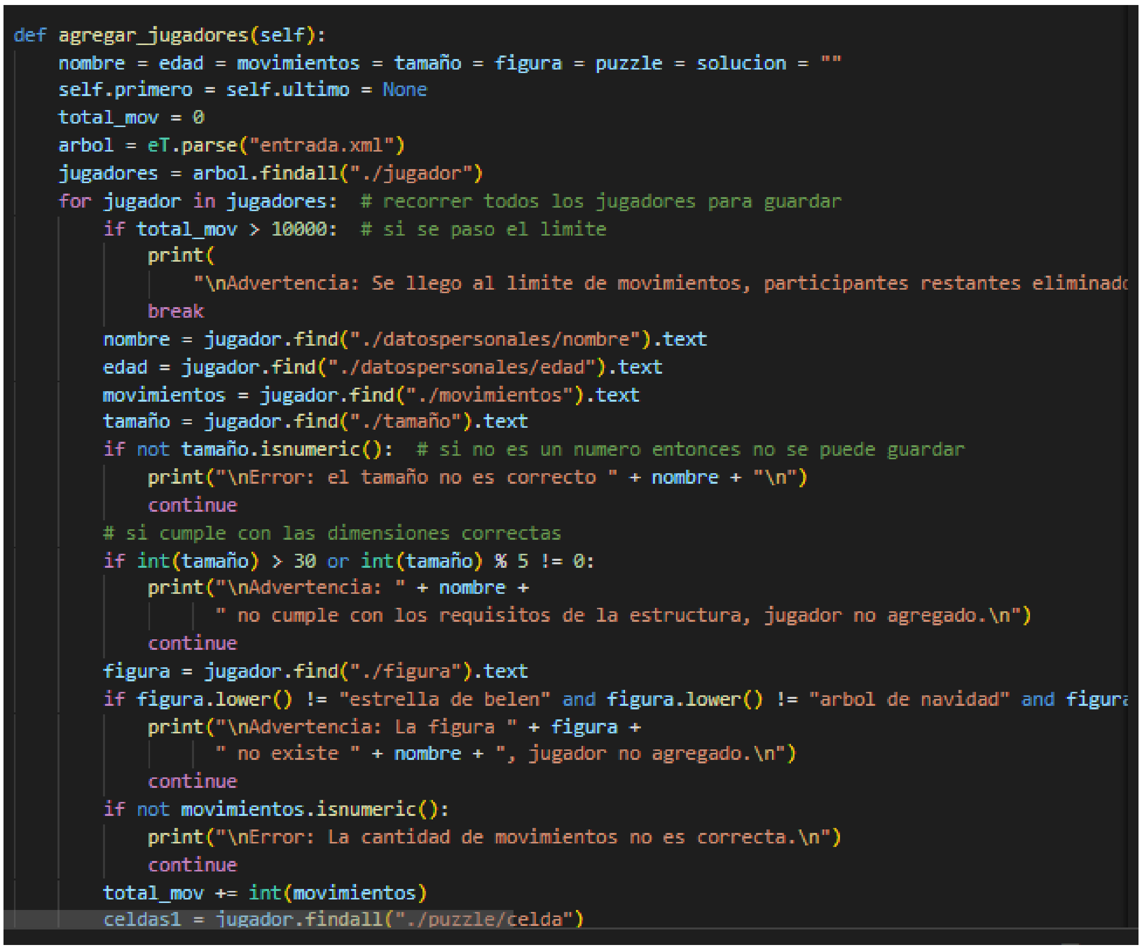


```python
from listas import *
from MatrizDispera import MatrizDispersa

lista_jugadores = ListaJugadores()
lista_top10 = ListaTop10()
premios = PilaPremios()
opcion = ""
vacio = False
cargado = False
jugador = None
```

Players List

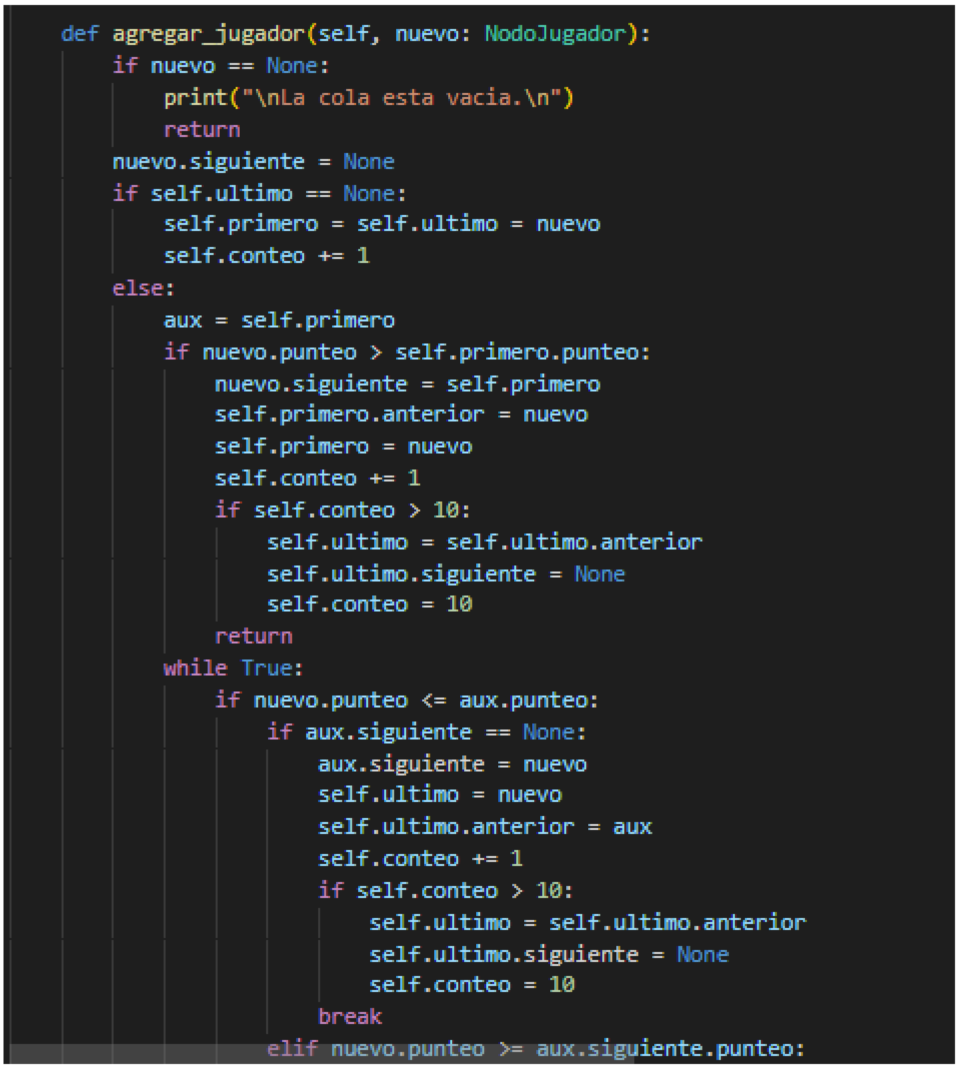
In this class, the players are added by reading the data from an XML file using the ElementTree class, first it looks for the tags that match "player" returning a list with all of them, a cycle is made to access one by one and in each iteration it is checked if it complies with the requirements, if it does not meet the requirements, it goes to the next one printing a non-compliance message, if it meets the requirements, it creates the object and calculates its score according to the criteria in the statement of the problem. Subsequently, if he is the first player to add the first and last variables, they are equal to this, if he is not the last player, he points to the new one and goes on to make the new last one. To carry out the behavior of a queue, remove the first player by saying that the first one is your next player.



```
def agregar_jugadores(self):
    nombre = edad = movimientos = tamaño = figura = puzzle = solucion = ""
    self.primero = self.ultimo = None
    total_mov = 0
    arbol = eT.parse("entrada.xml")
    jugadores = arbol.findall("./jugador")
    for jugador in jugadores:  # recorrer todos los jugadores para guardar
        if total_mov > 10000:  # si se paso el limite
            print(
                "\nAdvertencia: Se llego al limite de movimientos, participantes restantes eliminado
            break
        nombre = jugador.find("./datospersonales/nombre").text
        edad = jugador.find("./datospersonales/edad").text
        movimientos = jugador.find("./movimientos").text
        tamaño = jugador.find("./tamaño").text
        if not tamaño.isnumeric():  # si no es un numero entonces no se puede guardar
            print("\nError: el tamaño no es correcto " + nombre + "\n")
            continue
        # si cumple con las dimensiones correctas
        if int(tamaño) > 30 or int(tamaño) % 5 != 0:
            print("\nAdvertencia: " + nombre +
                  " no cumple con los requisitos de la estructura, jugador no agregado.\n")
            continue
        figura = jugador.find("./figura").text
        if figura.lower() != "estrella de belen" and figura.lower() != "arbol de navidad" and figura
            print("\nAdvertencia: La figura " + figura +
                  " no existe " + nombre + ", jugador no agregado.\n")
            continue
        if not movimientos.isnumeric():
            print("\nError: La cantidad de movimientos no es correcta.\n")
            continue
        total_mov += int(movimientos)
        celdas1 = jugador.findall("./puzzle/celda")
```

Top 10:

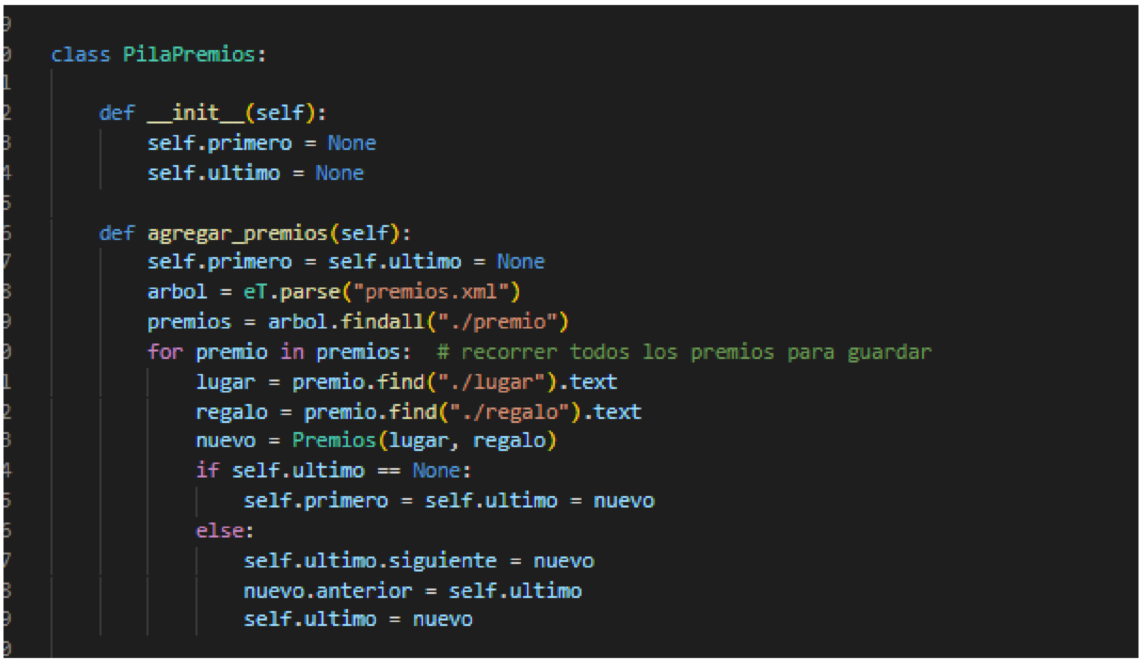
It adds a player in an ordered way from highest to lowest, for that it has a function that as a parameter is of the class "NodePlayer" it goes through a cycle where it checks if the current one is lower and its next one is higher, if it is fulfilled it adds it in this position of Otherwise, if the player count reaches 10, it does not add it, if it is less, it moves the last player. To make the awards ceremony, the last one must be eliminated saying that the last one is now the previous one that it is pointing to.



```python
    def agregar_jugador(self, nuevo: NodoJugador):
        if nuevo == None:
            print("\nLa cola esta vacia.\n")
            return
        nuevo.siguiente = None
        if self.ultimo == None:
            self.primero = self.ultimo = nuevo
            self.conteo += 1
        else:
            aux = self.primero
            if nuevo.punteo > self.primero.punteo:
                nuevo.siguiente = self.primero
                self.primero.anterior = nuevo
                self.primero = nuevo
                self.conteo += 1
                if self.conteo > 10:
                    self.ultimo = self.ultimo.anterior
                    self.ultimo.siguiente = None
                    self.conteo = 10
                return
            while True:
                if nuevo.punteo <= aux.punteo:
                    if aux.siguiente == None:
                        aux.siguiente = nuevo
                        self.ultimo = nuevo
                        self.ultimo.anterior = aux
                        self.conteo += 1
                        if self.conteo > 10:
                            self.ultimo = self.ultimo.anterior
                            self.ultimo.siguiente = None
                            self.conteo = 10
                        break
                    elif nuevo.punteo >= aux.siguiente.punteo:
```

Prize Stack:

Add the list of prizes from an XML file with the same method that is used in the list of players adding to the end always, to carry out the behavior of a stack, it removes the last prize added to the list saying that the last one is now its previous one.

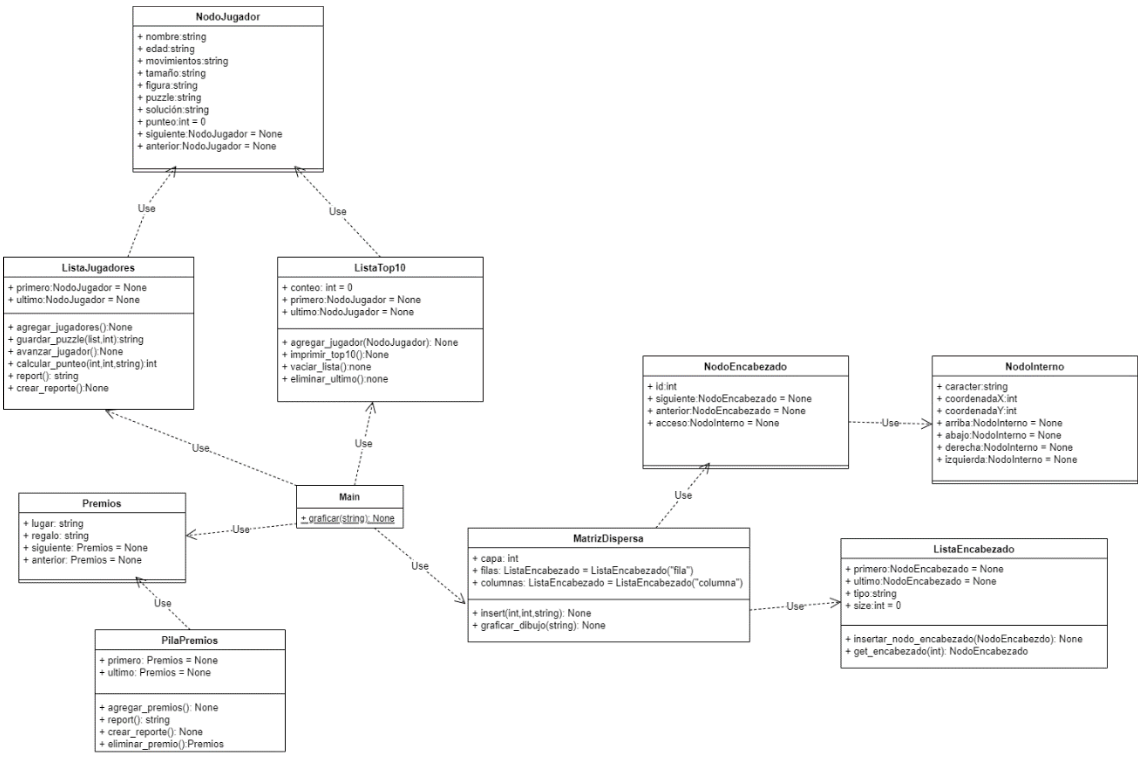


```python
class PilaPremios:

    def __init__(self):
        self.primero = None
        self.ultimo = None

    def agregar_premios(self):
        self.primero = self.ultimo = None
        arbol = eT.parse("premios.xml")
        premios = arbol.findall("./premio")
        for premio in premios:  # recorrer todos los premios para guardar
            lugar = premio.find("./lugar").text
            regalo = premio.find("./regalo").text
            nuevo = Premios(lugar, regalo)
            if self.ultimo == None:
                self.primero = self.ultimo = nuevo
            else:
                self.ultimo.siguiente = nuevo
                nuevo.anterior = self.ultimo
                self.ultimo = nuevo
```

class diagram


NodoJugador
+ nombre:string
+ edad:string
+ movimientos:string
+ tamaño:string
+ figura:string
+ puzzle:string
+ solución:string
+ punteo:int = 0
+ siguiente:NodoJugador = None
+ anterior:NodoJugador = None
Use
Use
ListaJugadores
+ primero:NodoJugador = None
+ ultimo:NodoJugador = None
+ agregar_jugadores():None
+ guardar_puzzle(list,int):string
+ avanzar_jugador():None
+ calcular_punteo(int,int,string):int
+ report(): string
+ crear_reporte():None
ListaTop10
+ conteo: int = 0
+ primero:NodoJugador = None
+ ultimo:NodoJugador = None
+ agregar_jugador(NodoJugador): None
+ imprimir_top10():None
+ vaciar_lista():none
+ eliminar_ultimo():none
Use
Use
Main
+ graficar(string): None
Use
Premios
+ lugar: string
+ regalo: string
+ siguiente: Premios = None
+ anterior: Premios = None
Use
PilaPremios
+ primero: Premios = None
+ ultimo: Premios = None
+ agregar_premios(): None
+ report(): string
+ crear_reporte(): None
+ eliminar_premio():Premios
Use
NodoEncabezado
+ id:int
+ siguiente:NodoEncabezado = None
+ anterior:NodoEncabezado = None
+ acceso:NodoInterno = None
Use
NodoInterno
+ caracter:string
+ coordenadaX:int
+ coordenadaY:int
+ arriba:NodoInterno = None
+ abajo:NodoInterno = None
+ derecha:NodoInterno = None
+ izquierda:NodoInterno = None
Use
MatrizDispersa
+ capa: int
+ filas: ListaEncabezado = ListaEncabezado("fila")
+ columnas: ListaEncabezado = ListaEncabezado("columna")
+ insert(int,int,string): None
+ graficar_dibujo(string): None
Use
ListaEncabezado
+ primero:NodoEncabezado = None
+ ultimo:NodoEncabezado = None
+ tipo:string
+ size:int = 0
+ insertar_nodo_encabezado(NodoEncabezdo): None
+ get_encabezado(int): NodoEncabezado