# TECHNICAL MANUAL

Diana Estefania Berducido Domingo

202000277

IPC2 – Laboratorio

Aux. Jackeline Benítez

# Programming used: OOP

• Through the implementation of data structures for simple linked lists and sparse arrays. These collect objects and data (players, winners, prizes, rows, columns) that it stores and mostly represents graphically.

## Objective of the Program

- The goal is to obtain two files with XML structure.
- The XML is read and the structures of simple linked lists are filled to execute the queues and stacks.
- The user has several options: Generate a simulated gift delivery on console, Generate the puzzles of all players while they withdraw from their queue graphically reflected and Generate a simulated gift delivery graphically where the start of the queue and stack and half of the execution are observed.
- The general objective is to calculate, according to the input files, the scores of the players and know who would be the first 10 places. These first places are given a prize and that delivery is reflected as a queue and a pile. In addition to being able to observe the figure in its messy and orderly puzzles.

#### ListaRegalos() proy1 +primero = None +ultimo = None +leer xml(ruta xml,ruta2 xml) +estaVacia(self) +generar puzzle2(ruta xml) +agregarUltimo(self, nombre, puntos) +generar puzzle(ruta xml) +agregarPrimero(self, nombre, puntos) +regalo +generar paracomun() +recorrido(self) +lugar +graphviz todos() +env nombres(self) +siguiente = None +entrega regalos() +eliminar primerol(self) +graphviz cola completa() lista simplee +eliminar solo uno(self) +graphviz cola media() +eliminar primero2(self) +graphviz pila completa() +eliminar ultimo(self) +graphviz pila mitad() +entrega oficial() ListaGanador() MatrizDispersa +primero = None +capa +ultimo = None +insert(pos x,pos y,caracter) +estaVacia(self) +graficarArbol(nombre) +agregarUltimo(self, nombre, puntos) +nombre +graficarRegalo(nombre) +agregarPrimero(self, nombre, puntos) +puntos +graficarEstrella(nombre) +recorrido(self) +siguiente = None +graficarDot(nombre) +env nombres(self) +env medio nombres(self) +eliminar primerol(self) +eliminar solo uno(self) +eliminar primero2(self) **Nodo Interno** +eliminar ultimo(self) +eliminar ult(self) +ordenar menor mayor(self) +ordenar menor mayor(self) +caracter +eliminar mitad(self) +arriba = None +abajo = None +derecha = None ListaJugador() +izquierda = None +primero = None +ultimo = None +estaVacia(self) +agregarUltimo(self,nombre,puntos) +nombre +agregarPrimero(self, nombre, puntos) +puntos +recorrido(self) +figura +env nombres(self) +tamaño +env medio nombres(self) +movimientos +eliminar mitad(self) +siquiente = None +eliminar primero(self) +ordenar menor mayor(self) +ordenar menor mayor(self) +eliminar mitad(self) +buscar(self,nombre)

Class

Diagram

Premio()

Ganador()

Jugador()

+env primeros10(self) +eliminar solo uno(self)

# Proy1 (MAIN)

After the declaration of several methods that help graph (next slides)
has the start menu, where it asks for the input files and asks to select
an action:

```
print("-----;Bienvenid@ al juego de Puzzle!-----")
global ruta
ruta = input("-----Por favor, ingrese el nombre del archivo XML de jugadores: ")
ruta2 = input("-----Por favor, ingrese el nombre del archivo XML de premios: ")
a = leer_xml(f"{ruta}", f"{ruta2}")
regalosss.recorrido()
#graphviz_entrega()
while True:
   print("Ingrese el número de la opción que desea ejecutar:")
   print("1. Simular Entrega de Regalos en Consola.") #listo
   print("2. Ver cola de participantes e ir generando su puzzle mientras se retiran de la cola.")
   print("3. Generar gráficos de Entrega de Regalos.") #listo
   print("4. Salir.")
   opcion = input("Opción: ")
   if opcion == "1": ···
   elif opcion == "2": ···
   elif opcion == "3": ···
   elif opcion == "4": ···
```

# Proy1 (MAIN)

• From here the other classes are called to execute their functions. The functions in this file are:

• And, all are responsible for the part of bringing data from the lists to graph them or display them in console.

```
proy1.py > ...
      from os import system
      from lista_simplee import ListaGanador
      from lista simplee import ListaJugador
      from lista simplee import ListaRegalos
      from MatrizDispersa import MatrizDispersa
  7 > def leer_xml(ruta_xml, ruta2_xml): ...
    > def generar_puzzle2(ruta_xml): ...
238
239 > def generar_puzzle(ruta_xml): ...
336 > def generar paracomun(): ...
370 > def graphviz_todos(): ...
389 > def entrega_regalos(): ...
430
431 > def graphviz_cola_completa():...
449
450 > def graphviz_cola_media(): ...
469 > def graphviz_pila_completa(): ...
489 > def graphviz pila mitad(): ...
```

### XML reading

- In this case, the reading was done through ElementTree.
- In the screenshots you can see the structure of the file.

```
jug.xml
      <?xml version="1.0" encoding="UTF-8"?>
      <jugadores>
          <jugador>
              <datospersonales>
                  <nombre>Felix6</nombre>
                  <edad>20</edad>
              </datospersonales>
              <movimientos>10</movimientos>
              <tamaño>5</tamaño>
              <figura>Arbol de Navidad</figura>
              <puzzle>
                  <celda f="1" c="0" />
                  <celda f="1" c="4" />
                  <celda f="1" c="3" />
                  <celda f="2" c="3" />
                  <celda f="3" c="4" />
                  <celda f="4" c="1" />
                  <celda f="4" c="2" />
                                                  134
                  <celda f="4" c="3" />
                  <celda f="4" c="4" />
                  <celda f="0" c="2" />
              </puzzle>
                                                  200
              <solucion>
                  <celda f="1" c="2" />
                  <celda f="2" c="1" />
                                                  233
                  <celda f="2" c="2" />
                  <celda f="2" c="3" />
                  <celda f="3" c="0" />
                  <celda f="3" c="1" />
                                                  299
                  <celda f="3" c="2" />
                  <celda f="3" c="3" />
                  <celda f="3" c="4" />
                   celds f-"4" c-"2" /\
```

```
jug.xml
       <?xml version="1.0" encoding="UTF-8"?>
       <jugadores>
           </jugador>
           </jugador>
           </jugador>
102 >
           <jugador>...
           </jugador>
135 >
           </jugador>
168 >
           </jugador>
201 >
           <jugador>...
           </jugador>
234 >
           </jugador>
267 >
           </jugador>
300 >
           <jugador>...
           </jugador>
333 >
           <jugador>...
365
           </jugador>
           </jugador>
       k/jugadores>
```

## XML reading

 The data was listed, and later traversed to store the xm data in the structures. this case first the structure of players was filled the after and calculation of points the filling was made to the structure of winners, in the same way with the gifts.

```
def leer_xml(ruta_xml, ruta2 xml):
    import xml.etree.ElementTree as ET
    try:
        #variable para cada jugador
        i = 1
        #variables para sumar puntos
        global puntos
        #lectura de ruta xml (jugadores)
        ruta xml = ruta xml
        xml_file = open(ruta xml, encoding="utf-8-sig")
        global jug
        global jug comun
        jug = ListaGanador()
        #Esta va a guardarlos todos, pero se van a procesar ganadores y así con la otra lista
        #esta lista va a servir también para hacer búsquedas y eso
        jug comun = ListaJugador()
        global jug consultas
        jug consultas = ListaJugador()
        if xml file.readable():
            datos = ET.fromstring(xml file.read())
            lista jugadores = datos.findall("jugador")
            print("Total de jugadores:", len(lista_jugadores))
            for jugador in lista jugadores:
                puntos = 0
                lista datos = jugador.findall("datospersonales")
                lista_movimientos = jugador.findall("movimientos")
                lista tamaño = jugador.findall("tamaño")
                lista_figura = jugador.findall("figura")
                lista puzzle = jugador.findall("puzzle")
                lista solucion = jugador.findall("solucion")
                for datos in lista datos:
```

### lista\_simplee.py: Linked lists

• File in charge of creating lists for the different objects: common player, winning player and gifts or prizes.

```
lista_simplee.py > ListaRegalos
from ganador import Ganador
from jugador import Jugador
from premio import Premio

class ListaGanador(): ...

class ListaRegalos(): ...

class ListaRegalos(): ...

class ListaJugador(): ...
```

• They have similar methods but are used in different ways. They will be seen on the next slide.

### lista\_simplee.py: Linked Lists

• Among its methods are add last or first (either queue or stack), delete (there is to delete only the first, for example), go through, send to another list ...

```
class ListaGanador():
    def init (self):
    def estaVacia(self): ...
    def agregarUltimo(self,nombre,puntos): ...
    def agregarPrimero(self, nombre, puntos):
    def recorrido(self): ...
    def env_nombres(self): ...
    def env_medio_nombres(self): ...
    def eliminar primero1(self): ...
    def eliminar_solo_uno(self): ...
    def eliminar primero2(self): ...
    def eliminar ultimo(self): ...
    def eliminar_ult(self): ...
    def ordenar menor mayor(self): ...
    def ordenar mayor menor(self): ...
    def ordenar_may_men(self): ...
    def eliminar_mitad(self): ...
```

### lista\_simplee.py: Linked Lists

• Among its methods are add last or first (either queue or stack), delete (there is to delete only the first, for example), go through, send to another list ...

```
class ListaRegalos():
    def init (self): ···
    def estaVacia(self): ...
    def eliminar_solo_uno(self): ...
    #si viene de 10 a 1
    def agregarUltimo(self,regalo,lugar): ...
    #si viene de 1 a 10
    def agregarPrimero(self, regalo, lugar): ...
    def recorrido(self): ...
    def eliminar_primero1(self): ...
    def eliminar_primero2(self): ...
    def env_nombres(self): ...
    def eliminar_ultimo(self): ...
class ListaJugador():
    def __init__(self): ···
    def estaVacia(self): ···
```

### lista\_simplee.py: Linked Lists

 Among its methods are add last or first (either queue or stack), delete (there is to delete only the first, for example), go through, send to another list ...

```
class ListaJugador():
    def init (self): ...
    def estaVacia(self): ...
   def agregarUltimo(self,nombre,puntos, figura, tamaño, movimientos):
   def agregarPrimero(self, nombre, puntos, figura, tamaño, movimientos):
    def eliminar solo uno(self): ...
   def recorrido(self): ...
   def env nombres(self): ...
   def env_primeros10(self): ...
    def env medio nombres(self): ...
   def eliminar primero(self): ...
    def eliminar mitad(self): ...
    #esta es la que funciona para el caso
    def ordenar_menor_mayor(self): ...
    #supongo que igual debería recorrer y luego ordenarlo
    def ordenar men may(self): ...
   def ordenar_mayor_menor(self): "
   def ordenar_may_men(self): ...
   def buscar(self, nombre): ...
```

# List objects

They are the objects that handle the structures mentioned above.

```
ganador.py > Ganador > _init_

class Ganador():

def __init__(self, nombre, puntos):

self.nombre = nombre

self.puntos = puntos

self.siguiente = None
```

```
premio.py > Premio > __init__

class Premio():

def __init__(self, regalo, lugar):

self.regalo = regalo

self.lugar = lugar

self.siguiente = None
```

```
jugador.py > Paguador > Data __init__

class Jugador():

def __init__(self, nombre, puntos, figura, tamaño, movimientos):

self.nombre = nombre

self.puntos = puntos

self.figura = figura

self.tamaño = tamaño

self.movimientos = movimientos

self.siguiente = None
```