Informe de Laboratorio 04

Tema: Sort y Listas Enlasadas

Tabla 1: Mi tabla extendida

INTRODUCCIÓN
aqui ira la intro
MARCO CONCEPTUAL
aqui ira la MARCO CONCEPTUAL
SOLUCIONES Y PRUEBAS

Ejercicio 1

//taylor

Ejercicio 2

Para lograr ejecutar el algoritmo de ordenamiento de insercion primero fue necesario crear la clase ListaDoble el cual tenia 2 nodos, uno inicial y otro final. Con esto se lograba recorrer la lista de el fin al inicio y en viceverza.

El metodo addFinal permitia insertar un nodo al final y el otro metodo addInicio al inicio,dependeiendo si es que estaba vacia o no la lista. Asismimo se agrego el metodo isEmpty para saber si estaba vacia y se sobreescribio el metodo toString para mostrar retornar la data del 1er al ultimo dato.

```
public void addInicio(E x){
    if(!isEmpty()){
        inicio=new Node<>(x,inicio,null);
        inicio.getNextNode().setPreviousNode(inicio);
    }else {
        inicio=fin= new Node<>(x);
    }
}
```

```
public void addFinal(E x){
    if(!isEmpty()){
        fin=new Node<>(x,null,fin);
        fin.getPreviousNode().setNextNode(fin);
    }else {
        inicio=fin= new Node<>(x);
    }
}
```

```
public boolean isEmpty() {
    return inicio == null;
}
//getters y setters de inicio y fin
@Override
public String toString() {
    String text="";
    if (!isEmpty()){
        Node<E> aux = inicio;
        while (aux!=null){
            text += aux.getData()+" ";
            aux = aux.getNextNode();
        }
    }
    return  text;
}
```

```java
public void addInicio(E x){
    if(!isEmpty()){
        inicio=new Node<>(x,inicio,null);
        inicio.getNextNode().setPreviousNode(inicio);
    }else {
        inicio=fin= new Node<>(x);
    }
}
```

```java
public void addFinal(E x){
    if(!isEmpty()){
        fin=new Node<>(x,null,fin);
        fin.getPreviousNode().setNextNode(fin);
    }else {
        inicio=fin= new Node<>(x);
    }
}
```

```java
public boolean isEmpty() {
    return inicio == null;
}
//getters y setters de inicio y fin
@Override
public String toString() {
    String text="";
    if (!isEmpty()){
        Node<E> aux = inicio;
        while (aux!=null){
            text += aux.getData()+" ";
            aux = aux.getNextNode();
        }
    }
    return  text;
}
```

LECCIONES APRENDIDAS Y CONCLUSIONES
aqui ira la LECCIONES APRENDIDAS Y CONCLUSIONES
REFERENCIAS Y BIBLIOGRAFÍA
aqui ira la REFERENCIAS Y BIBLIOGRAFÍA

0.1. USAR COMO GUIA PARA EL INFORME

- Se tuvieron que implementar las funciones negative, join y under. En el codigo se detalla lo que hacen.

Listing 1: Picture version 1

```python
import colors
from colors import *
class Picture:
    def __init__(self, img):
        self.img = img;
    def __eq__(self, other):
        return self.img == other.img
    def _invColor(self, color):
        if color not in inverter:
            return color
        return inverter[color]
```

```
12      def negative(self):
13          """ Devuelve un negativo de la imagen """
14          neg = []
15          for fila in self.img:
16              cadena = ""
17              for color in fila:
18                  cadena += self._invColor(color)
19              neg.append(cadena)
20          return Picture(neg)
21      def join(self, p):
22          """ Devuelve una nueva figura poniendo la figura del argumento
23              al lado derecho de la figura actual """
24          juntos = []
25          for i in range(len(self.img)):
26              juntos.append(self.img[i] + p.img[i])
27          return Picture(juntos)
28      def under(self, p):
29          """ Devuelve una nueva figura poniendo la figura p sobre la
30                  figura actual """
31          return Picture(self.img[::] + p.img[::])
```

Para implementar el negativo se cambia el color de cada carácter, a excepción del espacio ya que este no tiene inverso con doble for(1 anidado). Juntando estos caracteres por medio en una cadena y despues juntando esta cadena al nuevo arreglo con el metodo append.

Listing 2: Ejercicio 2 a

```
1 from interpreter import draw
2 from chessPictures import *
3
4 draw(knight.join(knight.negative()).under(knight.negative().join(knight)))
```

Para poder hacer que se dibuje como se mostrara el la figura de abajo se tuvo de juntar un caballo blanco con uno negro en la primera fila; para ello se utilizo join y negative. Para la segunda fila se necesito saltar a la siguiente fila por lo que se utilizo el metodo under.

Figura 1: Ejecución exitosa ejercicio 2 a

0.2. Pregunta: Explique: ¿Para qué sirve el directorio pycache?

- Sirve para guardar los compilados de python, asi como en java al ejecutarlos se crean los .class en python se crean los .pyc. Esto se hace automaticamente ya que se importan modulos de otras clases como se da en este caso.