| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| Estructura de Datos | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

# Actividad en Contacto con el Docente

## TAREA DE UNIDAD III

**Indicaciones:**

Este documento fue desarrollado con la finalidad de guiar al estudiante en las tareas de los contenidos tratados en la unidad III, posee una serie de instrucciones que el estudiante debe aplicar para reforzar los conocimientos profesionales correspondiente al nivel de estudio de la carrera.

**Objetivos**

El trabajo de esta asignatura consistirá en desarrollar su capacidad de conceptualización para complementar los temas tratados en la Unidad No.3.

**Descripción de la actividad y pautas de elaboración**

- Para realizar esta tarea deberá consultar el cronograma de actividades del aula virtual.
- La tarea es individual y por lo tanto cada estudiante asumirá la responsabilidad de autoría, si existen dos a más trabajos iguales, la calificación será de cero (0) para el número de alumnos que incurran en esta falta.
- En caso de que el estudiante tenga dudas en los temas tratados en clase, podrá utilizar las diapositivas, compendio y videos que se adjuntan en el aula virtual.
- Queda prohibido el uso de librerías no estudiadas en clases, tales como StringBuilder, Stack, ArrayList entre otras, de hacerlo su nota será de cero (0).
- Adjuntar carpeta comprimida con los códigos de programación

| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

**Actividades por desarrollar**

La Facultad de Ciencias Informática ha realizado una evaluación para cada estudiante que aspira un cupo en esta Facultad.

De cada Estudiante la facultad tiene el código único del postulante (de tipo entero), el nombre (String) y el puntaje sobre 100 puntos (puede ser float)

Estos datos se registran en una lista doble, cuando el proceso de registro finaliza, se deberá seleccionar a los primeros 50 estudiantes que obtuvieron un puntaje de postulación superior a los 75 puntos (nota tope), en caso de no cubrir los 50 cupos deberá completar los 50 cupos con los estudiantes con los puntajes más próximos a la nota tope.

**Se requiere:**

- **Procedimiento Insertar**: que genere una lista de estudiantes con 75 códigos de estudiantes (códigos consecutivos de 1, 2, 3 hasta llegar a los 75 que sería el último estudiante ingresado) y así mismo se genera una puntuación lograda (genere las calificaciones de forma aleatoria) en un rango de valor entre 20 y 100 puntos.
- Genera la selección de los 50 postulantes admitidos.
- Muestre todos los postulantes generados (Código único, Nombre y puntaje)
- Muestre la selección de los 50 cupos (Código único, Nombre y puntaje)

| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

Realizar pruebas y validar mi solución propuesta

**¿Por qué elegí esta estructura para conseguir la solución al problema?**

Para esta tarea elegí una lista doblemente enlazada con su NodoDoble porque me permitía recorrer la lista en ambos sentidos, aunque en la práctica solo la recorrí del inicio al final. Aun así, dejé los métodos porque más adelante podría mejorarla o ampliarla.
También sentí que al hacerla doble iba a aprender más y aplicar lo enseñado por el profesor, por lo que ese fue otro motivo para elegirla.

**Explicación de la creación de la lista, el Nodo y la entidad**

Primero empecé con la creación de la entidad, que en este caso se llama **Estudiante**.
La tarea me pedía que tuviera tres datos:

- un identificador,
- su nombre,
- y el puntaje de postulación.

En mi caso, el identificador lo creé con una variable llamada id de tipo int.
El otro atributo lo llamé nombre de tipo String.
Y el último atributo lo puse como puntaje de tipo float.

Los tres atributos los establecí como private.
Después creé los constructores (vacío y con datos), los métodos **getters** y **setters**, y sobreescribí el método toString() para mostrar bien los datos que pedía la tarea.

```
package com.tyrone.tarea6;

/**
 *
 * @author tyron
 */
public class Estudiante {
    private int id;
    private String nombre;
    private float puntaje;


    public Estudiante() {
```

```
}

public Estudiante(int id, String nombre, float puntaje) {
    this.id = id;
    this.nombre = nombre;
    this.puntaje = puntaje;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public float getPuntaje() {
    return puntaje;
}

public void setPuntaje(float puntaje) {
    this.puntaje = puntaje;
}
```

```
    @Override
    public String toString() {
        String nombreLimpio = (nombre.charAt(0) == ' ') ? nombre.substring(1)
: nombre;
        return "{" + "id= " + id + ", nombre= " + nombreLimpio + ", puntaje=
" + puntaje + '}';
    }



}
```

Ahora que ya tengo la entidad que contendrá los datos, procedí a crear el Nodo.
En mi caso escogí el Nodo doble por la practicidad que mencioné antes. Primero creé la clase con el nombre **NodoDoble**, luego creé los atributos **ante** y **sigue** del mismo tipo de clase, y otro atributo de nombre **dato** que contiene un objeto Estudiante.

Todos los atributos son **private**.
Luego hice los constructores (vacío y con datos) y los getters y setters.

```
package com.tyrone.tarea6;


/**
 *
 * @author tyron
 */
public class NodoDoble {
    private NodoDoble ante;
    private Estudiante dato;
    private NodoDoble sigue;

    public NodoDoble() {
    }

    public NodoDoble(NodoDoble ante, Estudiante dato, NodoDoble sigue) {
        this.ante = ante;
```

```
        this.dato = dato;
        this.sigue = sigue;
    }

    public NodoDoble getAnte() {
        return ante;
    }

    public void setAnte(NodoDoble ante) {
        this.ante = ante;
    }

    public Estudiante getDato() {
        return dato;
    }

    public void setDato(Estudiante dato) {
        this.dato = dato;
    }

    public NodoDoble getSigue() {
        return sigue;
    }

    public void setSigue(NodoDoble sigue) {
        this.sigue = sigue;
    }


}
```

Ya teniendo las dos clases, procedí a crear la clase **ListaDoble**, donde implementé los métodos característicos.
Primero creé dos atributos llamados inicio y fin, ambos de tipo NodoDoble.
Luego puse los constructores y a continuación los métodos principales.

El primero que uso es **ListaVacia()**, que devuelve un boolean dependiendo si la variable inicio es nula.

```
package com.tyrone.tarea6;


/**
 *
 * @author tyron
 */
public class ListaDoble {
    //atributos
    private NodoDoble inicio;
    private NodoDoble fin;

    //constructor

    public ListaDoble() {
    }

    public ListaDoble(NodoDoble inicio, NodoDoble fin) {
        this.inicio = inicio;
        this.fin = fin;
    }

    //Metodos
    //Lista vacia
    public boolean ListaVacia(){
        return inicio == null;
    }
```

Luego creé el método llamado **InsertInicio()**, en el cual primero genero un nuevo nodo llamado **nuevo**. Ese nodo lo creo llamando al constructor **NodoDoble(null, dato, null),** lo cual significa que inicialmente sus dos extremos (el anterior y el siguiente) están en **null**, y en la parte del centro le paso la variable **dato**, que en este caso son los datos del Estudiante.

Después de crear este nodo, hago una validación con un **if** usando el método **ListaVacia()** para saber si la lista está vacía. Si la lista está vacía, entonces simplemente hago que **inicio** sea igual a **nuevo** y también **fin** sea igual a **nuevo**, porque cuando hay un solo elemento, inicio y fin apuntan al mismo nodo.

Si la lista **no** está vacía (o sea, entra al else), entonces tengo que enlazar manualmente los extremos del nuevo nodo con el inicio actual. Para eso realizo los siguientes pasos:

1. Con **nuevo.setDato(dato)** me aseguro de colocar el dato del estudiante dentro del nodo nuevo.
2. Luego, con **nuevo.setSigue(inicio)** hago que el nodo nuevo apunte al nodo que antes era el inicio.
3. Después, con **inicio.setAnte(nuevo)** le digo al nodo que antes era el inicio que su anterior ahora sea el nodo nuevo.
4. Finalmente asigno **inicio = nuevo**, lo que significa que el nodo nuevo ahora es oficialmente el primer nodo de la lista básicamente lo actualizo.

```
public void InsertInicio(Estudiante dato){
    NodoDoble nuevo = new NodoDoble(null, dato, null);
    if (ListaVacia()) {
        inicio = nuevo;
        fin = nuevo;
    } else {
        nuevo.setDato(dato);
        nuevo.setSigue(inicio);
        inicio.setAnte(nuevo);
        inicio = nuevo;
    }
}
```

También hice los métodos InsertatFinal() que es similar al método InsertInicio() por lo que no lo explicare, EliminarFinal() y EliminarInicio() esos tampoco los explicare porq no los ocupe en la tarea.

```
public void InsertFinal(Estudiante dato){
      NodoDoble nuevo = new NodoDoble(null, dato, null);
      if (ListaVacia()) {
         inicio = nuevo;
         fin = nuevo;
      } else {
         fin.setSigue(nuevo);
         nuevo.setAnte(fin);
         fin = nuevo;
      }
   }

   // Eliminar inicio o final
   public String EliminarFinal(){
      if (ListaVacia()) {
         return "Lista vacia";
      } else {
         String n = fin.getDato().getNombre();
         fin = fin.getAnte();
         fin.setSigue(null);
         return n;
      }
   }

     public Estudiante EliminarInicio(){
      if (ListaVacia()) {
         System.out.println("Lista vacia");
         return null;
      } else {
```

```
        Estudiante n = inicio.getDato();
        inicio = inicio.getSigue();
        inicio.setAnte(null);
        return n;
    }
}
```

Posteriormente creé el método **Imprimir()**, en el cual primero verifico si la lista está vacía usando un if con el método **ListaVacia()**.
Si está vacía, entonces simplemente muestro un mensaje por consola indicando que la lista está vacía.

Pero si no está vacía, entonces procedo a recorrerla.
Para eso creo un nodo auxiliar llamado **aux** y le asigno el valor de inicio, para comenzar el recorrido desde el primer nodo.

Luego utilizo un **while** que seguirá ejecutándose mientras **aux** sea diferente de null. Dentro del ciclo:

- imprimo los datos del postulante usando **aux.getDato()**
- luego actualizo el auxiliar haciendo **aux = aux.getSigue(),** lo cual permite recorrer nodo por nodo hasta que finalmente llegue al final donde será null y terminará el ciclo.

```
public void Imprimir(){
        if (ListaVacia()) {
            System.out.println("Lista esta vacia");
        } else {
         NodoDoble aux = inicio;
            while(aux != null){
                System.out.println("Postulante " + aux.getDato());
                aux = aux.getSigue();

            }

        }
    }
```

| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

**Parte de la explicación del método de ordenamiento y la selección de 50 alumnos**

Aquí implementé una lógica para ordenar los datos sin alterar la lista original. Para eso creé el método **ListaPuntajeAscendente()** que devuelve un objeto de tipo ListaDoble, porque necesito retornar la lista ya ordenada.

Primero creo una ListaDoble llamada **copia**, que utilizaré para guardar la copia exacta de todos los estudiantes sin modificar nada de la lista original.

Después verifico con un if usando **ListaVacia**() si la lista está vacía. Si está vacía, imprimo un mensaje. Pero si no está vacía, procedo con la lógica.

Primero declaro un nodo auxiliar llamado **aux** y le asigno el inicio de la lista actual.
Luego hago un while que recorre toda la lista original desde el inicio hasta que **aux** sea null.
En cada vuelta del ciclo:

- uso **copia.InsertFinal(aux.getDato())** que inserta al final de la lista copia el dato del estudiante
- después actualizo el auxiliar con **aux = aux.getSigue()**

De esa forma, al finalizar ese **while**, tengo una lista copia con exactamente los mismos elementos que la lista original pero sin alterar nada en ella.

Luego viene la parte del ordenamiento.
Para ordenar la lista usé dos nodos:

- **nodoComparacion1**

  Toma el valor de **copia.inicio** y servirá para ir comparando elemento por elemento.

- **nodoComparacion2**

  Este nodo toma el valor **de nodoComparacion1.getSigue()** y sirve para comparar el nodo actual con todos los que vienen después.

Primero hago un **while** para que nodoComparacion1 recorra toda la lista.
Dentro de ese while creo nodoComparacion2 para que empiece desde el siguiente nodo hacia adelante.

Dentro del segundo **while** comparo los puntajes.
La condición es:

> Si el puntaje de nodoComparacion2 es mayor que el de nodoComparacion1, entonces intercambio los datos.

Para hacer ese intercambio:

1. guardo el dato de nodoComparacion1 en una variable temporal tipo Estudiante llamada **temp**.
2. luego asigno a nodoComparacion1 el dato de nodoComparacion2.
3. finalmente asigno a nodoComparacion2 el dato de **temp**.

Así se hace un intercambio básico de valores.

Luego actualizo nodoComparacion2 con **getSigue()** para que siga recorriendo su propio ciclo. Al terminar ese ciclo, actualizo nodoComparacion1 también con **getSigue().**

Finalmente retorno la lista copia ya ordenada.

```
public ListaDoble ListaPuntajeAscendente() {

        ListaDoble copia = new ListaDoble();

        if (ListaVacia()) {
           System.out.println("Lista vacía");

        }else{

           NodoDoble aux = inicio;

           while (aux != null) {
              copia.InsertFinal(aux.getDato());
              aux = aux.getSigue();
           }


           NodoDoble nodoComparacion1 = copia.inicio;

           while (nodoComparacion1 != null) {
              NodoDoble nodoComparacion2 = nodoComparacion1.getSigue();

              while (nodoComparacion2 != null) {
```

```
                if (nodoComparacion2.getDato().getPuntaje() >
nodoComparacion1.getDato().getPuntaje()) {
                    Estudiante temp = nodoComparacion1.getDato();
                    nodoComparacion1.setDato(nodoComparacion2.getDato());
                    nodoComparacion2.setDato(temp);
                }
                nodoComparacion2 = nodoComparacion2.getSigue();
            }

            nodoComparacion1 = nodoComparacion1.getSigue();
        }
    }
    return copia;
}
```

Con el método anterior ya tengo una lista ordenada, por lo que ahora creé el método **SelecTop50MejAlumnosEnCalif()** para escoger los 50 estudiantes con puntajes mayores o iguales a 75 y, si no alcanza, completar con los siguientes más cercanos al puntaje tope.

Primero verifico si la lista está vacía.
Si no está vacía:

- creo una ListaDoble llamada **seleccionados**
- creo otra ListaDoble llamada **ordenada** que captura el retorno de **ListaPuntajeAscendente()**
- creo un NodoDoble llamado **aux** que apunta al inicio de la lista ordenada
- creo dos variables:
    - cupos = 50
    - cuposActuales = 0

Luego uso un ciclo for desde **i = 1** hasta lo q contenga la variable **cupos**.
Uso **i** desde 1 para ir numerando los postulantes aceptados.

Dentro del **for** hago una condición:

**Si el puntaje de aux es mayor o igual a 75**

Entones lo inserto a la lista seleccionados, actualizo cuposActuales = cupos - 1 e imprimo el dato de ese aux.

**Si el puntaje es menor a 75**

También lo inserto porque la tarea manda a completar los cupos con los más cercanos. Actualizo cuposActuales e imprimo, básicamente hago lo mismo que lo que tiene la primera condición.

**Si no se cumplen las condiciones (algo raro o cantidad de postulante menor a 50)**

Muestro cuántos cupos sobraron.

Al final de cada iteración actualizo aux con **aux.getSigue().**

Y aunque tenga esa lista con esos postulantes seleccionados, no la retornare y dejare la lógica como esta ahora, pero en un futuro cuando vuelva a descargar mis apuntes practicaré y cambiare el tipo de dato de este método y retornare la lista

```
public void SelecTop50MejAlumnosEnCalif(){

        if (ListaVacia()) {
            System.out.println("Lista vacía");

        }else{
            ListaDoble seleccionados = new ListaDoble();
            ListaDoble ordenada = ListaPuntajeAscendente();
            NodoDoble aux = ordenada.inicio;

            int cupos = 50, cuposActuales=0;



            for (int i = 1; i <= cupos; i++) {
                if (aux.getDato().getPuntaje()>=75) {

                    seleccionados.InsertFinal(aux.getDato());
                    cuposActuales = cupos-1 ;
                    System.out.println(i+"~ Estudiante "+ aux.getDato());

                } else if (aux.getDato().getPuntaje()<75) {
```

```
                seleccionados.InsertFinal(aux.getDato());
                cuposActuales = cupos-1 ;
                System.out.println(i+"~ Estudiante "+ aux.getDato());

            } else{
                System.out.println("sobraron"+ cuposActuales + "  cupos");
            }
            aux = aux.getSigue();

        }

    }


}
```

## Cómo ocupo y funciona mi solución

Uso este código en mi clase Main llamada MainPuntajeMejoresPostulacion.

Primero creo una ListaDoble llamada **listaDoble**.
Luego muestro un mensaje por consola simulando el registro de datos.

Después uso un ciclo **for** que se repetirá 75 veces, desde i = 1 hasta i = 75.
Lo puse así porque la tarea pide 75 códigos con datos de estudiantes y además necesito que el ID vaya del 1 al 75 y por eso inicie **i** en 1 para poder usarlo.

Dentro del for:

1. **Creo el puntaje aleatorio**
    - Uso Math.random() multiplicado por 8000 y dividido para 100.0
    - Esto me da un número con decimales
    - Lo redondeo con Math.round
    - Y le sumo 20 para asegurar que no baje de ese valor
2. **Simulo nombres**
   Hice un String muy largo llamado nombres.
   Para sacar una parte de ese string uso substring, pero para eso necesito índices válidos.
   Creo la variable indice usando Math.random() multiplicado por la longitud del string

menos 12 caracteres (para no salirme del rango cunado lo sume posteriormente).
Le sumo i para intentar que no se repitieran, pero al final eso no importa mucho porque en el toString se limpia el espacio inicial y el índice al caer en un espacio se nota como diferente pero al eliminar ese espacio en el tosTring no retorna un valor igual a un numero anterior o posterior.

Luego creo el nombre con:

nombres.substring(indice, indice + 12)

3. **Creo un objeto Estudiante con estas variables**

   Estudiante estudiante = new Estudiante(i, nombre.toUpperCase(), puntaje);

4. **Lo inserto a la lista**

   listaDoble.InsertInicio(estudiante);

5. **Simulo la barra de carga**

   System.out.print("=");

Cuando el ciclo llega a 75 muestras el mensaje “Datos cargados con éxito”.

Después imprimo un mensaje pro consola “Todos los postulantes” y a todos los postulantes usando el método Imprimir(), y finalmente muestro por sonsola el mensaje "Selección de los 50 cupos:" y llamo al método SelecTop50MejAlumnosEnCalif() para mostrar quienes alcanzaron estos cupos.

```
package com.tyrone.tarea6;


/**
 *
 * @author tyron
 */
public class MainPuntajeMejoresPostulacion {


    public static void main(String[] args) {
        ListaDoble listaDoble = new ListaDoble();


        System.out.println("Procesando registros de datos...");


        for (int i = 1; i <= 75; i++) {
```

```
            float puntaje = (float)((Math.round(Math.random() * 8000) / 100.0)+ 20.0);

            String nombres = "Tyrone Joel Juan Gabriel Pedro Luis Carlos Maria Elena Pepa Sofia Alejandra
Ana";
            int indice = ((int)(Math.random() * (nombres.length() - 12)) + i) % (nombres.length() - 12);

            String nombre = nombres.substring(indice, indice + 12);

            Estudiante estudiante = new Estudiante(i, nombre.toUpperCase(), puntaje);
            listaDoble.InsertInicio(estudiante);
            System.out.print("=");

            if (i == 75) {
               System.out.println("\n===Datos Cargados con exito===");
            }
        }


        System.out.println("Todos los postulantes:");
        listaDoble.Imprimir();

        System.out.println("Selección de los 50 cupos:");
        listaDoble.SelecTop50MejAlumnosEnCalif();
    }

}
```

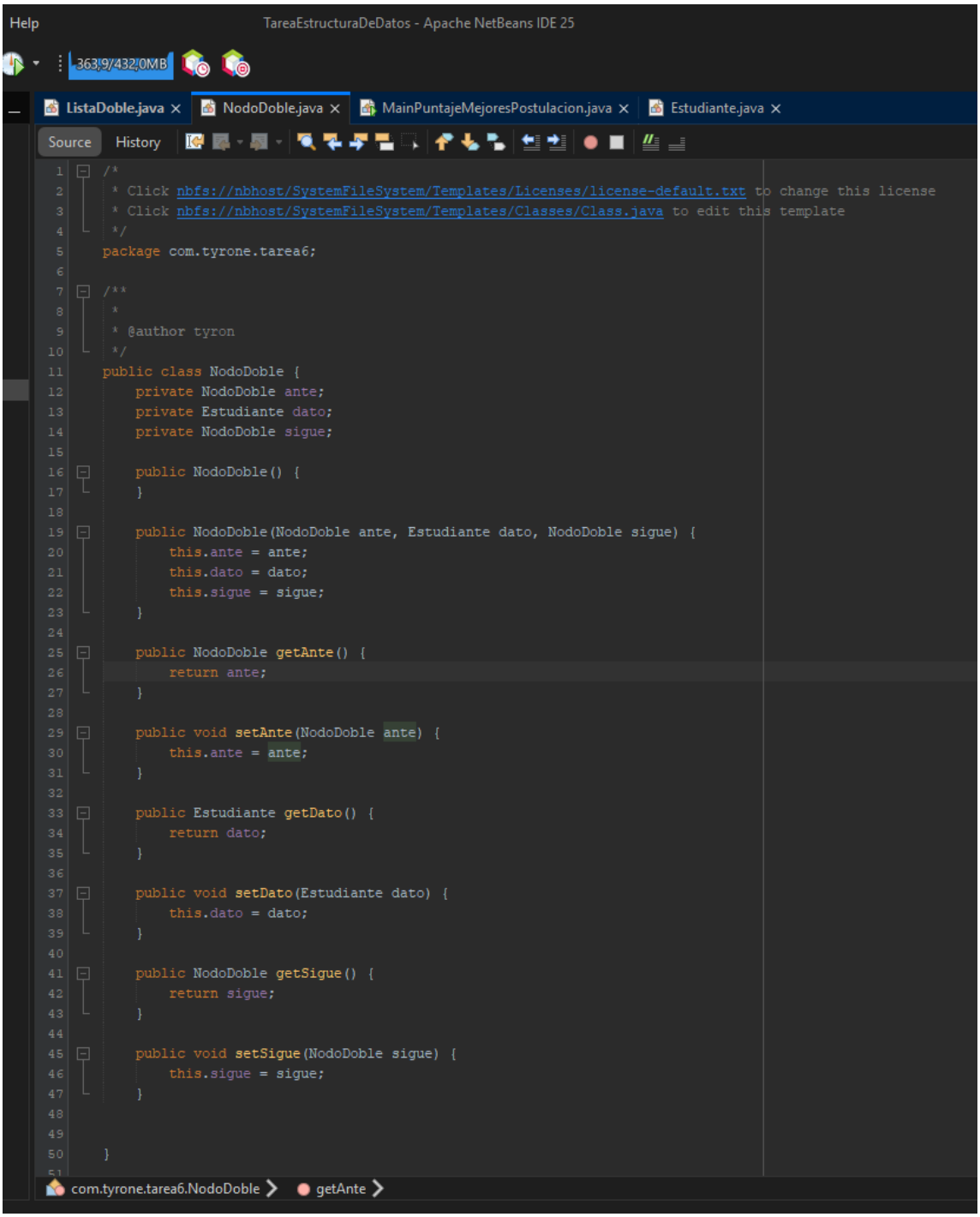
| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

**Captura de las clases**

**Clase Estudiante:**

| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

## Clase NodoDoble



| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

## Clase ListaDoble:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
 */
package com.tyrone.tarea6;

/**
 *
 * @author tyron
 */
public class ListaDoble {
    //atributos
    private NodoDoble inicio;
    private NodoDoble fin;

    //constructor

    public ListaDoble() {
    }

    public ListaDoble(NodoDoble inicio, NodoDoble fin) {
        this.inicio = inicio;
        this.fin = fin;
    }

    //Metodos
    //Lista vacia
    public boolean ListaVacia(){
        return inicio == null;
    }

    //Insertar inicio o final
    public void InsertInicio(Estudiante dato){//metodo que uso el profe en clase
        NodoDoble nuevo = new NodoDoble(null, dato, null);
        if (ListaVacia()) {
            inicio = nuevo;
            fin = nuevo;
        } else {
            nuevo.setDato(dato);
            nuevo.setSigue(inicio);
            inicio.setAnte(nuevo);
            // a mi parecer el codigo mas entendible era nuevo.setAnte(inicio); e
```

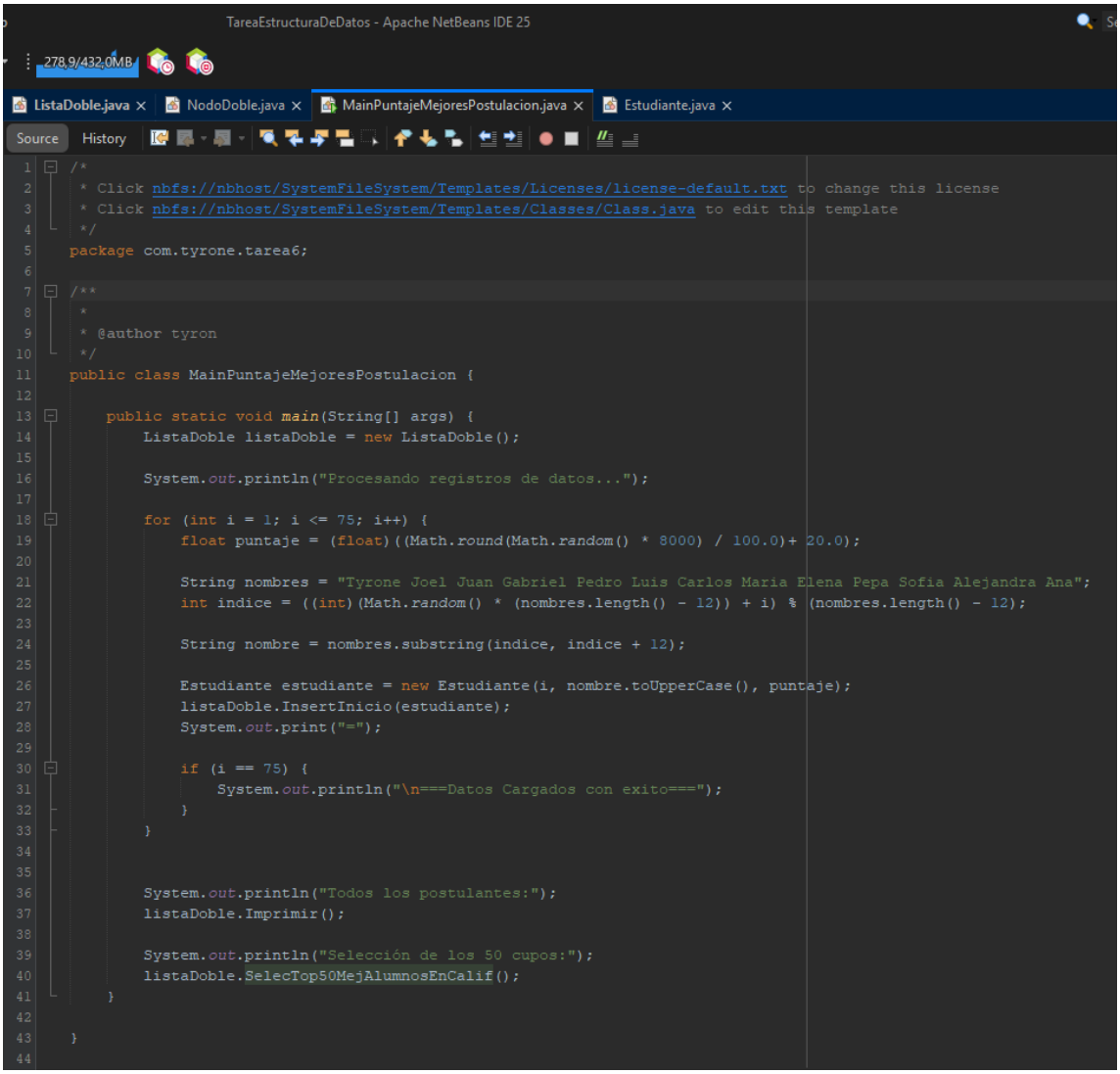
```
            inicio.setAnte(nuevo);
            // a mi parecer el codigo mas entendible era nuevo.setAnte(inicio); e in
            inicio = nuevo;
        }
    }

    public void InsertFinal(Estudiante dato){
        NodoDoble nuevo = new NodoDoble(null, dato, null);
        if (ListaVacia()) {
            inicio = nuevo;
            fin = nuevo;
        } else {//aqui basicamente no ingrese el dato con un setDato porq se esta ha
            fin.setSigue(nuevo);
            nuevo.setAnte(fin);
            fin = nuevo;
        }
    }

    // Eliminar inicio o final
    public String EliminarFinal(){
        if (ListaVacia()) {
            return "Lista vacia";
        } else {
            String n = fin.getDato().getNombre();
            fin = fin.getAnte();
            fin.setSigue(null);
            return n;
        }
    }

    public Estudiante EliminarInicio(){
     if (ListaVacia()) {
         System.out.println("Lista vacia");
         return null;
     } else {
         Estudiante n = inicio.getDato();
         inicio = inicio.getSigue();
         inicio.setAnte(null);
         return n;
     }
    }
```

```
    }
}

//Imprimir sin perder dato
  public void Imprimir(){
      if (ListaVacia()) {
          System.out.println("Lista esta vacia");
      } else {
//            while(inicio != null){
//                System.out.println("el valor  es " + n);
//                Estudiante n = EliminarInicio();

          NodoDoble aux = inicio;
              while(aux != null){
                  System.out.println("Postulante " + aux.getDato());
                  aux = aux.getSigue();

              }

          }
      }

   public ListaDoble ListaPuntajeAscendente() {

       ListaDoble copia = new ListaDoble();

       if (ListaVacia()) {
           System.out.println("Lista vacia");

       }else{

           NodoDoble aux = inicio;

           while (aux != null) { //podia usar el metodo Imprimir pero no queria
               copia.InsertFinal(aux.getDato());
               aux = aux.getSigue();
           }


           NodoDoble nodoComparacion1 = copia.inicio;

           while (nodoComparacion1 != null) {
```

```
            while (nodoComparacion2 != null) {
                if (nodoComparacion2.getDato().getPuntaje() > nodoComparacion1.getDato().getPuntaje()) {
                    Estudiante temp = nodoComparacion1.getDato();
                    nodoComparacion1.setDato(nodoComparacion2.getDato());
                    nodoComparacion2.setDato(temp);
                }
                nodoComparacion2 = nodoComparacion2.getSigue();
            }

            nodoComparacion1 = nodoComparacion1.getSigue();
        }
        return copia;
    }

    public void SelecTop5OMejAlumnosEnCalif(){

        if (ListaVacia()) {
            System.out.println("Lista vacia");

        }else{
            ListaDoble seleccionados = new ListaDoble();
            ListaDoble ordenada = ListaPuntajeAscendente();
            NodoDoble aux = ordenada.inicio;

            int cupos = 50, cuposActuales=0;



            for (int i = 1; i <= cupos; i++) {
                if (aux.getDato().getPuntaje()>=75) {

                    seleccionados.InsertFinal(aux.getDato());
                    cuposActuales = cupos-1 ;
                    System.out.println(i+"- Estudiante "+ aux.getDato());

                } else if (aux.getDato().getPuntaje()<75) {
                    seleccionados.InsertFinal(aux.getDato());
                    cuposActuales = cupos-1 ;
                    System.out.println(i+"- Estudiante "+ aux.getDato());

                } else{
                    System.out.println("sobraron"+ cuposActuales + "  cupos");
                }
                aux = aux.getSigue();

            }

        }

    }
```

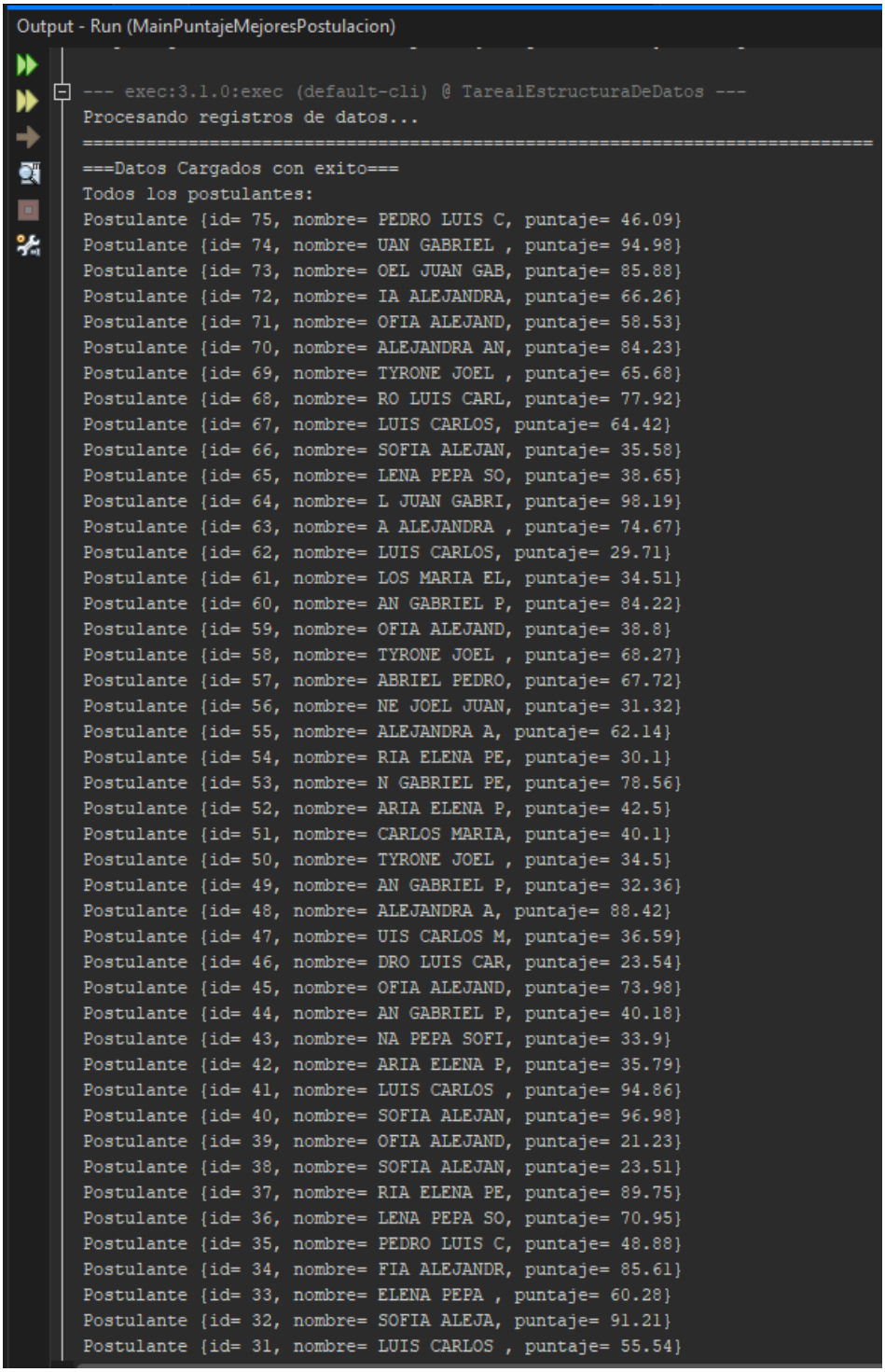
| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

**Clase Main:**

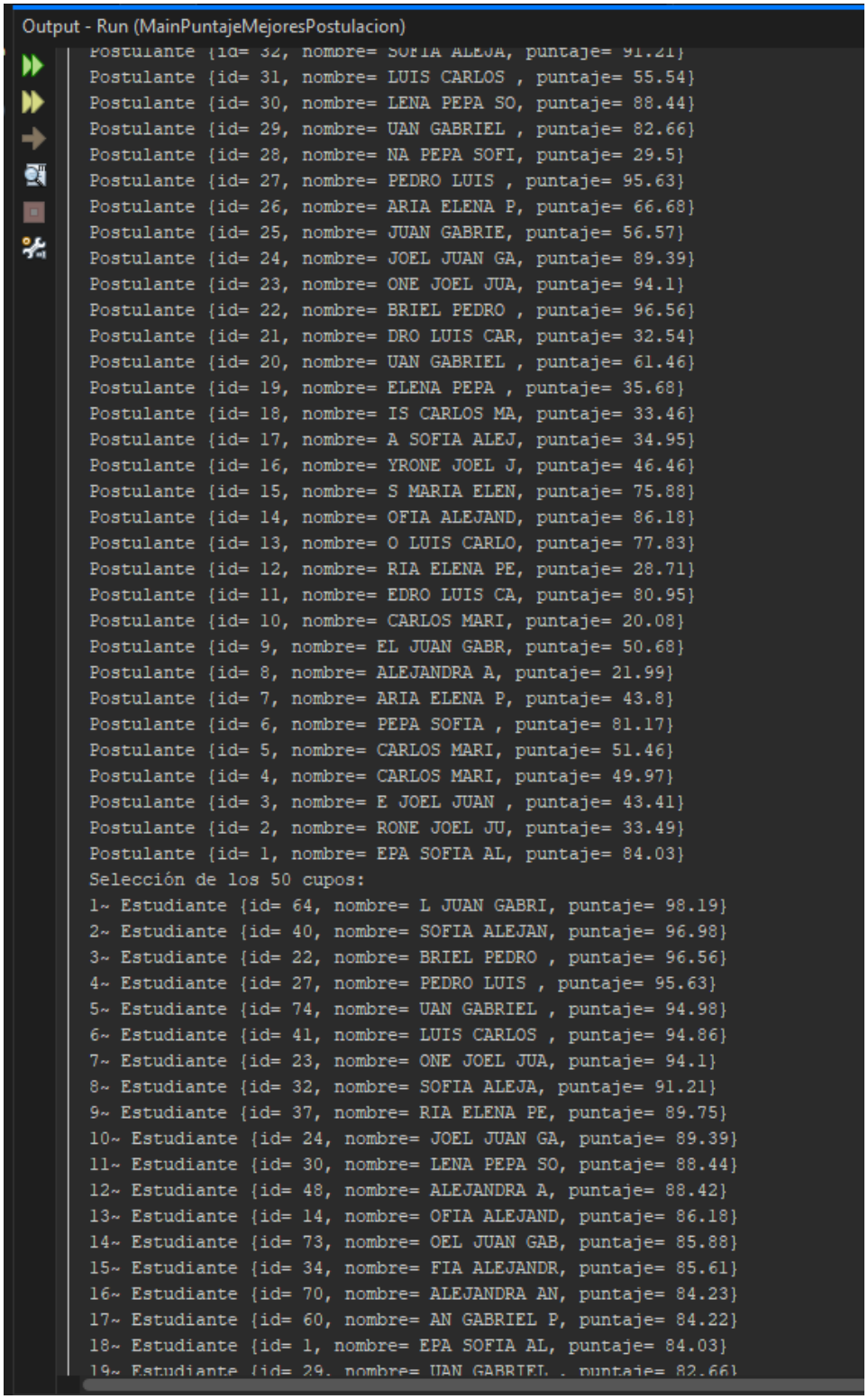


| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |

**Capturas de la ejecución:**



| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |



| Facultad de Ciencias Informáticas \| Carrera de Tecnologías de la Información | | |
|---|---|---|
| **Asignatura** | **Datos del alumno** | **Fecha** |
| **Estructura de Datos** | Apellidos: Pilay *** | 11/12/2025 |
| | Nombre: Tyrone **** | |