



**UTPL**

*La Universidad Católica de Loja*

PROGRAMACIÓN ORIENTADA A OBJETOS

# SISTEMA DE GESTIÓN DE ADMISIONES EN UTPL

## INTEGRANTES:

- EIMER ARMIJOS
- RAUL ANDINO





01  
Introducción

02  
ProblematICA

03  
Puntos  
Claves

04  
Diagrama  
Uml

# INTRODUCCION

Hicimos un proyecto usando programación orientada a objetos, donde se nota lo importante que es usar diagramas UML y tener buenas prácticas al programar, para que el código sea claro y esté bien organizado, siguiendo el modelo MVC.

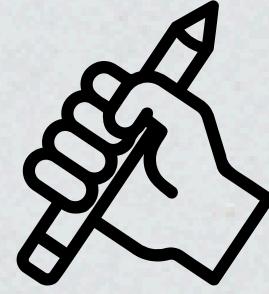


# PROBLEMATICA

La UTPL va a ofrecer 28 carreras y necesita un sistema que ayude a manejar las admisiones. Los estudiantes deben inscribirse, dar un examen y según su puntaje o méritos, se les asignan los cupos. El sistema también debe mostrar reportes como qué carreras tuvieron poca demanda o cuáles rechazaron postulantes por falta de espacio.



# PUNTOS CLAVES A CONSIDERAR



En este proyecto vamos a tener 4 puntos claves.

**01** Diagrama del programa en Dia.uml



**02** Programa realizado en el lenguaje Java.



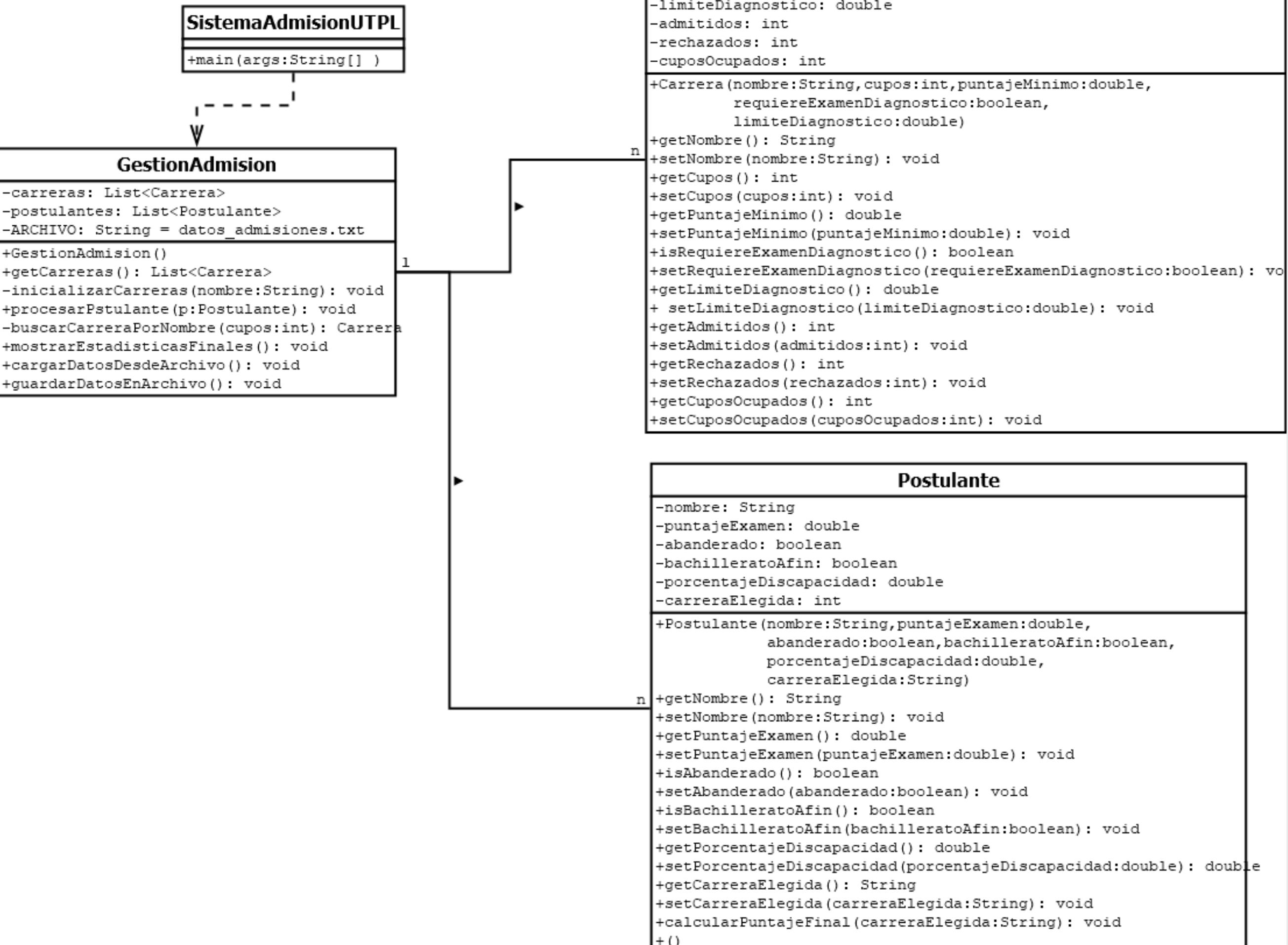
**03** Ejecución del código.

```
state: { products: storeProducts }
render() {
  return (
    <React.Fragment>
      <div className="py-5">
        <div className="container">
          <Title name="our" title="product" />
          <div className="row">
            <ProductConsumer>
              {(value) => {
                console.log(value)
              }}
            </ProductConsumer>
          </div>
        </div>
      </React.Fragment>
    )
}
```

**04** Conclusiones



# DIAGRAMA UML



# EXPLICACION DEL DIAGRAMA

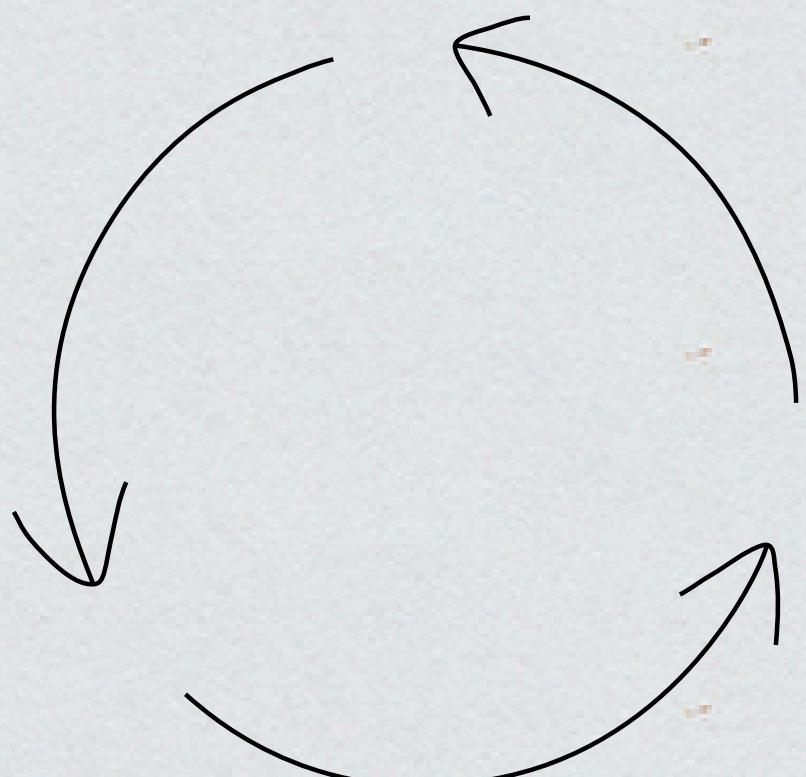
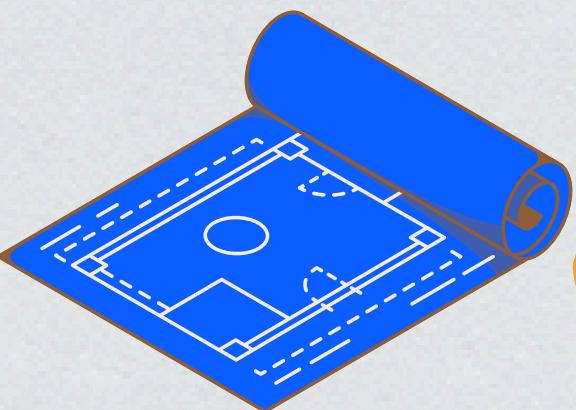


**CONTIENE 4 CLASES**

Clase: GestionAdmision



Clase: Carrera



Clase: Postulante



Clase: SistemaAdmisionUTPL

**SIGN UP NOW**

# CLASE GESTION ADMISSION

## GestionAdmision

```
-carreras: List<Carrera>
-postulantes: List<Postulante>
-ARCHIVO: String = datos_admisiones.txt

+GestionAdmision()
+getCarreras(): List<Carrera>
-inicializarCarreras(nombre:String): void
+procesarPostulante(p:Postulante): void
-buscarCarreraPorNombre(cupos:int): Carrera
+mostrarEstadisticasFinales(): void
+cargarDatosDesdeArchivo(): void
+guardarDatosEnArchivo(): void
```



# CLASE CARRERA

## Carrera

```
-nombre: String  
-cupos: int  
-puntajeMinimo: double  
-requiereExamenDiagnostico: boolean  
-limiteDiagnostico: double  
-admitidos: int  
-rechazados: int  
-cuposOcupados: int  
  
+Carrera(nombre:String, cupos:int, puntajeMinimo:double,  
          requiereExamenDiagnostico:boolean,  
          limiteDiagnostico:double)  
+getNombre(): String  
+setNombre(nombre:String): void  
+getCupos(): int  
+setCupos(cupos:int): void  
+getPuntajeMinimo(): double  
+setPuntajeMinimo(puntajeMinimo:double): void  
+isRequiereExamenDiagnostico(): boolean  
+setRequiereExamenDiagnostico(requiereExamenDiagnostico:boolean): void  
+getLmiteDiagnostico(): double  
+ setLmiteDiagnostico(lmiteDiagnostico:double): void  
+getAdmitidos(): int  
+setAdmitidos(admitidos:int): void  
+getRechazados(): int  
+setRechazados(rechazados:int): void  
+getCuposOcupados(): int  
+setCuposOcupados(cuposOcupados:int): void
```



# CLASE POSTULANTE

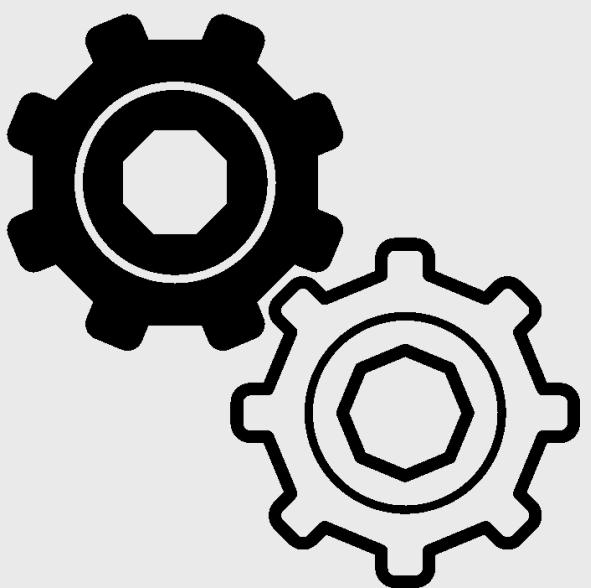
## Postulante

```
-nombre: String  
-puntajeExamen: double  
-abanderado: boolean  
-bachilleratoAfin: boolean  
-porcentajeDiscapacidad: double  
-carreraElegida: int  
  
+Postulante(nombre:String, puntajeExamen:double,  
            abanderado:boolean, bachilleratoAfin:boolean,  
            porcentajeDiscapacidad:double,  
            carreraElegida:String)  
+getNombre(): String  
+setNombre(nombre:String): void  
+getPuntajeExamen(): double  
+setPuntajeExamen(puntajeExamen:double): void  
+isAbanderado(): boolean  
+setAbanderado(abanderado:boolean): void  
+isBachilleratoAfin(): boolean  
+setBachilleratoAfin(bachilleratoAfin:boolean): void  
+getPorcentajeDiscapacidad(): double  
+setPorcentajeDiscapacidad(porcentajeDiscapacidad:double): double  
+getCarreraElegida(): String  
+setCarreraElegida(carreraElegida:String): void  
+calcularPuntajeFinal(carreraElegida:String): void  
+()
```



# CLASE MAIN O TAMBIEN LLAMADA SISTEMA ADMISSIONES UTPL

```
SistemaAdmisionUTPL
+
+main (args : String[] )
```





# CLASES EN JAVA

- 1** `public class Postulante`
- 2** `public class Carrera`
- 3** `public class GestionAdmision`
- 4** `public class SistemaAdmisionUTPL`

# SISTEMA DE ADMISIÓN DE UTPL

**Gestionar el proceso completo de admisión de postulantes a la UTPL, incluyendo:**

- Verificar que la fecha ingresada esté dentro del periodo de admisión permitido (del 23-05-2025 al 23-06-2025).
- Pedir al usuario los datos del postulante: nombre, puntaje, si fue abanderado, si su bachillerato es afín, discapacidad y carrera.
- Crear un objeto postulante con esos datos.
- Calcular su puntaje final y procesarlo en el sistema de admisión.
- Repetir el proceso para más postulantes si el usuario desea.
- Mostrar estadísticas finales y guardar los datos en un archivo.

```
package proyecto.poo;
import java.util.*;
import java.text.*;

public class SistemaAdmisionUTPL {
    Run | Debug
    public static void main(String[] args) {
        GestionAdmision gestion = new GestionAdmision();
        Scanner tcl = new Scanner(System.in);
        String continuar;

        System.out.println(x:"Bienvenido al Sistema de Admisión UTPL");
        System.out.print(s:"Ingrese la fecha actual 23-05-2025 hasta 23-06-2025: ");
        String fechastr = tcl.nextLine();

        SimpleDateFormat formato = new SimpleDateFormat(pattern:"dd-MM-yyyy");
        formato.setLenient(lenient:false);
        Date fechaIngreso;
        Date fechaInicio, fechaFin;

        try {
            fechaIngreso = formato.parse(fechastr);
            fechaInicio = formato.parse(source:"23-05-2025");
            fechaFin = formato.parse(source:"23-06-2025");

            if (fechaIngreso.before(fechaInicio) || fechaIngreso.after(fechaFin)) {
                System.out.println(x:" No es posible ingresar postulantes fuera del periodo de admisión (23-05-2025 a 23-06-2025).");
                tcl.close();
                return;
            }
        } catch (ParseException e) {
            System.out.println(x:" Fecha inválida. Debe ingresar en formato dd-MM-yyyy.");
            tcl.close();
            return;
        }
    }
}
```



\*Fragmento del programa de la clase antes mencionada.

# GESTION ADMISSION

Como su nombre mismo lo dice, esta clase se encarga de gestionar todo el proceso de admisión de postulantes a la UTPL.

Se encarga de cargar los datos desde un archivo :

= datos\_admision.txt

Si existe, recupera carreras y postulantes anteriores.

Si no existe, crea una lista de carreras predefinidas.

Registra y procesa postulantes nuevos:

- Verifica si cumplen los requisitos de puntaje.
- Aplica reglas especiales (como si requiere examen diagnóstico).
- Admite o rechaza al postulante según los cupos disponibles.

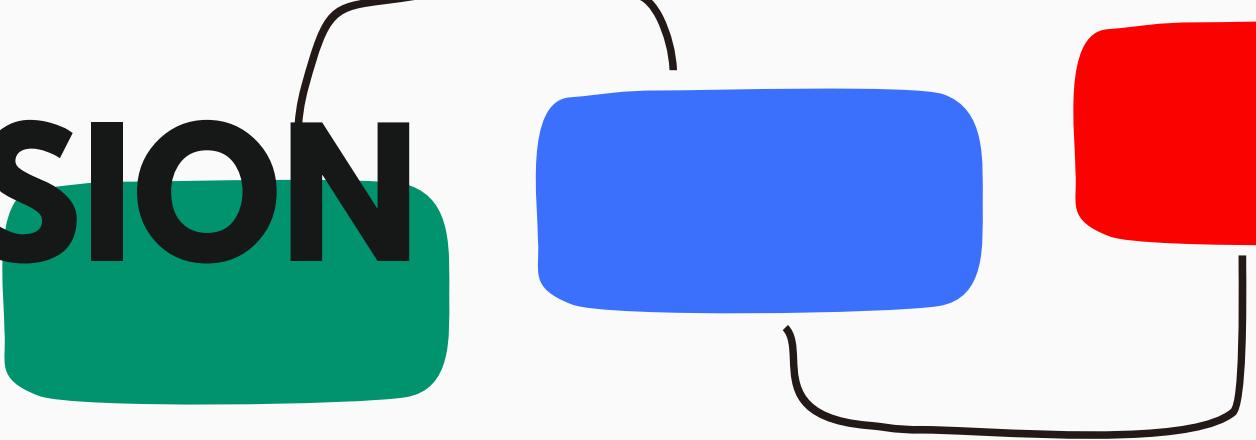
```
package proyecto.poo;
import java.util.*;
import java.io.*;
public class GestionAdmision {
    private List<Carrera> carreras;
    private List<Postulante> postulantes;
    private static final String ARCHIVO = "datos_admision.txt";

    public GestionAdmision() {
        carreras = new ArrayList<>();
        postulantes = new ArrayList<>();
        cargarDatosDesdeArchivo();
        if (carreras.isEmpty()) {
            inicializarCarreras();
        }
    }
    public List<Carrera> getCarreras() {
        return carreras;
    }
    private void inicializarCarreras() {
        carreras.add(new Carrera(nombre:"Administración de Empresas", cupos:50, puntajeMinimo:65, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Agropecuaria", cupos:40, puntajeMinimo:60, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Alimentos", cupos:40, puntajeMinimo:60, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Artes Escénicas", cupos:30, puntajeMinimo:50, requiereExamenDi...true, 30));
        carreras.add(new Carrera(nombre:"Artes Visuales", cupos:30, puntajeMinimo:50, requiereExamenDi...true, 30));
        carreras.add(new Carrera(nombre:"Biología", cupos:35, puntajeMinimo:60, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Contabilidad y Auditoría", cupos:50, puntajeMinimo:65, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Derecho", cupos:40, puntajeMinimo:65, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Economía", cupos:40, puntajeMinimo:65, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Finanzas", cupos:40, puntajeMinimo:65, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Geología", cupos:35, puntajeMinimo:60, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Ingeniería Ambiental", cupos:40, puntajeMinimo:65, requiereExamenDi...false, 0));
        carreras.add(new Carrera(nombre:"Pedagogía de los Idiomas Nacionales y Extranjeros", cupos:30, puntajeMinimo:50, requiereExame...
```



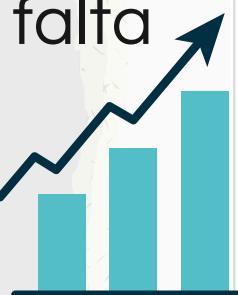
\*Fragmento del programa de la clase antes mencionada.

# GESTION ADMISSION



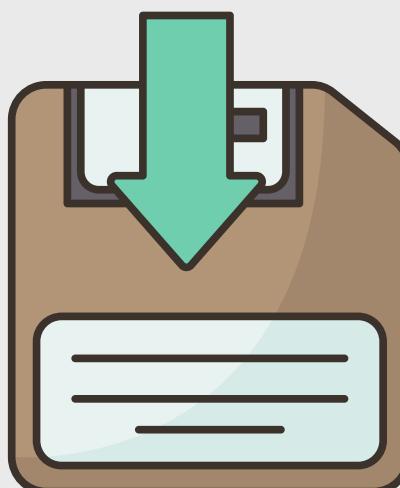
Controla los cupos y estadísticas:

- Aumenta el número de admitidos/rechazados según corresponda.
- Al final, muestra un resumen con:
  - Carreras con menos del 50% de cupos ocupados.
  - Carreras que rechazaron postulantes por falta de cupos.



```
public List<Carrera> getCarreras() {  
    return carreras;  
}  
  
private void inicializarCarreras() {  
    carreras.add(new Carrera(nombre: "Administración de Empresas", cupos:50, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Agropecuaria", cupos:40, puntajeMinimo:60, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Alimentos", cupos:40, puntajeMinimo:60, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Artes Escénicas", cupos:30, puntajeMinimo:50, requiereExamenDirigido:true, 30));  
    carreras.add(new Carrera(nombre: "Artes Visuales", cupos:30, puntajeMinimo:50, requiereExamenDirigido:true, 30));  
    carreras.add(new Carrera(nombre: "Biología", cupos:35, puntajeMinimo:60, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Contabilidad y Auditoría", cupos:50, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Derecho", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Economía", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Finanzas", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Geología", cupos:35, puntajeMinimo:60, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Ingeniería Ambiental", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Pedagogía de los Idiomas Nacionales y Extranjeros", cupos:30, puntajeMinimo:50, requiereExamenDirigido:true, 30));  
    carreras.add(new Carrera(nombre: "Psicopedagogía", cupos:30, puntajeMinimo:50, requiereExamenDirigido:true, 30));  
    carreras.add(new Carrera(nombre: "Telecomunicaciones", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Arquitectura", cupos:30, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Bioquímica y Farmacia", cupos:30, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Computación", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Enfermería", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Fisioterapia", cupos:30, puntajeMinimo:60, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Gastronomía", cupos:35, puntajeMinimo:50, requiereExamenDirigido:true, 30));  
    carreras.add(new Carrera(nombre: "Ingeniería Civil", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Ingeniería Industrial", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Ingeniería Química", cupos:35, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Medicina", cupos:30, puntajeMinimo:75, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Nutrición y Dietética", cupos:35, puntajeMinimo:60, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Psicología", cupos:40, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
    carreras.add(new Carrera(nombre: "Psicología Clínica", cupos:30, puntajeMinimo:65, requiereExamenDirigido:false, 0));  
}
```

\*Fragmento del programa de la clase antes mencionada.



# CARRERA

**Esta clase representa una carrera universitaria disponible en el sistema de admisión.**

Define los datos de una carrera:

- Nombre (ej. Medicina, Derecho, etc.).
- Cupos totales disponibles.
- Puntaje mínimo requerido para ingresar.
- Si requiere o no un examen diagnóstico.
- Límite mínimo para aprobar ese examen (si aplica).



```
package proyecto.poo;
public class Carrera {
    private String nombre;
    private int cupos;
    private double puntajeMinimo;
    private boolean requiereExamenDiagnostico;
    private double limiteDiagnstico;
    private int admitidos;
    private int rechazados;
    private int cuposOcupados;

    public Carrera(String nombre, int cupos, double puntajeMinimo, boolean requiereExamenDiagnostico, double limiteDiagnstico) {
        this.nombre = nombre;
        this.cupos = cupos;
        this.puntajeMinimo = puntajeMinimo;
        this.requiereExamenDiagnostico = requiereExamenDiagnostico;
        this.limiteDiagnstico = limiteDiagnstico;
        this.admitidos = 0;
        this.rechazados = 0;
        this.cuposOcupados = 0;
    }

    public String getNombre() {
        return nombre;
    }
}
```

\*Fragmento del programa de la clase antes mencionada.

# CARRERA

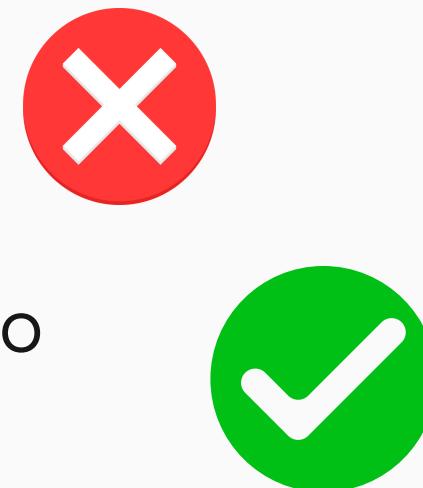
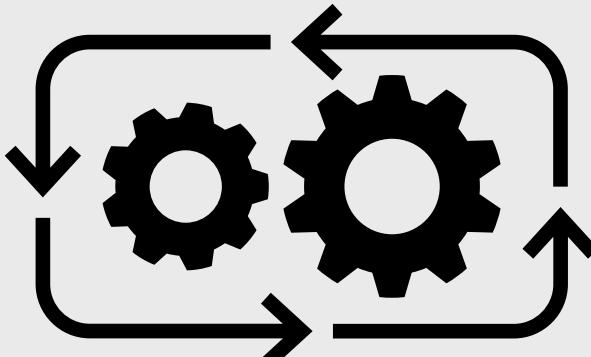
Lleva el control del proceso de admisión:

También gestiona las estadísticas de admisión de la carrera:

- admitidos: cuántos postulantes han sido aceptados.
- rechazados: cuántos postulantes fueron rechazados.
- cuposOcupados: cuántos cupos ya han sido asignados.

Proporciona métodos para leer y modificar estos datos

- `getNombre()` devuelve el nombre de la carrera.
- `setCupos(int c)` permite cambiar el número de cupos.



```
public String getNombre() {  
    return nombre;  
}  
  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}  
  
public int getCupos() {  
    return cupos;  
}  
  
public void setCupos(int cupos) {  
    this.cupos = cupos;  
}  
  
public double getPuntajeMinimo() {  
    return puntajeMinimo;  
}  
  
public void setPuntajeMinimo(double puntajeMinimo) {  
    this.puntajeMinimo = puntajeMinimo;  
}  
  
public boolean isRequiereExamenDiagnostico() {  
    return requiereExamenDiagnostico;  
}  
  
public void setRequiereExamenDiagnostico(boolean requiereExamenDiagnostico) {  
    this.requiereExamenDiagnostico = requiereExamenDiagnostico;  
}
```

\*Fragmento del programa de la clase antes mencionada.

# POSTULANTE

**Esta clase representa a un estudiante que desea ingresar a una carrera universitaria.**

Guarda la información personal y académica del postulante:

La clase almacena los siguientes datos:

- nombre: nombre del estudiante.
- puntajeExamen: resultado obtenido en el examen de admisión.
- abanderado: si fue abanderado (true/false).
- bachilleratoAfin: si su bachillerato es afín a la carrera elegida.
- porcentajeDiscapacidad: grado de discapacidad (si tiene).
- carreraElegida: nombre de la carrera a la que se está postulando.



```
package proyecto.poo;
public class Postulante {
    private String nombre;
    private double puntajeExamen;
    private boolean abanderado;
    private boolean bachilleratoAfin;
    private double porcentajeDiscapacidad;
    private String carreraElegida;

    public Postulante(String nombre, double puntajeExamen, boolean abanderado, boolean bachilleratoAfin,
                      this.nombre = nombre;
                      this.puntajeExamen = puntajeExamen;
                      this.abanderado = abanderado;
                      this.bachilleratoAfin = bachilleratoAfin;
                      this.porcentajeDiscapacidad = porcentajeDiscapacidad;
                      this.carreraElegida = carreraElegida;
    }

    public String getNombre() {
        return nombre;
    }

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

    public double getPuntajeExamen() {
        return puntajeExamen;
    }
}
```

\*Fragmento del programa de la clase antes mencionada.

# POSTULANTE

Permite acceder o modificar estos datos:



- Mediante getters y setters, se pueden consultar o actualizar los atributos del postulante de forma segura, protegiendo el acceso directo a los datos.



Calcula el puntaje final del postulante:

El método calcularPuntajeFinal() ajusta el puntaje del

examen según ciertos beneficios:



- Si es abanderado, suma +5 puntos.
- Si tiene un bachillerato afín, suma +2 puntos.
- Si tiene más del 35% de discapacidad, suma +3 puntos.
- El puntaje total nunca excede 100 puntos.

**SCORE!**

```
public void setPorcentajeDiscapacidad(double porcentajeDiscapacidad) {  
    this.porcentajeDiscapacidad = porcentajeDiscapacidad;  
}  
  
public String getCarreraElegida() {  
    return carreraElegida;  
}  
  
public void setCarreraElegida(String carreraElegida) {  
    this.carreraElegida = carreraElegida;  
}  
  
public double calcularPuntajeFinal() {  
    double puntajeFinal = puntajeExamen;  
  
    if (abanderado) {  
        puntajeFinal += 5;  
    }  
    if (bachilleratoAfin) {  
        puntajeFinal += 2;  
    }  
    if (porcentajeDiscapacidad > 35) {  
        puntajeFinal += 3;  
    }  
  
    return puntajeFinal > 100 ? 100 : puntajeFinal;  
}
```

\*Fragmento del programa de la clase antes mencionada.

# EJECUCIÓN DEL CÓDIGO



```
Bienvenido al Sistema de Admisión UTPL
```

```
Ingrese la fecha actual (formato dd-MM-yyyy, entre 23-05-2025 y 23-06-2025): 3-6-2025
```

```
Ingrese los datos del postulante:
```

```
Nombre: Bryan
```

```
Puntaje obtenido (sobre 100): 65
```

```
¿Fue abanderado? (si/no): no
```

```
¿Su bachillerato es afín a la carrera? (si/no): si
```

```
Porcentaje de discapacidad (0 si no aplica): 25
```

```
Seleccione una carrera por su número:
```

1. Administración de Empresas
2. Agropecuaria
3. Alimentos
4. Artes Escénicas
5. Artes Visuales
6. Biología
7. Contabilidad y Auditoría
8. Derecho
9. Economía
10. Finanzas
11. Geología
12. Ingeniería Ambiental
13. Pedagogía de los Idiomas Nacionales y Extranjeros
14. Psicopedagogía
15. Telecomunicaciones
16. Arquitectura
17. Bioquímica y Farmacia
18. Computación
19. Enfermería
20. Fisioterapia
21. Gastronomía

# EJECUCIÓN DEL CÓDIGO

```
22. Ingeniería Civil  
23. Ingeniería Industrial  
24. Ingeniería Química  
25. Medicina  
26. Nutrición y Dietética  
27. Psicología  
28. Psicología Clínica  
Opción (1 - 28): 18  
  
Calculando puntaje final...  
Puntaje final del postulante: 67.0  
Admitido en Computación  
  
¿Desea ingresar otro postulante? (s/n): s  
  
Ingrese los datos del postulante:  
Nombre: Eimer  
Puntaje obtenido (sobre 100): 85  
¿Fue abanderado? (si/no): no  
¿Su bachillerato es afín a la carrera? (si/no): no  
Porcentaje de discapacidad (0 si no aplica): 0  
  
Seleccione una carrera por su número:  
1. Administración de Empresas  
2. Agropecuaria  
3. Alimentos  
4. Artes Escénicas  
5. Artes Visuales  
6. Biología  
7. Contabilidad y Auditoría  
8. Derecho  
9. Economía  
10. Finanzas
```



# EJECUCIÓN DEL CÓDIGO

```
11. Geología  
12. Ingeniería Ambiental  
13. Pedagogía de los Idiomas Nacionales y Extranjeros  
14. Psicopedagogía  
15. Telecomunicaciones  
16. Arquitectura  
17. Bioquímica y Farmacia  
18. Computación  
19. Enfermería  
20. Fisioterapia  
21. Gastronomía  
22. Ingeniería Civil  
23. Ingeniería Industrial  
24. Ingeniería Química  
25. Medicina  
26. Nutrición y Dietética  
27. Psicología  
28. Psicología Clínica  
Opción (1 - 28): 19
```

Calculando puntaje final...

Puntaje final del postulante: 85.0

Admitido en Enfermería

¿Desea ingresar otro postulante? (s/n): s

Ingrese los datos del postulante:

Nombre: Raul

Puntaje obtenido (sobre 100): 45

¿Fue abanderado? (si/no): no

¿Su bachillerato es afín a la carrera? (si/no): si

Porcentaje de discapacidad (0 si no aplica): 25



# EJECUCIÓN DEL CÓDIGO

Seleccione una carrera por su número:

- 1. Administración de Empresas
- 2. Agropecuaria
- 3. Alimentos
- 4. Artes Escénicas
- 5. Artes Visuales
- 6. Biología
- 7. Contabilidad y Auditoría
- 8. Derecho
- 9. Economía
- 10. Finanzas
- 11. Geología
- 12. Ingeniería Ambiental
- 13. Pedagogía de los Idiomas Nacionales y Extranjeros
- 14. Psicopedagogía
- 15. Telecomunicaciones
- 16. Arquitectura
- 17. Bioquímica y Farmacia
- 18. Computación
- 19. Enfermería
- 20. Fisioterapia
- 21. Gastronomía
- 22. Ingeniería Civil
- 23. Ingeniería Industrial
- 24. Ingeniería Química
- 25. Medicina
- 26. Nutrición y Dietética
- 27. Psicología
- 28. Psicología Clínica

Opción (1 - 28): 18

Calculando puntaje final...

Puntaje final del postulante: 47.0



# EJECUCIÓN DEL CÓDIGO

Puntaje final del postulante: 47.0

No alcanza el puntaje mínimo para Computación

¿Desea ingresar otro postulante? (s/n) : n

--- Estadísticas Finales ---

Carreras con menos del 50% de cupos ocupados:

Administración de Empresas - 0.00% de cupos ocupados

Agropecuaria - 0.00% de cupos ocupados

Alimentos - 0.00% de cupos ocupados

Artes Escénicas - 0.00% de cupos ocupados

Artes Visuales - 0.00% de cupos ocupados

Biología - 0.00% de cupos ocupados

Contabilidad y Auditoría - 0.00% de cupos ocupados

Derecho - 0.00% de cupos ocupados

Economía - 0.00% de cupos ocupados

Finanzas - 0.00% de cupos ocupados

Geología - 0.00% de cupos ocupados

Ingeniería Ambiental - 0.00% de cupos ocupados

Pedagogía de los Idiomas Nacionales y Extranjeros - 0.00% de cupos ocupados

Psicopedagogía - 0.00% de cupos ocupados

Telecomunicaciones - 0.00% de cupos ocupados

Arquitectura - 0.00% de cupos ocupados

Bioquímica y Farmacia - 0.00% de cupos ocupados

Computación - 5.00% de cupos ocupados

Enfermería - 2.50% de cupos ocupados

Fisioterapia - 0.00% de cupos ocupados

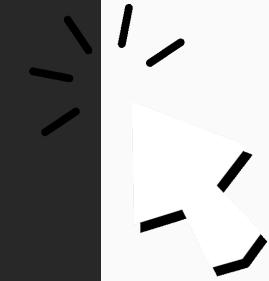
Gastronomía - 0.00% de cupos ocupados

Ingeniería Civil - 0.00% de cupos ocupados

Ingeniería Industrial - 0.00% de cupos ocupados

Ingeniería Química - 0.00% de cupos ocupados

Medicina - 0.00% de cupos ocupados



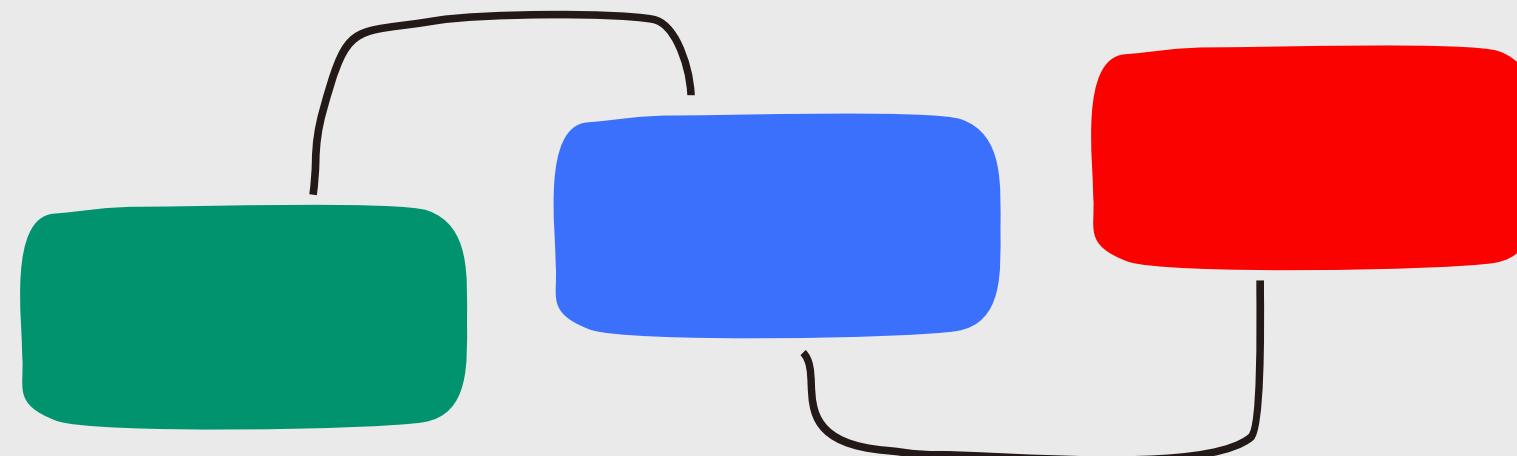
# EJECUCIÓN DEL CÓDIGO

Medicina - 0.00% de cupos ocupados  
Nutrición y Dietética - 0.00% de cupos ocupados  
Psicología - 0.00% de cupos ocupados  
Psicología Clínica - 0.00% de cupos ocupados

Carreras que tuvieron que rechazar postulantes por falta de cupos:  
Computación - 1 postulantes rechazados

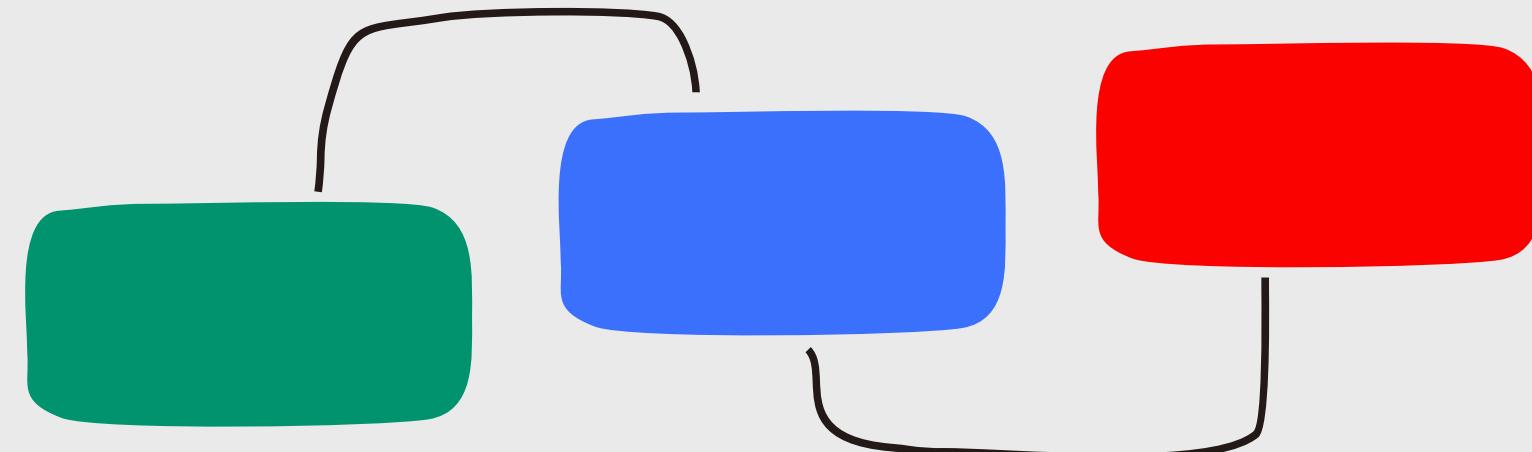
# RELACIÓN ENTRE CLASES DEL SISTEMA DE ADMISIÓN

- GESTIONADMISION USA A CARRERA Y POSTULANTE PARA PROCESAR ADMISIONES.
- CARRERA REPRESENTA LAS CARRERAS DISPONIBLES (CUPOS Y REQUISITOS).
- POSTULANTE REPRESENTA AL ESTUDIANTE Y SU PUNTAJE FINAL.
- POSTULANTE SE CONECTA SOLO CUANDO GESTIONADMISION LO PROCESA.



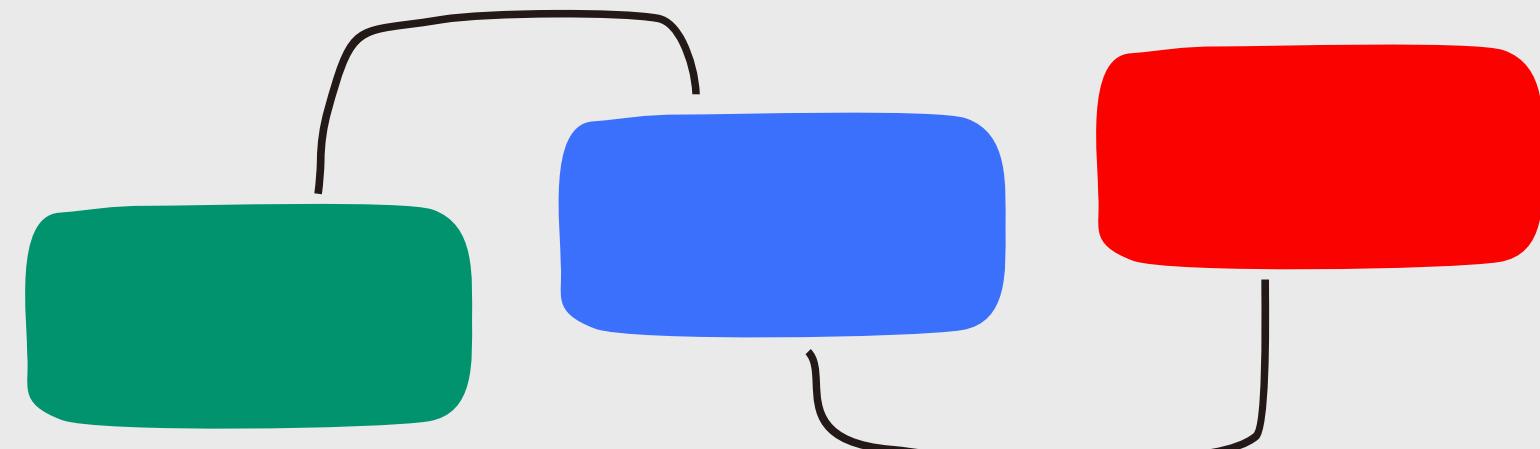
# DIFICULTADES

- EL DIAGRAMA UML SE NOS COMPLICÓ UN POCO, PERO NADA DIFÍCIL
- AL MOMENTO DE LEER Y ESCRIBIR LOS ARCHIVOS PERO NADA RELEVANTE



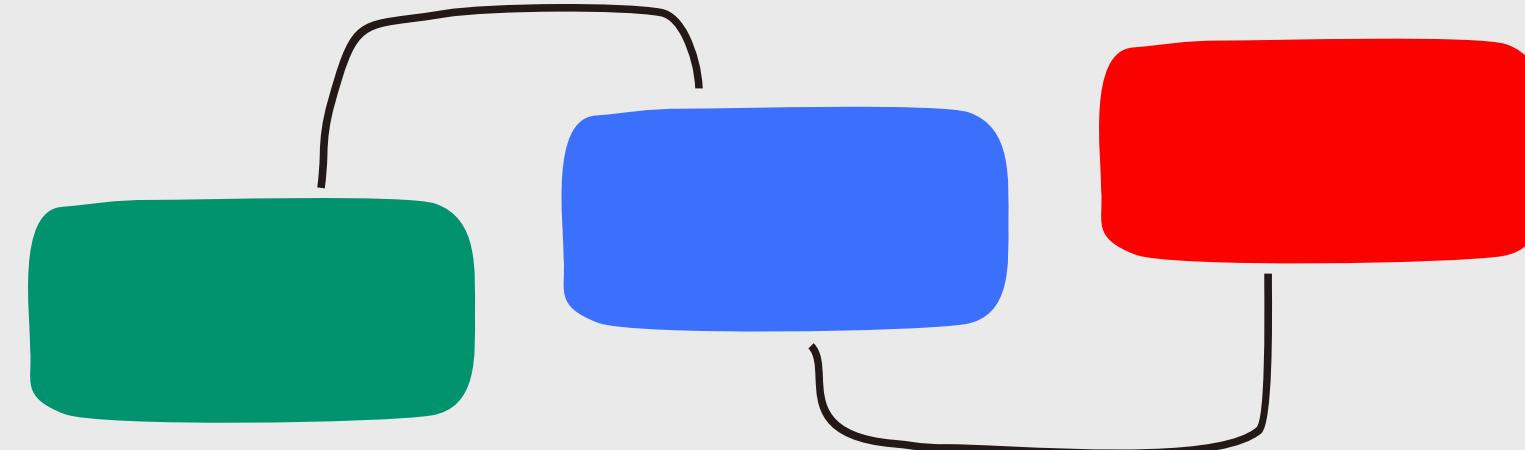
# APRENDIZAJES

- EL MANEJO DE LISTAS DE UNA MEJOR FORMA.
- MANEJO MEJOR DEL PROGRAMA DIA
- COMPAÑERISMO
- TRABAJO EN GRUPO



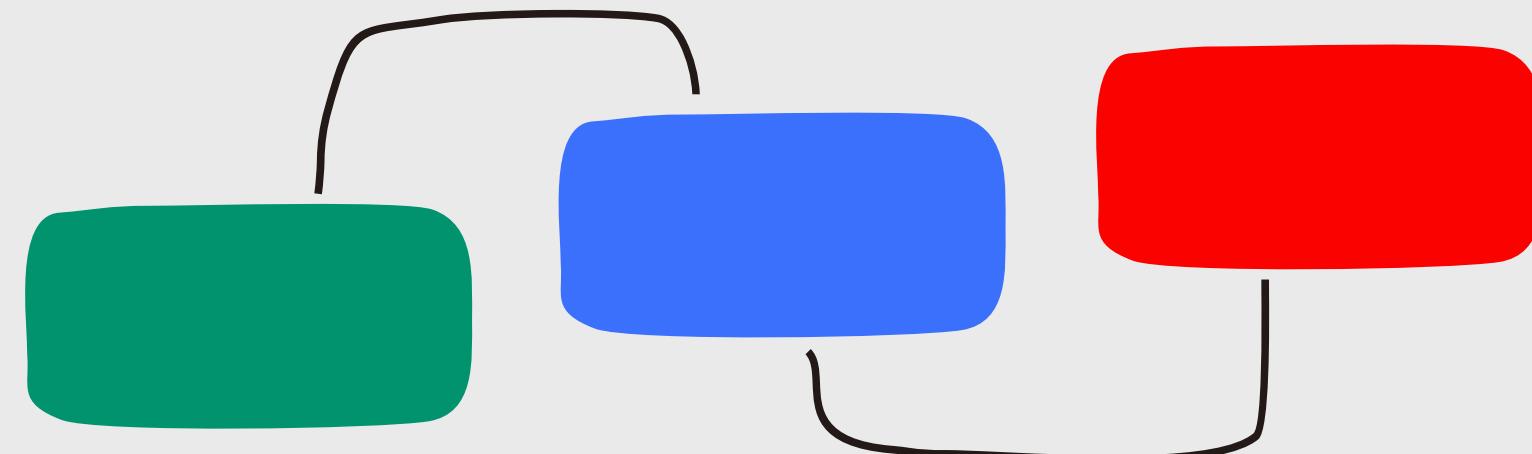
# EXPERIENCIAS

- AL REALIZAR EN GRUPO EL PROYECTO ENTENDI Y COMPRENDI MEJOR LA ESTRUCTURA DE LOS PROGRAMAS Y LAS BUENAS PRACTICAS DE PROGRAMACIÓN.



# SUGERENCIAS

- INCLUIR UNA INTERFAZ GRAFICA EN UN FUTURO.
- MEJORAR LA BASE DE DATOS.



**MUCHAS GRACIAS  
POR SU ATENCIÓN**

