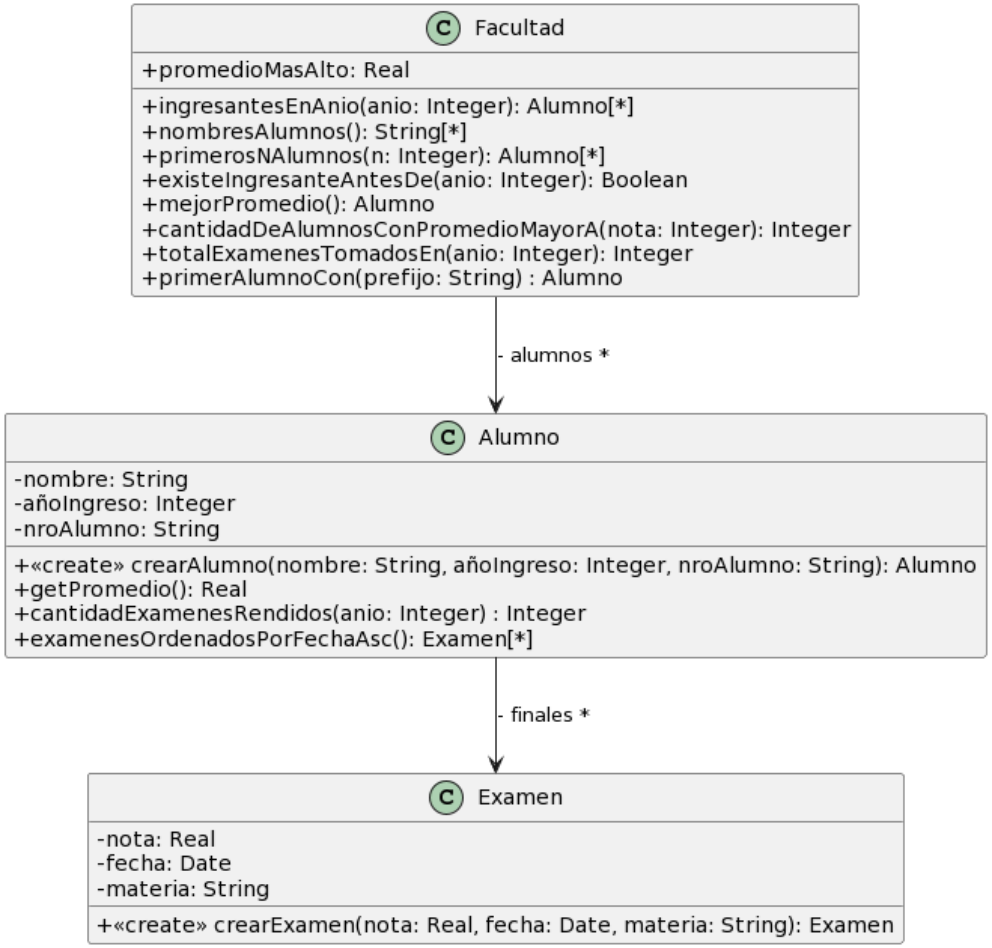

STREAMS

MATERIAL DE APOYO

API STREAM

FILTER

Filtra los elementos de un stream según el predicado que recibe como parámetro. Ej: obtener los alumnos que ingresaron en un año dado

```
public List<Alumno> ingresantesEnAnio(int anio){
  return alumnos.stream()
  .filter(alumno->alumno.getAnioIngreso() == anio)
  .collect(Collectors.toList());
```

MAP

Genera un stream, de igual longitud que el original, a partir de aplicar la función que recibe como parámetro sobre cada elemento del stream original. Ej: obtener una lista de los nombres de todos los alumnos.

```
public List<String> nombresAlumnos(){
   return alumnos.stream()
          .map(alumno -> alumno.getNombre())
          .collect(Collectors.toList());
}
```

Op. relacionadas: MAPTODOUBLE | MAPTOINT

Genera un subtipo de stream (DoubleStream, IntStream...), de igual longitud que el original, a partir de aplicar la función que recibe como parámetro sobre cada elemento del stream original. Se utiliza cuando se trabaja con tipos primitivos como double e int respectivamente.

LIMIT

Trunca el stream dejando los primeros N elementos. Ej: obtener una lista de los primeros N alumnos.

```
public List<Alumno> primerosNAlumnos(int n){
   return alumnos
       .stream().limit(n)
       .collect(Collectors.toList()); }
```

ANYMATCH

Evalúa si existe al menos un elemento del stream que satisface el predicado que se recibe como parámetro, y en ese caso retorna verdadero. Caso contrario, retorna falso. Ej: consultar si algún alumno ingresó antes de un año dado o no.

```
public boolean existeIngresanteAntesDe(int anio){
  return alumnos.stream()
 .anyMatch(alumno->alumno.getAnioIngreso()<anio);
}
```

Op. relacionadas: ALLMATCH | NONEMATCH

Evalúa si todos los elementos (o ninguno de los elementos) del stream satisfacen el predicado que se recibe como parámetro, y en ese caso retorna verdadero. Caso contrario, retorna falso.

MAX | MIN

Retorna el elemento máximo del stream de acuerdo a la expresión indicada como parámetro.
Ej: obtener el alumno con el mayor promedio

```
public Alumno mejorPromedio(){
  return alumnos.stream()
  .max((a1, a2)-> Double.compare(
       a1.getPromedio(), a2.getPromedio()))
  .orElse(null);
}
```

Si trabajamos con un stream de números (DoubleStream, IntStream...) no requiere parámetros.
Ej: se quiere obtener el promedio más alto.

```
public double promedioMasAlto(){
  return  alumnos.stream()
         .mapToDouble(alumno->alumno.getPromedio())
         .max().orElse(0);
}
```

COUNT

Retorna la cantidad de elementos en el stream. Ej: obtener la cantidad de alumnos con promedio mayor a una nota determinada

```
public int cantidadDeAlumnosConPromedioMayorA(int nota){
  return (int) alumnos.stream()
  .filter(alumno-> alumno.getPromedio() >= nota)
  .count();
```

SUM

Retorna la suma de los elementos de un stream de números (DoubleStream, IntStream...). Ej: calcular cuántos exámenes se tomaron en total a todos los alumnos, en un año determinado.

```
public int totalExamenesTomadosEn(int anio){
 return alumnos.stream()
 .mapToInt(alumno->alumno.cantidadExamenesRendidos(anio))
 .sum();
}
```

AVERAGE

Retorna el promedio de los elementos de un stream de números (DoubleStream, IntStream...).
Ej: En la clase Alumno, obtener el promedio de un alumno

```
public double getPromedio(){
  return  examenes.stream()
         .mapToDouble(examen->examen.getNota())
         .average().orElse(0);
}
```

SORTED

Ordena los elementos de un stream de acuerdo a la expresión que recibe como parámetro.
Ej: En la clase Alumno, obtener los exámenes ordenados por fecha de forma ascendente

```
public List<Examen> examenesOrdenadosPorFechaAsc(){
  return finales.stream()
     .sorted((ex1, ex2) ->
        ex1.getFecha().compareTo(ex2.getFecha()))
  .collect(Collectors.toList());
}
```

FINDFIRST | FINDANY

Ej: obtener el primer alumno cuyo nombre comience con una cadena dada. Si no existe ninguno, obtiene null.

```
public Alumno primerAlumnoCon(String x){
  return alumnos.stream()
.filter(alumno->alumno.getNombre().startsWith(x))
  .findFirst().orElse(null);
}
```