

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

**OBLIGATORIO ALGORITMOS Y
ESTRUCTURAS DE DATOS 1**

DOCUMENTO



Nicolas Gimenez – 291950



Cristian García – 317010

Grupo N3D

Docente: Rafael Cohen

Analista en tecnologías de la información

14/10/2024

Índice

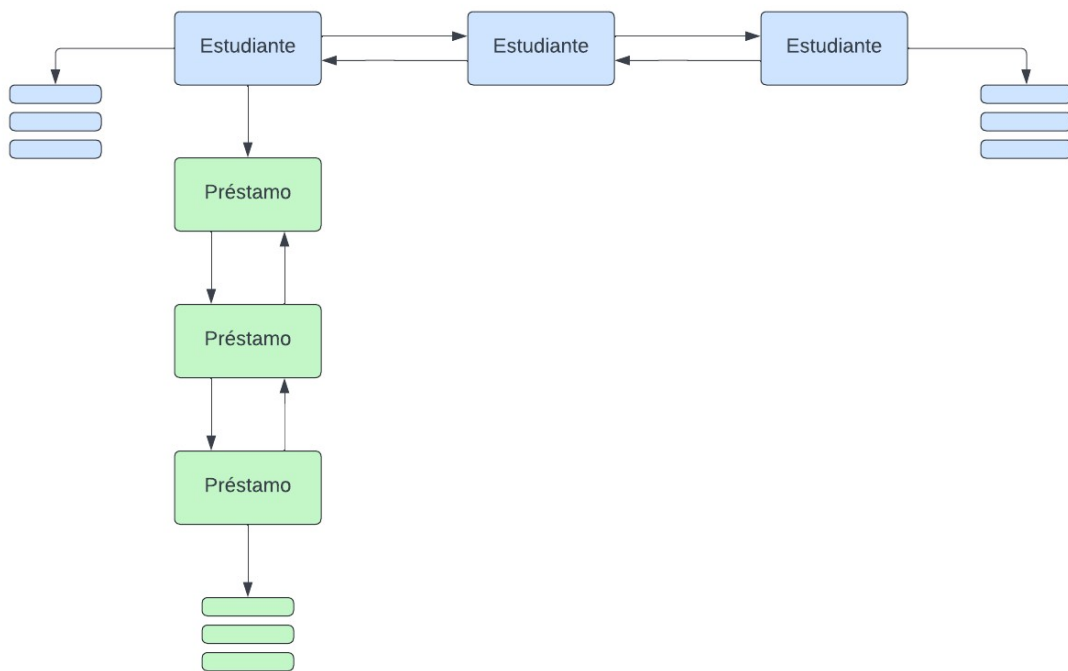
1.	Representación gráfica de estructuras.....	3
2.	Juego de prueba y evidencia	7
2.1	Agregar estudiante y mostrar el listado.	7
2.2	Obtener un estudiante por su número.	7
2.3	Eliminar estudiante y mostrar la lista.	8
2.4	Agregar 3 estudiantes de forma ordenada y mostrar la lista.....	8
2.5	Vaciar la lista de estudiantes y mostrar.	9
2.6	Agregar 4 estudiantes de forma desordenada y mostrar.....	9
2.7	Mostrar lista de libros (vacía).....	9
2.8	Agregar un libro y mostrar la lista.....	10
2.9	Agregar 4 libros ordenados y mostrar la lista.....	10
2.10	Vaciar la lista de libros.	11
2.11	Agregar 4 libros desordenados y mostrar la lista.	11
2.12	Mostrar la lista de libros según la categoría indicada.....	12
2.13	Intentar eliminar un estudiante con préstamo activo.	12

1. Representación gráfica de estructuras.

Lista doble encadenada de **estudiantes** ordenado creciente por número de estudiante.

Se define este tipo de dato para cumplir por ejemplo con el requerimiento 3.1: Se listan los datos de los estudiantes ordenados creciente por número, cargando el resultado en el valor String del retorno.

Cada estudiante contiene una lista de sus préstamos.



ESTUDIANTE

Nombre: string

Apellido: string

Número: int

PrestamosActivos: ListaDoble<Préstamo>

PRESTAMO

Libro: Libro

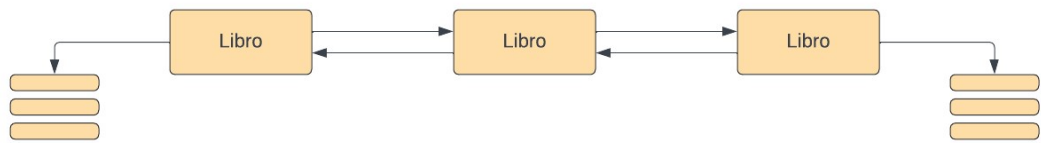
Estudiante: Estudiante

Fecha: LocalDateTime

Activo: boolean

Lista doble encadenada de **libros** ordenada creciente por ISBN.

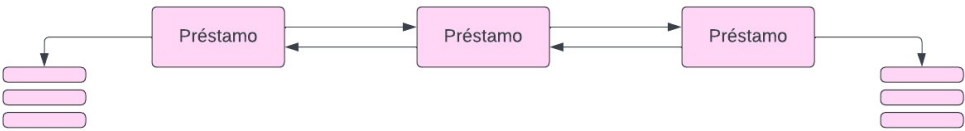
Se define este tipo de dato para cumplir por ejemplo con el requerimiento 3.2: Se listan los datos de todos los libros de la biblioteca ordenados creciente por ISBN cargando el resultado en el valor String del retorno.



LIBRO
Nombre: string
ISBN: string
Categoria: string
Total: int

Lista doble encadenada de **préstamos** ordenada creciente por ISBN.

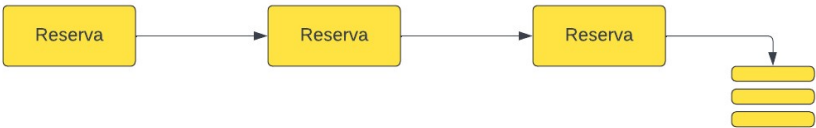
Se define este tipo de dato para cumplir por ejemplo con el requerimiento 3.2: Se deben listar el/los libros más prestados ordenado por ISBN, cargando el resultado en el valor String del retorno. En caso de existir libros con la misma cantidad “máxima de préstamos”, se deberán mostrar todos.



PRESTAMO
Libro: Libro
Estudiante: Estudiante
Fecha: LocalDateTime
Activo: boolean

Cola de **reservas**.

Se define este tipo de dato para cumplir por ejemplo con el requerimiento 2.7: Registrar una reserva de un ejemplar de un libro para un estudiante. Solo se puede realizar una reserva si no existe stock disponible en el momento para realizar el préstamo.



RESERVA
Libro: Libro
Estudiante: Estudiante
Fecha: LocalDateTime

2. Juego de prueba y evidencia

Se realizó un juego de prueba dentro del proyecto donde se ejecutan los métodos solicitados y se muestran los resultados obtenidos.

A continuación se detalla el código con la prueba y print del resultado obtenido.

2.1 Agregar estudiante y mostrar el listado.

Código:

```
sis.agregarEstudiante("Nombre", "Apellido", 1111);
sis.agregarEstudiante("Nombre", "Apellido", 1112);
System.out.println("Lista de estudiantes: " + sis.listarEstudiantes().valorString);

System.out.println("-----\n");
```

Resultado:

```
Lista de estudiantes: Nombre#Apellido#1111|Nombre#Apellido#1112
-----
```

2.2 Obtener un estudiante por su número.

Código:

```
System.out.println("Obtengo estudiante 1111: " +
sis.obtenerEstudiante(1111).valorString);

System.out.println("-----\n");
```

Resultado:

```
Obtengo estudiante 1111:Nombre#Apellido#1111
-----
```

2.3 Eliminar estudiante y mostrar la lista.

Código:

```
sis.eliminarEstudiante(1111);
```

```
System.out.println("Elimino el estudiante 1111 muestro la lista: " +  
sis.listarEstudiantes().valorString);
```

```
System.out.println("-----\n");
```

Resultado:

```
Elimino el estudiante 1111 muestro la lista: Nombre#Apellido#1112  
-----
```

2.4 Agregar 3 estudiantes de forma ordenada y mostrar la lista.

Código:

```
sis.agregarEstudiante("Nombre2", "Apellido2", 2222);
```

```
sis.agregarEstudiante("Nombre3", "Apellido3", 3333);
```

```
sis.agregarEstudiante("Nombre4", "Apellido4", 4444);
```

```
System.out.println("Agrego 3 estudiantes ordenados y muestro lista: " +  
sis.listarEstudiantes().valorString);
```

```
System.out.println("-----\n");
```

Resultado:

```
Agrego 3 estudiantes ordenados y muestro lista:  
Nombre#Apellido#1112|Nombre2#Apellido2#2222|Nombre3#Apellido3#3333|Nombre4#Apellido4#4444  
-----
```


2.5 Vaciar la lista de estudiantes y mostrar.

Código:

```
sis.Estudiantes.vaciar();  
  
System.out.println("Vaciamos la lista: " + sis.listarEstudiantes().valorString);  
  
System.out.println("-----\n");
```

Resultado:

```
Vaciamos la lista:  
-----
```

2.6 Agregar 4 estudiantes de forma desordenada y mostrar.

Código:

```
sis.agregarEstudiante("Nombre7", "Apellido7", 7777);  
sis.agregarEstudiante("Nombre1", "Apellido1", 1111);  
sis.agregarEstudiante("Nombre4", "Apellido4", 4444);  
sis.agregarEstudiante("Nombre3", "Apellido3", 3333);  
  
System.out.println("Agrego 4 estudiantes desordenados y muestro lista: \n" +  
sis.listarEstudiantes().valorString);  
  
System.out.println("-----\n");
```

Resultado:

```
Agrego 4 estudiantes desordenados y muestro lista:  
Nombre1#Apellido1#1111|Nombre3#Apellido3#3333|Nombre4#Apellido4#4444|Nombre7#Apellido7#7777  
-----
```

2.7 Mostrar lista de libros (vacía).

Código:

```
System.out.println("Lista de libros vacia: " + sis.listarLibros().valorString);  
  
System.out.println("-----\n");
```

Resultado:

```
Lista de libros vacia:  
-----
```

2.8 Agregar un libro y mostrar la lista.

Código:

```
sis.agregarLibro("NombreLibro", "ISBN", "Categoria", 150);  
System.out.println("Agrego un libro y muestro la lista: " + sis.listarLibros().valorString);  
  
System.out.println("-----\n");
```

Resultado:

```
Agrego un libro y muestro la lista: NombreLibro#ISBN#Categoria  
-----
```

2.9 Agregar 4 libros ordenados y mostrar la lista.

Código:

```
sis.agregarLibro("NombreLibro1", "ISBN1", "Filosofia", 150);  
sis.agregarLibro("NombreLibro2", "ISBN2", "Matematica", 250);  
  
sis.agregarLibro("NombreLibro3", "ISBN3", "Filosofia", 350);  
  
sis.agregarLibro("NombreLibro4", "ISBN4", "Matematica", 450);  
  
System.out.println("Agrego 4 libros ordenados y muestro la lista: " +  
sis.listarLibros().valorString);  
  
System.out.println("-----\n");
```

Resultado:

```
Agrego 4 libros ordenados y muestro la lista:  
[NombreLibro#ISBN#Categoria|NombreLibro1#ISBN1#Filosofia|NombreLibro2#ISBN2#Matematica|  
NombreLibro3#ISBN3#Filosofia|NombreLibro4#ISBN4#Matematica  
-----]
```

2.10 Vaciar la lista de libros.

Código:

```
sis.Libros.vaciar();  
  
System.out.println("Vaciamos la lista: " + sis.listarLibros().valorString);  
  
System.out.println("-----\n");
```

Resultado:

```
Vaciamos la lista:  
-----
```

2.11 Agregar 4 libros desordenados y mostrar la lista.

Código:

```
sis.agregarLibro("NombreLibro8", "ISBN8", "Filosofia", 250);  
sis.agregarLibro("NombreLibro7", "ISBN7", "Matematica", 150);  
  
sis.agregarLibro("NombreLibro2", "ISBN2", "Filosofia", 450);  
  
sis.agregarLibro("NombreLibro5", "ISBN5", "Arquitectura", 350);  
  
System.out.println("Agrego 4 libros desordenados y muestro la lista: \n" +  
sis.listarLibros().valorString);  
  
System.out.println("-----\n");
```

Resultado:

```
Agrego 4 libros desordenados y muestro la lista:  
[NombreLibro2#ISBN2#Filosofia|NombreLibro5#ISBN5#Arquitectura|NombreLibro7#ISBN7#Matematica|  
NombreLibro8#ISBN8#Filosofia  
-----
```

2.12 Mostrar la lista de libros según la categoría indicada.

Código:

```
System.out.println("Libros de Filosofía: " + sis.listarLibros("Filosofía").valorString);  
System.out.println("-----\n");
```

Resultado:

```
Libros de Filosofía:  
NombreLibro2#ISBN2#Filosofía|NombreLibro8#ISBN8#Filosofía  
-----
```

2.13 Intentar eliminar un estudiante con préstamo activo.

Código:

```
Libro libro = sis.Libros.obtenerElemento(new Libro(null, "ISBN10", null, 0));  
Estudiante estudiante = sis.Estudiantes.obtenerElemento(e1);  
  
estudiante.agregarPrestamo(libro);  
  
System.out.println("Intentar eliminar estudiante con préstamo activo: " +  
sis.eliminarEstudiante(1111).resultado);
```

Resultado:

```
Intentar eliminar estudiante con préstamo activo: ERROR_3
```

2.14 Evidencia de test.

The image displays two screenshots of a JUnit test runner interface, showing successful test results for two test classes.

Top Screenshot: sistemaAutogestion.IObligatorioTest

- Tests passed: 100,00 %
- All 17 tests passed. (0,242 s)
- Test results list:
 - sistemaAutogestion.IObligatorioTest passed
 - testAgregarLibroOk passed (0,025 s)
 - testAgregarEstudianteError1 passed (0,0 s)
 - testAgregarEstudianteError2 passed (0,001 s)
 - testAgregarEstudianteError3 passed (0,004 s)
 - testListarLibrosXCategoriaError1 passed (0,0 s)
 - testAgregarEstudianteOk passed (0,0 s)
 - testCrearSistemaDeGestion passed (0,001 s)
 - testObtenerEstudianteOk passed (0,0 s)
 - testObtenerEstudianteError1 passed (0,001 s)
 - testObtenerEstudianteError2 passed (0,001 s)
 - testListarLibros passed (0,0 s)
 - testEliminarEstudianteOk passed (0,001 s)
 - testEliminarEstudianteError1 passed (0,0 s)
 - testEliminarEstudianteError2 passed (0,001 s)
 - testEliminarEstudianteError3 passed (0,02 s)
 - testListarEstudiantes passed (0,001 s)
 - testListarLibrosXCategoriaOk passed (0,001 s)

Bottom Screenshot: sistemaAutogestion.IObligatorioTest2

- Tests passed: 100,00 %
- All 22 tests passed. (0,154 s)
- Test results list:
 - sistemaAutogestion.IObligatorioTest2 passed
 - testListarLibrosVacio passed (0,015 s)
 - testAgregarEstudianteError1 passed (0,0 s)
 - testAgregarEstudianteError2 passed (0,001 s)
 - testAgregarEstudianteError3 passed (0,004 s)
 - testListarEstudiantesIngresoOrdenado passed (0,0 s)
 - testListarLibrosXCategoriaError1 passed (0,002 s)
 - testListarLibrosIngresoNoOrdenado passed (0,002 s)
 - testAgregarEstudianteOk passed (0,0 s)
 - testCrearSistemaDeGestion passed (0,0 s)
 - testObtenerEstudianteOk passed (0,0 s)
 - testObtenerEstudianteError1 passed (0,0 s)
 - testObtenerEstudianteError2 passed (0,0 s)
 - testEliminarEstudianteOk passed (0,0 s)
 - testListarEstudiantesIngresoNoOrdenado passed (0,0 s)
 - testListarLibrosUnElemento passed (0,001 s)
 - testListarEstudiantesVacio passed (0,001 s)
 - testListarEstudiantesUnElemento passed (0,001 s)
 - testListarLibrosIngresoOrdenado passed (0,0 s)
 - testEliminarEstudianteError1 passed (0,001 s)
 - testEliminarEstudianteError2 passed (0,001 s)
 - testEliminarEstudianteError3 passed (0,013 s)
 - testListarLibrosXCategoriaOk passed (0,0 s)

