Algoritmos y Programación II
2020-1 Laboratorio Unidad 4, 5 y 6

Nombre: **Sebastian Villa Ávila**	**Codigo:** A00361589
Profesor: **Juan Manuel Reyes**	

Link repositorio: https://github.com/sebastianvilla17/racer.git

Requerimientos Funcionales:

El sistema está en capacidad de:

Nota: el laboratorio por consola también imprime los tiempos y la actualización, tener en cuenta que la interfaz gráfica se queda trabajando y en la consola se comprueba esto.

1. Agregar, buscar y eliminar elementos utilizando la estructura de datos ArrayList, mediante métodos iterativos recibiendo los datos por parámetro y realizando la operación correspondiente.
2. Agregar, buscar y eliminar elementos utilizando la estructura de datos ArrayList, mediante métodos recursivos recibiendo los datos por parámetro y realizando la operación correspondiente.
3. Agregar, buscar y eliminar elementos utilizando la estructura de datos LinkedList, mediante métodos iterativos recibiendo los datos por parámetro y realizando la operación correspondiente.
4. Agregar, buscar y eliminar elementos utilizando la estructura de datos LinkedList, mediante métodos recursivos recibiendo los datos por parámetro y realizando la operación correspondiente.
5. Agregar, buscar y eliminar elementos utilizando la estructura de datos ABB, mediante métodos iterativos recibiendo los datos por parámetro y realizando la operación correspondiente.
6. Agregar, buscar y eliminar elementos utilizando la estructura de datos ABB, mediante métodos recursivos recibiendo los datos por parámetro y realizando la operación correspondiente.
7. Medir el tiempo que toma cada operación y mostrarla en la interfaz de usuario
8. Generar la cantidad de datos aleatorios seleccionada por el usuario, ingresando el número de tipo long para realizar las operaciones correspondientes
9. Mostrar un cronometro que mida el tiempo en general que se tardó la carrera en ejecutarse
10. Mostrar una interfaz de usuario completa con los datos botones y elementos necesarios para cumplir con las restricciones pedidas por el usuario
11. Mostrar un dibujo en 2D en este caso dos círculos que estarán en movimiento de expansión y contracción durante la carrera.

Requerimientos no Funcionales:

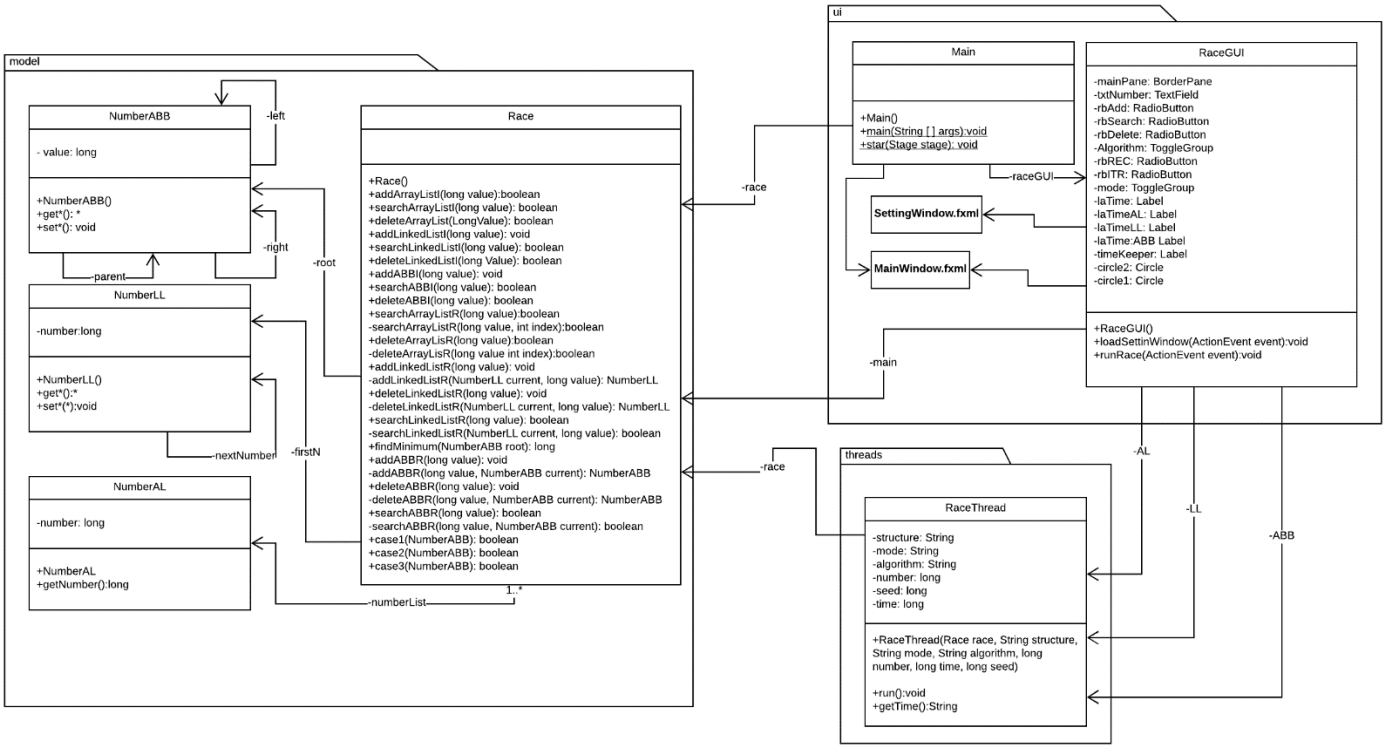
El sistema en su implementación utiliza:

1. Métodos iterativos de búsqueda, adición y eliminación de elementos en la estructura de datos ArrayList
2. Métodos iterativos de búsqueda, adición y eliminación de elementos en la estructura de datos LinkedList
3. Métodos iterativos de búsqueda, adición y eliminación de elementos en la estructura de datos ABB
4. Métodos recursivos de búsqueda, adición y eliminación de elementos en la estructura de datos ArrayList
5. Métodos recursivos de búsqueda, adición y eliminación de elementos en la estructura de datos LinkedList
6. Métodos recursivos de búsqueda, adición y eliminación de elementos en la estructura de datos ABB
7. El complemento Scene Builder para diseñar la interfaz de usuario que tendrá la interaccion con el usuario
8. Hilos (threads) para que cada estructura de datos corra su propio hilo y pueda realizar las acciones solicitadas por el usuario

Diagrama:

Link: https://app.lucidchart.com/invitations/accept/2ac4ac54-99f7-4e00-8718-73740e04be05

Imagen:



Diseño de casos de prueba:

Escenario:

Nombre	Clase	Escenario
SetUp1	NumberLL	Objetos de tipo NumberLL [1, 15, 52, 487, 13, 78, 65]

Nombre	Clase	Escenario
SetUp2	NumberABB	Objetos de tipo NumberABB [13, 25, 456, 89, 1, 45, 0, 46, 302, 54, 49, 21, 3, 5, 65]

Nombre	Clase	Escenario
SetUp3	RaceThread	Race race; String structure: Al String algorithm: delete String mode: recursive long number: 2000

Nombre	Clase	Escenario
SetUp4	RaceThread	Race race; String structure: LL String algorithm: add String mode: iterative long number: 300

Nombre	Clase	Escenario
SetUp5	NumberLL	Race race; String structure: ABB String algorithm: search String mode: recursive long number: 2530

Casos:

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **agregar** un elemento a la estructura de datos LinkedList de forma **iterativa**. Es decir, prueba que el elemento (numero), se agregue de forma correcta utilizando iteraciones.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	addLinkedListI	SetUp1		El registro correcto de un nuevo elemento (NumberLL)

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **agregar** un elemento a la estructura de datos LinkedList de forma **recursiva**. Es decir, prueba que el elemento (numero), se agregue de forma correcta utilizando recursión.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	addLinkedListR	SetUp1		El registro correcto de un nuevo elemento (NumberLL)

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **buscar** un elemento a la estructura de datos LinkedList de forma **iterativa**. Es decir, prueba que el elemento (numero), sea buscado de forma correcta con iteraciones.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	searchLinkedListI	SetUp1		La búsqueda de n elementos retornando true si se encontró y false si no se encontró

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **buscar** un elemento a la estructura de datos LinkedList de forma **recursiva**. Es decir, prueba que el elemento (numero), sea buscado de forma correcta con recursión.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	searchLinkedListR	SetUp1		La búsqueda de n elementos retornando true si se encontró y false si no se encontró

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **eliminar** un elemento a la estructura de datos LinkedList de forma **iterativa**. Es decir, prueba que el elemento (numero), sea eliminado de forma correcta con iteraciones.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	deleteLinkedListI	SetUp1		La eliminación de n elementos anteriormente creados, haciendo uso de iteraciones y retornando true si se eliminó y false si no se eliminó.

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **eliminar** un elemento a la estructura de datos LinkedList de forma **recursiva**. Es decir, prueba que el elemento (numero), sea eliminado de forma correcta con recursion.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	deleteLinkedListR	SetUp1		La eliminación de n elementos anteriormente creados, haciendo uso de recursion y retornando true si se eliminó y false si no se eliminó.

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **agregar** un elemento a la estructura de datos ABB de forma **iterativa**. Es decir, prueba que el elemento (numero), se agregue de forma correcta con iteraciones.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	addABBI	SetUp2		El registro correcto de un nuevo elemento (NumberABB), con uso de iteraciones

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **agregar** un elemento a la estructura de datos ABB de forma **recursiva**. Es decir, prueba que el elemento (numero), se agregue de forma correcta con recursión.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	addABBR	SetUp2		El registro correcto de un nuevo elemento (NumberABB), con uso de recursividad

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **buscar** un elemento a la estructura de datos ABB de forma **iterativa**. Es decir, prueba que el elemento (numero), se busque de forma correcta con iteraciones..				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	searchABBI	SetUp2		La búsqueda de n elementos retornando true si se encontró y false si no se encontró

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **buscar** un elemento a la estructura de datos ABB de forma **recursiva**. Es decir, prueba que el elemento (numero), se busque de forma correcta con recursion..				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	searchABBR	SetUp2		La búsqueda de n elementos retornando true si se encontró y false si no se encontró

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **eliminar** un elemento a la estructura de datos ABB de forma **iterativa**. Es decir, prueba que el elemento (numero), sea eliminado de forma correcta con iteraciones.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	deleteABBI	SetUp2		La eliminación de n elementos anteriormente creados, haciendo uso de iteraciones y retornando true si se eliminó y false si no se eliminó.

Objetivo de la Prueba: esta prueba se encarga de probar el método de la clase Race que tiene la finalidad de **eliminar** un elemento a la estructura de datos ABB de forma **recursiva**. Es decir, prueba que el elemento (numero), sea eliminado de forma correcta con recursión.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Race	deleteABBR	SetUp2		La eliminación de n elementos anteriormente creados, haciendo uso de recursión y retornando true si se eliminó y false si no se eliminó.

Objetivo de la Prueba: esta prueba está encargada de que el hilo se ejecute de forma correcta. Es decir, recibir los parámetros necesarios y aplicar a la condición correspondientes para retornar el tiempo que se tardó en realizar la acción debida en la estructura de datos ArrayList.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
RaceThread	run	SetUp3		el tiempo que tardo la estructura de datos en eliminar n cantidad de datos anteriormente creados, en el modo correspondiente (iterativo o recursivo).

Objetivo de la Prueba: esta prueba está encargada de que el hilo se ejecute de forma correcta. Es decir, recibir los parámetros necesarios y aplicar a la condición correspondientes para retornar el tiempo que se tardó en realizar la acción debida en la estructura de datos LinkedList.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
RaceThread	run	SetUp4		el tiempo que tardo la estructura de datos en agregar n cantidad de datos en el modo correspondiente (iterativo o recursivo)

Objetivo de la Prueba: esta prueba está encargada de que el hilo se ejecute de forma correcta. Es decir, recibir los parámetros necesarios y aplicar a la condición correspondientes para retornar el tiempo que se tardó en realizar la acción debida en la estructura de datos ABB.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
RaceThread	run	SetUp5		el tiempo que tardo la estructura de datos en eliminar n cantidad de datos anteriormente creados, en el modo correspondiente (iterativo o recursivo).