Diseño de pruebas.

Clase: *.Konoha*

- **Prueba**: El método permite conocer si un clan se encuentra en la lista de clanes registrados.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Konoha	BuscarClanRepetido (String nombreClan)	Se crea una nueva clase “Konoha” con una lista de clanes, los cuales no tienen nombres repetidos.	El nombre del clan que se quiere verificar si esta repetido.	Verdadero: Se debe encontrar un clan repetido.
Konoha	BuscarClanRepetido (String nombreClan)	Se crea una nueva clase “Konoha” con una lista de clanes, los cuales no tienen nombres repetidos.	El nombre del clan que se quiere verificar si esta repetido.	Falso: No debe encontrar ningún clan repetido.

- **Prueba**: El método puede eliminar un clan de la lista de clanes.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Konoha	EliminarClan (String nombreClan)	Se crea una nueva clase “Konoha” con una lista de clanes, los cuales no tienen nombres repetidos.	El nombre del clan que se quiere eliminar.	Se elimina el clan.

- **Prueba**: El método ordena una lista de clanes de acuerdo a su nombre.

Clase	Método probado	Escenario	Entradas	Resultado
Konoha	ordernaClanesPorNombreSelectionSort ()	Se crea una nueva clase con una lista de 5 clanes, los cuales se ingresaron de manera aleatoria.	No requiere entradas.	La lista de clanes que será ordenada debe coincidir con una lista de clanes diferente que ya se ordenó lexicográficamente de menor a mayor.
Konoha	ordernaClanesPorNombreSelectionSort ()	Se crea una nueva clase con una lista de 5 clanes, los cuales se ingresaron de mayor a menor	No requiere entradas.	La lista de clanes que será ordenada debe coincidir con una lista de clanes diferente que ya se ordenó

		siguiendo el criterio lexicografico.		lexicográficamente de menor a mayor.
Konoha	ordernaClanesPorNombreSelectionSort ()	Se crea una nueva clase con una lista de 5 clanes, los cuales se ingresaron de menor a mayor siguiendo el criterio lexicografico.	No requiere entradas.	La lista de clanes que será ordenada debe coincidir con una lista de clanes diferente que ya se ordenó lexicográficamente de menor a mayor.

- El método busca y encuentra un clan por medio de su nombre, si el nombre ingresado no corresponde a ningún a un clan, entonces no retorna nada

Clase	Método probado	Escenario	Entradas	Resultado
Konoha	BuscarClan(String nombreClan)	Se crea una nueva clase con una lista de 5 clanes, los cuales se ingresaron de manera aleatoria.	El nombre del clan que se quiere buscar.	Se debe devolver la información correspondiente al clan buscado.
Konoha	BuscarClan(String nombreClan)	Se crea una nueva clase con una lista de 5 clanes, los cuales se ingresaron de manera aleatoria.	El nombre del clan que no existe.	El método retorna un mensaje concluyendo que el clan buscado no existe.

Diseño de pruebas.

Clase: *Clan*

- **Prueba:** El método ingresa un personaje al final de la lista doblemente enlazada.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Clan	IngresarPersonajeAlfinal (String nombre, String personalidad, String fechaCreacion, int poder,Tecnica tecnicaBase)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	Nombre del personaje, personalidad, fecha de creación y poder en números. Su primera técnica sera inicializada en “null”.	Se añade un nuevo personaje al final de la lista.

- **Prueba:** El método ingresa un personaje al principio de la lista.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Clan	IngresarPersonajeAlinicio (String nombre, String personalidad, String fechaCreacion, int poder,Tecnica tecnicaBase)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	Nombre del personaje, personalidad, fecha de creación y poder en números. Su primera técnica sera inicializada en “null”.	Se añade un nuevo personaje al inicio de la lista

- **Prueba**: El método permite conocer si un personaje se encuentra en la lista de clanes registrados.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Clan	BuscarPersonajeRepetido (String nombreClan)	Se crea una nueva clase “Clan” con una lista de personajes, los cuales no tienen nombres repetidos.	El nombre de un personaje que se quiere verificar si esta repetido.	Verdadero: Se debe encontrar un personaje repetido.
Clan	BuscarPersonajeRepetido (String nombreClan)	Se crea una nueva clase “Clan” con una lista de personajes, los cuales no tienen nombres repetidos.	El nombre de un personaje que se quiere verificar si esta repetido.	Falso: No debe encontrar ningún personaje repetido.

- **Prueba**: El método puede eliminar un personaje de la lista doblemente enlazada de personajes.

Clase	Método probado	Escenario	Valores de entrada	Resultado

Clan	EliminarPersonaje (String nombre)	Se crea una nueva clase “Clan” con una lista de personajes, los cuales no tienen nombres repetidos.	El nombre del personaje que se quiere eliminar.	Se elimina el primer elemento de la lista doble.
Clan	EliminarPersonaje (String nombre)	Se crea una nueva clase “Clan” con una lista de personajes, los cuales no tienen nombres repetidos.	El nombre del personaje que se quiere eliminar.	Se elimina el ultmino elemento de la lista doble.
Clan	EliminarPersonaje (String nombreClan)	Se crea una nueva clase “Clan” con una lista de personajes, los cuales no tienen nombres repetidos.	El nombre del personaje que se quiere eliminar.	Se elimina el personaje siempre y cuando este sea diferente del primer o ultimo elemento de la lista doble.

- **Prueba:** El método permite modificar el nombre de un personaje.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Clan	modificarNombrePersonaje(String nombreActual, String nombreActualizar)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	Nombre del personaje al cual se desea cambiarle el nombre y el nuevo nombre a actualizar.	El nombre del personaje es actualizado.

- **Prueba:** El método permite modificar la personalidad de un personaje.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Clan	modificarPersonajePersonaje(String nombreActual, String nuevaPersonalidad)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	Nombre del personaje al cual se desea cambiarle la personalidad y la nueva personalidad a actualizar.	La personalidad del personaje es actualizada.

- **Prueba:** El método regresa un elemento de la lista basado en una posición que se le asigna.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Clan	retornarIndice (int pos)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	La posicion de la cual se desea obtener el personaje	Se retorna el primer personaje
Clan	retornarIndice (int pos)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	La posicion de la cual se desea obtener el personaje	Se retorna el ultimo personaje
Clan	retornarIndice (int pos)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	La posicion de la cual se desea obtener el personaje	Se retorna un personaje diferente del primer o ultimo personaje

- **Prueba:** El método permite ordenar la lista de personajes de acuerdo a su nombre usando el método (Que para nada es eficiente y es horrible) de bubble sort.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Clan	ordenarPorNombreBubbleSort()	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada en orden aleatorio.	No requiere entradas.	La lista de personajes que será ordenada debe coincidir con una lista de personajes diferente que ya se ordenó lexicográficamente de menor a mayor.
Clan	ordenarPorNombreBubbleSort()	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada ordenados de menor a mayor de acuerdo al criterio lexicógrafo.	No requiere entradas.	La lista de personajes que será ordenada debe coincidir con una lista de personajes diferente que ya se ordenó lexicográficamente de menor a mayor.
Clan	ordenarPorNombreBubbleSort()	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada ordenado de mayor a menor de acuerdo al criterio lexicógrafo.	No requiere entradas.	La lista de personajes que será ordenada debe coincidir con una lista de personajes diferente que ya se ordenó lexicográficamente de menor a mayor.

- El método busca y encuentra un clan por medio de su nombre, si el nombre ingresado no corresponde a ningún a un clan, entonces no retorna nada

Clase	Método probado	Escenario	Entradas	Resultado
Clan	BuscarSecuencialPorNombre(String nombre)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	El nombre del personaje que se quiere buscar.	Se debe devolver la información correspondiente al personaje buscado.
Clan	BuscarSecuencialPorNombre(String nombre)	Se crea un nuevo Clan al cual se le añaden 5 personajes a su lista doblemente enlazada.	El nombre de un personaje que no existe.	El método retorna un mensaje concluyendo que el personaje buscado no existe.

Diseño de pruebas.

Clase: *Personaje*

- **Prueba:** El método permite insertar una técnica al final de la lista simple de técnicas.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Personaje	InsertarAlfinal (String nombreTecnica, int factorDeInfluencia)	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple.	Nombre del la técnica y su factor de influencia.	Se añade una nueva técnica al final de la lista simple de técnicas.

- **Prueba:** El método permite insertar una técnica al inicio de la lista simple de técnicas.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Personaje	InsertarAlInicio (String nombreTecnica, int factorDeInfluencia)	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple.	Nombre del la técnica y su factor de influencia.	Se añade una nueva técnica al inicio de la lista simple de técnicas.

- **Prueba:** El método permite saber si una técnica ya se encuentra en la lista de técnicas.

Clase	Método probado	Escenario	Valores de entrada	Resultado

Personaje	repetido(String nombre)	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple.	Nombre de una técnica que ya se encuentra en la lista de técnicas.	Verdadero. La técnica ya existe en la lista de técnicas.
Personaje	repetido(String nombre)	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple.	Nombre de una técnica que no se encuentre en la lista de técnicas.	Falso La técnica no existe en la lista de técnicas.

- **Prueba**: El método puede eliminar una técnica de la lista simple de técnicas.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Personaje	EliminarTecnica (String nombreTecnica)	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple.	El nombre de la técnica que se quiere eliminar.	Se elimina el primer elemento de la lista.
Personaje	EliminarTecnica (String nombreTecnica)	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple.	El nombre de la técnica que se quiere eliminar.	Se elimina el ultimo elemento de la lista.
Personaje	EliminarTecnica (String nombreTecnica)	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple.	El nombre de la técnica que se quiere eliminar.	Se elimina la técnica siempre y cuando este sea diferente del primer o ultimo elemento de la lista doble.

- **Prueba:** El método permite ordenar la lista de técnicas de acuerdo a su factor usando el método (Que para nada es eficiente y es horrible) de insertion sort.

Clase	Método probado	Escenario	Valores de entrada	Resultado

Personaje	ordenarPorPoderInsertionSort()	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple. Estas técnicas para poder ser ordenadas requieren que estén en desorden, de otro modo el ordenamiento puede no funcionar.	No requiere entradas.	La lista de tecnicas que será ordenada debe coincidir con una lista de tecnicas diferente que ya se ordenó por factor de influencia de menor a mayor.

- **Prueba:** El método permite ordenar la lista de técnicas de acuerdo a su nombre usando el método (Que para nada es eficiente y es horrible) de selection sort.

Clase	Método probado	Escenario	Valores de entrada	Resultado
Personaje	ordenarNombresPorselectionSort()	Se crea un nuevo Personaje al cual se le añaden 9 técnicas a su lista simple. Estas técnicas para poder ser ordenadas requieren que estén en desorden, de otro modo el ordenamiento puede no funcionar.	No requiere entradas.	La lista de técnicas que será ordenada debe coincidir con una lista de técnicas diferente que ya se ordenó por factor de influencia de menor a mayor.

Diseño de pruebas.

Clase: *Técnica*

Clase	Método probado	Escenario	Valores de entrada	Resultado
Técnica	compareTo(Tecnica o)	Se crea una nueva tecnica cuyo atributo “nombreTecnica” = “Uzugan”	Una tecnica con su atributo “nombreTecnica” = “Valdio”	Verdadero, el resultado debe ser mayor que 0.
Técnica	compareTo(Tecnica o)	Se crea una nueva tecnica cuyo atributo “nombreTecnica” = “Uzugan”.	Una tecnica con su atributo “nombreTecnica” = “Alma”	Verdadero, el resultado debe ser menor que 0.

Técnica	compareTo(Tecnica o)	Se crea una nueva tecnica cuyo atributo “nombreTecnica” = “Uzugan”	Una tecnica con su atributo “nombreTecnica” = “Uzugan”	Verdadero, el resultado debe ser igual a 0.