**Objetivos**

**Unidad 3: Estructuras Lineales Enlazadas**

Al finalizar esta unidad, el estudiante estará en capacidad de:

OE5.1 Utilizar estructuras enlazadas de objetos para modelar grupos de atributos no primitivos de tamaño flexible.

OE5.2 Escribir los algoritmos necesarios para manipular estructuras lineales que almacenan sus elementos enlazándolos entre ellos.

**Preparación**

* Lea cuidadosamente el enunciado, la documentación suministrada y cada uno de los puntos que debe desarrollar antes de empezar su desarrollo. Pregunte a su profesor cualquier duda respecto al enunciado o a los requerimientos funcionales que debe desarrollar.
* El trabajo debe ser realizado individualmente.
* El trabajo será entregado en la fecha y hora establecida en Moodle.
* Usted debe utilizar git para manejar el versionamiento de su trabajo desarrollado localmente y manejar como repositorio remoto **privado**, un repositorio en GitHub o un proyecto GitLab. Su trabajo debe ser gestionado con git desde el inicio del desarrollo, los commit deben hacerse regularmente, así como los push al remoto. Esto se verificará con las fechas de los commits. En el momento de la fecha de entrega máxima, usted debe hacer público su repositorio. Recuerde las convenciones de nombre y estructura de directorios indicadas en esta [presentación de Git](https://drive.google.com/open?id=1RmergYAoyEvs-Q8g6gklipoKqeHyRhWNW8N77ewTQDQ).

**Enunciado**

Naruto es un manga escrito e ilustrado por Masashi Kishimoto y su adaptación al anime es dirigida por Hayato Date. La obra narra la historia de un ninja adolescente hiperactivo, posteriormente adulto llamado Naruto Uzumaki, quien aspira a convertirse en Hokage, con el propósito de ser reconocido como alguien importante dentro de su aldea y entre sus amigos.

La universidad tiene un proyecto de crear un videojuego y para ello le ha solicitado a usted que le ayude a construir algunas de las funcionalidades. Para esto es importante saber que todos los personaje de la Naruto shippuden tiene un puntaje para determinar su poder, por ejemplo: Naruto: 899, Naruto sennin: 1200, Naruto kyubi mode: 3.000m, Naruto sabio de los 6 caminos: 4.200, Naruto sabio de los 6 caminos Fusionado con kurama:14.500, Sasuke: 900, Sasuke susanno: 1.900, Sasuke sharingan eternal: 2.300, Kakachi: 1.400, Yiraiya: 3.000, kiba: 740, akamaru: 230, konohamaru: 101, tsunade: 890, Neji: 830, etc.

Del juego se desea tener clanes, por cada clan se tienen los personajes y de ellos se sabe su nombre, su personalidad, la fecha de creación del personaje, su poder y las técnicas usadas para este personaje. De las técnicas nos interesa saber el nombre de la técnica y el factor que influye en el poder de cada personaje, también es importante tener en orden ascendentemente todos las técnicas de un personaje por el factor. Se debe tener en cuenta que el nombre del clan, del personaje y de la técnica no se repitan. El inversionista del proyecto ha solicitado que los personajes sean implementados usando una lista doblemente enlazada y las técnicas una lista sencilla.

En su programa usted deberá:

1. Realizar el CRUD de cada una de las clases del sistema (del original en inglés: Create, Read, Update and Delete) y realizar la respectiva persistencia.
2. Implementar y utilizar los tres métodos de ordenamiento clásicos: burbuja, selección e inserción.
3. Implementar y utilizar las dos estrategias de búsqueda clásicas (secuencial).
4. Utilizar la interface Comparable.
5. Utilizar la interface Comparator.

Ya que en el programa hay que hacer diferentes ordenamientos y búsquedas, para cumplir con los requisitos anteriores, usted utilizará un algoritmo de ordenamiento y búsqueda en un caso y otro algoritmo ordenamiento búsqueda en otro caso, implementando todos los vistos hasta la fecha.

**Entregables.**

**1.** Requerimientos Funcionales y no funcionales.

**2.** Diagrama de clases de modelo y de la interfaz del usuario (no generado automáticamente).

**3.** Implementación completa de todos los requerimientos en Java.

**4.** Tabla de trazabilidad de requerimientos vs métodos (tabla con una columna de los requerimientos, tal que, por cada requerimiento se indica en la columna siguiente todos los métodos que contribuyen a resolverlo).

**5.** Investigue e incluya en su reporte de laboratorio qué es trazabilidad en desarrollo de software.

**Fecha de Entrega Máxima:** martes 24 de septiembre de 2019 a las 11:59 AM a través de Moodle. El laboratorio debe trabajarse y entregarse **individualmente**. Lo que usted debe entregar de su trabajo es la url de su repositorio en GitHub o proyecto en GitLab. Recuerde que el repositorio o proyecto debe ser privado durante el desarrollo del laboratorio y hacerse público solo en el momento justo de la entrega máxima indicada aquí.

Requerimientos:

tener en orden ascendentemente todos las técnicas de un personaje por el factor

1. Implementar y utilizar los tres métodos de ordenamiento clásicos: burbuja, selección e inserción.

|  |  |
| --- | --- |
| **Nombre** | **RF1:** Ordenar las técnicas de un personaje |
| **Resumen** | Permite ordenar las técnicas de un personaje de forma ascendente por sus factores. |
| **Entrada** | El personaje que desea ordenarles las técnicas. |
| **Salida** | un listado ordenados las técnicas del personaje. |

Se debe tener en cuenta que el nombre del clan, del personaje y de la técnica no se repitan

|  |  |
| --- | --- |
| **Nombre** | **RF2:** Verificar que el nombre del clan, personaje y técnica sean igual |
| **Resumen** | Permite verificar que los nombre de los clanes, personajes y de las técnicas no sean iguales entre sí. |
| **Entrada** | -.-.- |
| **Salida** | Un Booleano donde me va a ayudar a el método de agregar algún objeto para verificar si es que está creando ya existe. |

Create, Read, Update and Delete

Del personaje:

|  |  |
| --- | --- |
| **Nombre** | **RF3:** Crearun personaje(un Naruto…..) |
| **Resumen** | Permite crear un personaje con sus respectivos atributos y se verifica si el personaje ya está creado que mande un mensaje donde diga que ya está agregado. |
| **Entrada** | Todos los parámetros para poder crea un personaje. |
| **Salida** | Imprimir en pantalla donde diga que se han agregado correctamente el personaje o que ya está que creer otro. |

|  |  |
| --- | --- |
| **Nombre** | **RF4:** Eliminar un personaje |
| **Resumen** | Permite eliminar un personaje, donde el usuario digita el nombre del personaje para eliminarlo. |
| **Entrada** | Nombre del personaje que desea eliminar |
| **Salida** | Mostrar en pantalla que el personaje fue eliminado correctamente o donde diga que este personaje no existía. |

1. Implementar y utilizar los tres métodos de ordenamiento clásicos: burbuja, selección e inserción.
2. Implementar y utilizar las dos estrategias de búsqueda clásicas (secuencial).

|  |  |
| --- | --- |
| **Nombre** | **RF5:** Buscar un personaje |
| **Resumen** | Permite buscar un determinado personaje que quiere ver el dueño. |
| **Entrada** | Nombre del personaje que desea buscar. |
| **Salida** | Mostrar en pantalla el personaje que está buscando o un mensaje que diga que no lo encontró. |

Del clan:

|  |  |
| --- | --- |
| **Nombre** | **RF6:** Crear un clan |
| **Resumen** | Permite crear un clan con el nombre de este, y se verifica si hay otro clan con el mismo nombre. |
| **Entrada** | El nombre que le quiere colocar al clan |
| **Salida** | Mostrar un mensaje donde diga que el clan fue creado o que ya está creado |

|  |  |
| --- | --- |
| **Nombre** | **RF7:** Eliminar un clan |
| **Resumen** | Permite eliminar un clan con el nombre de este haciendo que todo lo que este dentro del se elimine. |
| **Entrada** | El nombre del clan que desea eliminar |
| **Salida** | Mostrar en pantalla si se pudo eliminar el clan correctamente o que el clan no existía. |

1. Implementar y utilizar los tres métodos de ordenamiento clásicos: burbuja, selección e inserción.
2. Implementar y utilizar las dos estrategias de búsqueda clásicas (secuencial).

|  |  |
| --- | --- |
| **Nombre** | **RF8:** Buscar un clan |
| **Resumen** | Permite buscar un clan por el nombre imprimiendo este y todo sus integrantes, si no existe retorna un mensaje donde le indica a el usuario que no existe. |
| **Entrada** | El nombre del clan que desea buscar. |
| **Salida** | Mostrar en pantalla un mensaje donde está el clan que está buscando el usuario o un mensaje donde diga que no se encontró el clan. |

De una tecnica:

|  |  |
| --- | --- |
| **Nombre** | R**F9:** Crear una técnica |
| **Resumen** | Permite crea una técnica que son las que tienen los personajes, verifica además si ya está creada la técnica. |
| **Entrada** | Los respectivos atributos una técnica. |
| **Salida** | Mostrar en pantalla un mensaje donde diga que se ha agregado la técnica correctamente o un mensaje donde diga que esa técnica ya esta creada. |

|  |  |
| --- | --- |
| **Nombre** | **RF10:** Eliminar una técnica. |
| **Resumen** | Permite eliminar una técnica que el usuario para el nombre por parámetro. |
| **Entrada** | Número de la técnica que desea eliminar. |
| **Salida** | Mensaje donde diga que se ha eliminado la técnica o no se pudo encontrar la técnica que desea eliminar. |

1. Implementar y utilizar los tres métodos de ordenamiento clásicos: burbuja, selección e inserción.
2. Implementar y utilizar las dos estrategias de búsqueda clásicas (secuencial).

|  |  |
| --- | --- |
| **Nombre** | **RF11:** Buscar una técnica. |
| **Resumen** | Permite buscar un técnica por su nombre. |
| **Entrada** | Nombre de la técnica que desea buscar |
| **Salida** | Mensaje donde diga que se ha eliminado la técnica correctamente o no se pudo encontrar el dueño que desea eliminar. |

|  |  |
| --- | --- |
| **Nombre** | **RF12:** Actualizar la información. |
| **Resumen** | Permite actualizar la información de cualquiera objeto cambiándole cualquiera de sus atributos. |
| **Entrada** | El cambio del atributo que desea cambiar, ósea el usuario señala un objeto y dice que le quiere cambiar el nombre y digita el nombre nuevo del objeto. |
| **Salida** | Mensaje donde diga que se pudo actualizar la información correctamente. |

Diagrama de clases:

Trazabilidad:

|  |  |
| --- | --- |
| **Requerimiento** | **Metodos** |
| **RF1:** Ordenar las técnicas de un personaje | HoldingPets  orderingClubs(); |
| RF2: Crear listado de los dueños | ClubPets  orderingOwner(); |
| RF3: Crear listado de las mascotas | Owner:  orderingPet(); |
| RF4: Crear listado ordenado de dueños según número de mascotas que posee. | ClubPets  orderingOwnersNumPets(); |
| RF5: Crear listado ordenado de clubs según el número de dueños | ClubPets  orderingClubNumOwner(); |
| RF6: Búsqueda binaria o secuencia de un objeto por determinado criterio. | HoldingPets:  searchBiTraByCritierion (); |
| RF7: Añadir un club | HoldingPets:  addClubPets(); |
| RF8: Añadir una mascota. | Owner:  addPet(); |
| RF9: Añadir un dueño. | ClubPets:  addOwner(); |
| RF10: Eliminar un club. | HoldingPets:  removeClubPets(); |
| RF11: Eliminar un dueño. | ClubPet:  removeOwner(); |
| RF12: Eliminar una mascota. | Owner:  removePet(); |
| RF13: Guardar la información. | HoldingPets:  actualizationInf(); |