Universidad Católica del Norte
Facultad de Ingeniería y Ciencias Geológicas
Departamento de Ingeniería de Sistemas y Computación

Taller N°2

Informe

Integrantes: Christopher Córtes,
Romina Espinoza
Correo:
Christopher.cortes@alumnos.ucn.cl
Romina.espinoza@alumnos.ucn.cl
Rut: 20.415.086-9, 20.960.710-7
Profesor: Tomás Reimann
Paralelo: C2
Fecha: 29-05-2024

Índice

Introducción

En este informe se presentará el diagrama de dominio, diagrama de clases para el taller 2 de programación avanzada, se utilizará como apoyo el diagrama de dominio y clases a la hora de mostrar el proyecto al cliente.

El diagrama de dominio y clases serán fundamentales para el desarrollo del código ya que será nuestra guía a lo largo de la creación del programa. Al mismo tiempo se aplicará lo aprendido en clases en el avance del taller.

Enunciado

“Los juegos de cartas coleccionables son un tipo de juego de cartas variadas, tanto en tipos y características, al igual que la forma en que se juegan, pero lo que la gran mayoría comparte es el requerimiento de construir una baraja o un mazo, es por esto por lo que para este taller el cliente solicitó a los estudiantes de programación avanzada un gestor de mazos para el juego de cartas “Magic: The Gathering”.

Para esta tarea el cliente otorgó una lista de definiciones que permitirá a los desarrolladores estar más familiarizados con el contexto del juego de cartas.”

Diagrama de dominio

Habrá 4 entidades en este diagrama conformados por: carta, mazo, listaDeMazos, sistemaDeCartas.

Cada una de estas tendras sus respectivas características:

- **carta:** Nombre, coste, tipo, power, toughness, cmc, color. Agregar carta, eliminar carta.
- **mazo:** Nombre de mazo, cantidad de cartas.
- **listaDeMazos:** cantidad de mazos, ver mazo, agregar nuevo mazo, eliminar mazo.
- **sistemaDeCartas:** modificar deck, modificar sideboard, construir mazo, ver mazos.

Dentro de la relación carta - mazos: carta formará mazos.

Dentro de la relación mazos - listaDeMazos: Mazos puede tener listaDeMazos.

Dentro de la relación listaDeMazos - sistemaDeCartas: listaDeMazos sera parte de sistemaDeCartas.

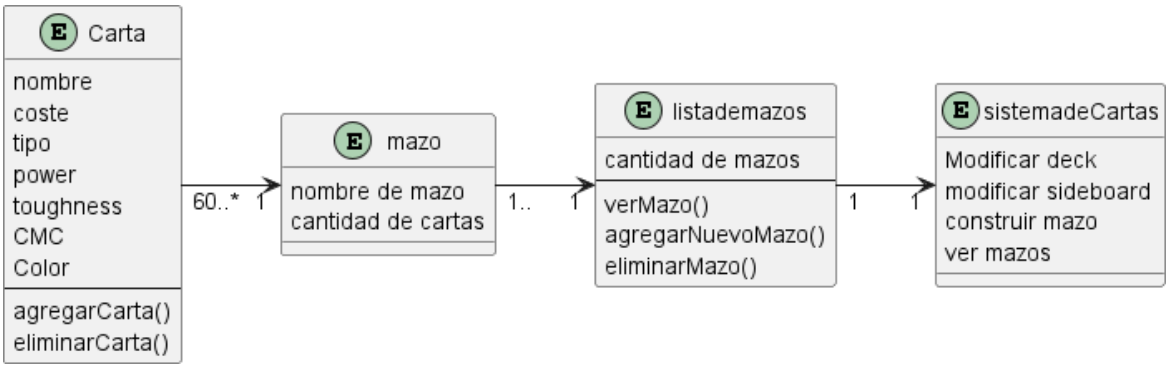


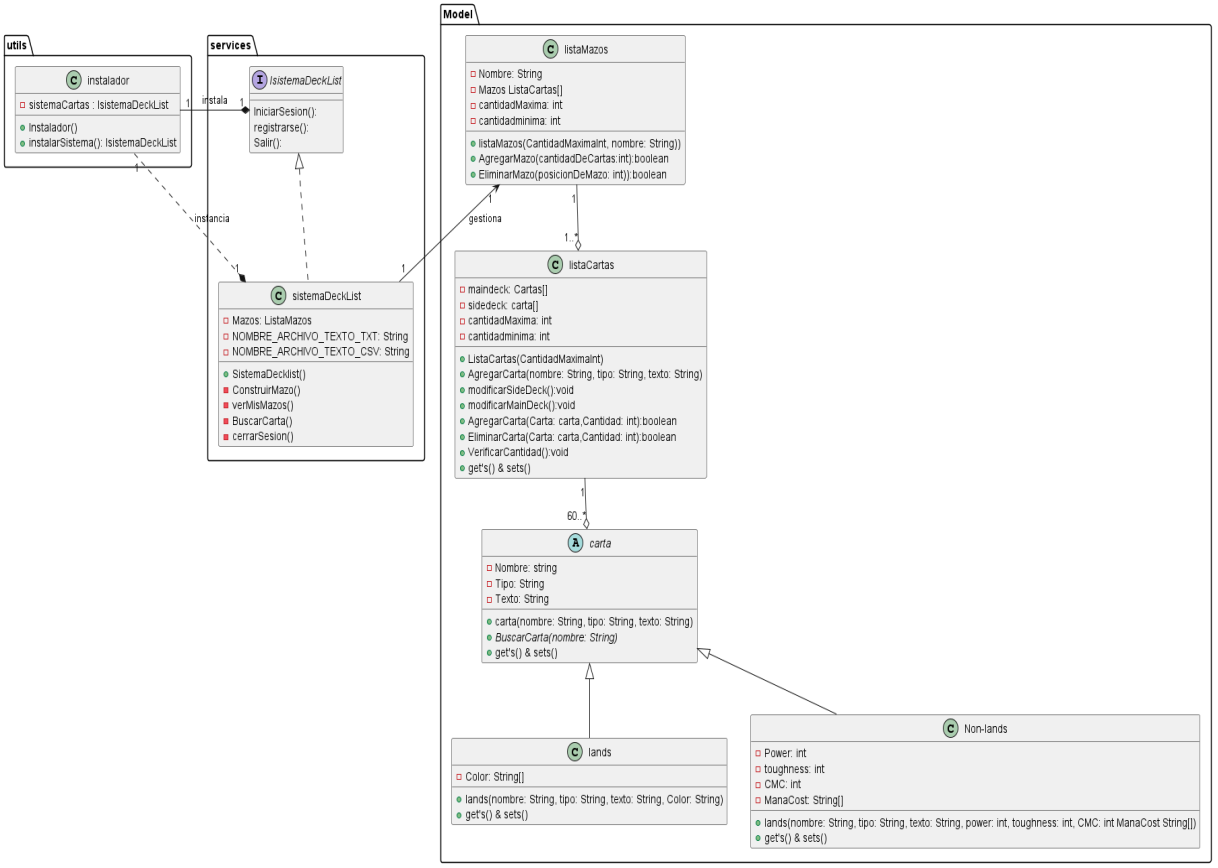
Diagrama de clases

Consta de 7 clases compuestas por:

- Package model: dentro de esta tendrá 5 clases, una de estas será una clase abstracta y dos que estarán estrechamente relacionadas.
    - Abstract Carta: Nombre, tipo, texto y sus respectivos set's y get's. esta clase es relacionada junto a la siguientes clases:
        - lands extends carta: Color, necesita de la clase abstracta ya que tiene características en común con otras clases más, debido a esto se utiliza herencia.
        - Non-lands extends carta: Power, toughness, cmc, manaCost y sus respectivos get´s y set´s.

    - **ListaCartas:**Maindeck, sidedeck, cantidadMaxima, cantidadMinima, agregarCarta, modificarSidedeck, modificarMainDeck, verificarCantidad, get's y set's
    - **ListaMazos:** nombre, mazos ListaCartas, cantidadMaxima, cantidadMinima, agregarMazo, eliminarMazo.
- **Package utils:** estará a cargo de tener la clase instalador para hacer las funciones pedidas, esta clase estara conformada de las siguiente forma:
    - **Instalador:** SistemaCarta.
    - **Package services:** tendrá el interface junto a una clase:
        - **ISistemaDeckList:** El uso de este interface sera de iniciar sesion, registrarse y salir.
        - **SistemaDeckList:** Esta clase implementa el interface ISistemaDeckList de esta manera internamente tendra que leer los archivos dados de las cartas magic y hacer cada función pedida que formarán parte de esta clase(mazos ListaMazos, NOMBRE_ARCHIVO_TECTO_TXT, NOMBRE_ARCHIVO_TEXTO_CSV, construir Mazo, verMisMazos, BuscarCarta, cerrarSesion)

Dentro de la relación **ListaMazo - SistemaDeckList:** ListaMazo solamente puede contener un sistemaDeckList, este gestiona ListaMazo.
Dentro de la relación **Instalador - ISistemaDeckList:** Instalador puede contener 1 ISistemaDeckList, este se instalará en ISistemaDeckList.
Dentro de la relación **Instalador - SistemaDeckList:** Instalador puede tener 1 a más SistemaDeckList.
Dentro de la relación **ListaCartas - Carta: ListaCarta** contiene una carta, mientras que carta puede tener minimo 60 o más.
Dentro de la relación **ListaMazos - ListaCartas:** ListaMazos contiene una ListaCartas, mientras que ListaCartas puede tener 1 ó más cliente.


utils
services
Model
instalador
sistemaCartas : IsistemaDeckList
Instalador()
instalarSistema(): IsistemaDeckList
instala
IsistemaDeckList
IniciarSesion():
registrarse():
Salir():
instancia
sistemaDeckList
Mazos: ListaMazos
NOMBRE_ARCHIVO_TEXTO_TXT: String
NOMBRE_ARCHIVO_TEXTO_CSV: String
SistemaDecklist()
ConstruirMazo()
verMisMazos()
BuscarCarta()
cerrarSesion()
gestiona
listaMazos
Nombre: String
Mazos ListaCartas[]
cantidadMaxima: int
cantidadminima: int
listaMazos(CantidadMaximaInt, nombre: String))
AgregarMazo(cantidadDeCartas:int):boolean
EliminarMazo(posicionDeMazo: int)):boolean
listaCartas
maindeck: Cartas[]
sidedeck: carta[]
cantidadMaxima: int
cantidadminima: int
ListaCartas(CantidadMaximaInt)
AgregarCarta(nombre: String, tipo: String, texto: String)
modificarSideDeck():void
modificarMainDeck():void
AgregarCarta(Carta: carta,Cantidad: int):boolean
EliminarCarta(Carta: carta,Cantidad: int):boolean
VerificarCantidad():void
get's() & sets()
carta
Nombre: string
Tipo: String
Texto: String
carta(nombre: String, tipo: String, texto: String)
BuscarCarta(nombre: String)
get's() & sets()
lands
Color: String[]
lands(nombre: String, tipo: String, texto: String, Color: String)
get's() & sets()
Non-lands
Power: int
toughness: int
CMC: int
ManaCost: String[]
lands(nombre: String, tipo: String, texto: String, power: int, toughness: int, CMC: int ManaCost String[])
get's() & sets()

Código

1. **Menú Principal:** Al iniciar el programa se leerán dos archivos texto con sus registros, la cual uno tendrá la mayoría de cartas excepto las tierras y en el otro se hallará las cartas tierras.

   Se imprimirá a través de la pantalla un menú que indicará “1-. Iniciar sesión”, “2-. Registrar”, “3-.cerrar sesión”.

   a. **Iniciar sesión:** Se ingresa por teclado el nombre y contraseña del usuario, si al ingresar los datos no están en el sistema no permitirá el acceso, por otro lado si las credenciales fueran correctas se mostrará un submenú. Todo esto se hará a través de la clase booleana inicioDeSesion, internamente tendrá un ciclo for para buscar en la lista si se encuentran los datos ingresados. Cabe la posibilidad de que no se encuentre el usuario o contraseña por ende aparecerá como opción registrarse.
   b. **Registrarse:** Si el usuario indica la opción registrar podrá crear un usuario y contraseña, esto será posible por medio de la clase ListaUsuario que utilizara un subprograma boolean llamado agregarUsuario que creará un lista en donde se guardará todo aquel que se registre con su respectivo usuario y contraseña.
   c. **Salir:**
2. **Submenú 1:** Luego de iniciar sesión se abrirá el siguiente submenú
   a. **Construir mazo:** mediante esta opción se abrirá un tercer submenú que se centrará en la creación de mazo, por medio de la clase SistemaDeckList será que se construirá este menú.
      i. **Crear mazo nuevo**
      ii. **Modificar uno existente**

      Al escoger cualquiera de estas dos opciones se mostrará por pantalla las siguientes:

      1. **Añadir carta:** Está dentro de la clase listaMazos, por medio del subprograma void agregarMazos se pedirá el nombre de la carta que buscara el sistema, si la encuentra debe consultar cuantas copias se añadirán a la baraja. Si no cumple el formato se desplegará un mensaje indicando el error y se consultará nuevamente qué carta busca. Al guardar los datos del mazo.
      2. **Eliminar carta:**

3. **Buscar carta:** Está dentro de la clase ListaCartas con un subprograma llamado buscarCartas que llama la clase abstracta Carta, dentro de esta está formado por un ciclo for para poder recorrer la lista y encontrar la carta indicada por el nombre de esta. arroja por pantalla todos los detalles de la carta.
4. **Modificar Sideboard:**
5. **volver**

iii. **volver**

b. **Ver mis mazos:** Abrirá un menú con los mazos respectivos a escoger para imprimir por pantalla.
c. **Buscar carta:** Función parecida dentro del submenú de modificar y crear mazo, sin embargo ocurrirá dentro de la clase land. Necesitará que el usuario indique el nombre de la carta a buscar.
d. **Cerrar sesión:** Al cerrar sesion se debe de guardar en memoria las modificaciones de los mazos junto a los nuevos que se hayan agregado, esto ocurrirá en la clase SistemDeckList dentro de un subprograma boolean que contendrá tanto la lectura como el guardado de las modificaciones, al cerrar el archivo se procesa un ciclo para crear un archivo texto con las nuevas modificaciones aplicadas en el mazo principal como en el mazo de cartas tierras y al mismo tiempo volver hacer una lectura con todo los detalles de los mazos.

Conclusión

El uso del diagrama de clases junto a herencia y ordenamiento nos ha dado un mayor entendimiento en la materia y su aplicación en este taller, a través de esto hemos logrado tener una mejor estructura al momento de avanzar el código y una manera más amplia de leer enunciados obteniendo en el momento las posibles conexiones entre cada clase leída.

Horas trabajadas por estudiante:

- El diagrama de dominio se hizo por partes, por lo que fue un total de 1 hora y 30 min.
    - Christopher
        - 1 hora de planificación
        - 15 min de correcciones
        - 15 creación

    La creación del informe fue de la misma manera, tomandonos 1 hora en ser acabado.

- El diagrama de clases de igual forma fue hecha en conjunto, con un total de 1 hora y 30 min aproximadamente:
    - Christopher
        - 10 min de planificación
        - 45 min de correcciones
        - 40 min en la creación

    - Romina
        - 25 min junto a Cristopher de la creación del diagrama de clases
        - Respecto al informe fue hecho en una hora por parte de Romina.

- Código:
    - Christopher
        - 1 hora de planificación
        - 1 hora de correcciones
        - 16 horas en la creación
    - Romina
        - 1 hora en modificaciones del código.
        - El informe tomó un total de 3 horas en la creación y organización de esto.