Las magias polimórficas de Harry Potter

Alumno: Daniel Silva.

Profesor: Pablo Vidal.

Asignatura: Paradigmas de Programación

Fecha: 29/11/2020

Índice

En el presente informe se detallará la tanto la implementación como los criterios de diseño aplicados al proyecto Las magias polimórficas de Harry Potter.

La problemática a solucionar consistía en modelar las entidades que componen al mundo de Harry Potter con la mayor precisión posible, así como también la interacción entre ellas de forma digital. Para esto, se utilizó el paradigma Orientado a objetos y el lenguaje de programación Java.

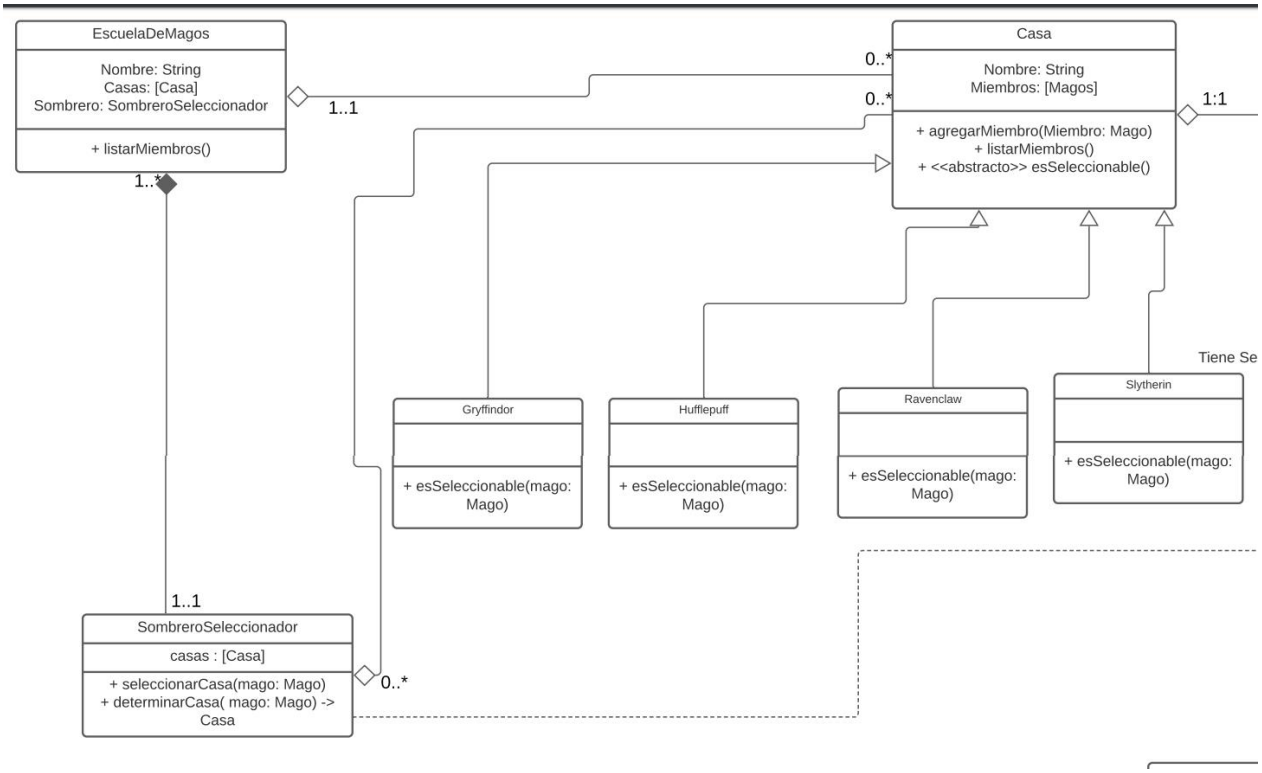
Para la prueba del correcto funcionamiento del proyecto, se dieron pautas y requerimientos que se verán implementados en un archivo .jar adjunto con el informe, así como también el diagrama de clases asociado al modelo, el cual se va a detallar a continuación:

Diagrama de Clases

Como el diagrama de clases del proyecto es de una magnitud significativa, la explicación del mismo se procederá en 3 secciones, estas secciones también están delimitadas en el *scaffolding* del proyecto y son las siguientes:

- Instituciones: Entidades relacionadas al funcionamiento de las instituciones
- Magos: Entidades que componen y se derivan del concepto de Mago.
- Hechizos: Entidades relacionadas a los hechizos y variaciones.

Instituciones



La primera parte del diagrama de clases posee 3 clases principales y 4 subclases, la clase Escuela de Magos, actúa como contenedor de las casas asociadas a la escuela, por eso se da una relación de agregación entre ambas.

Se tomó como criterio de diseño que la entidad Escuela de Magos no conozca de la clase Mago, si no que los magos tengan una relación de agregación a las casas, esto se planteó de esta manera ya que, por la implementación exigida, los magos debían pertenecer a una casa, y si también eran agregados a la Escuela de Magos se iban a duplicar los datos en ambas clases

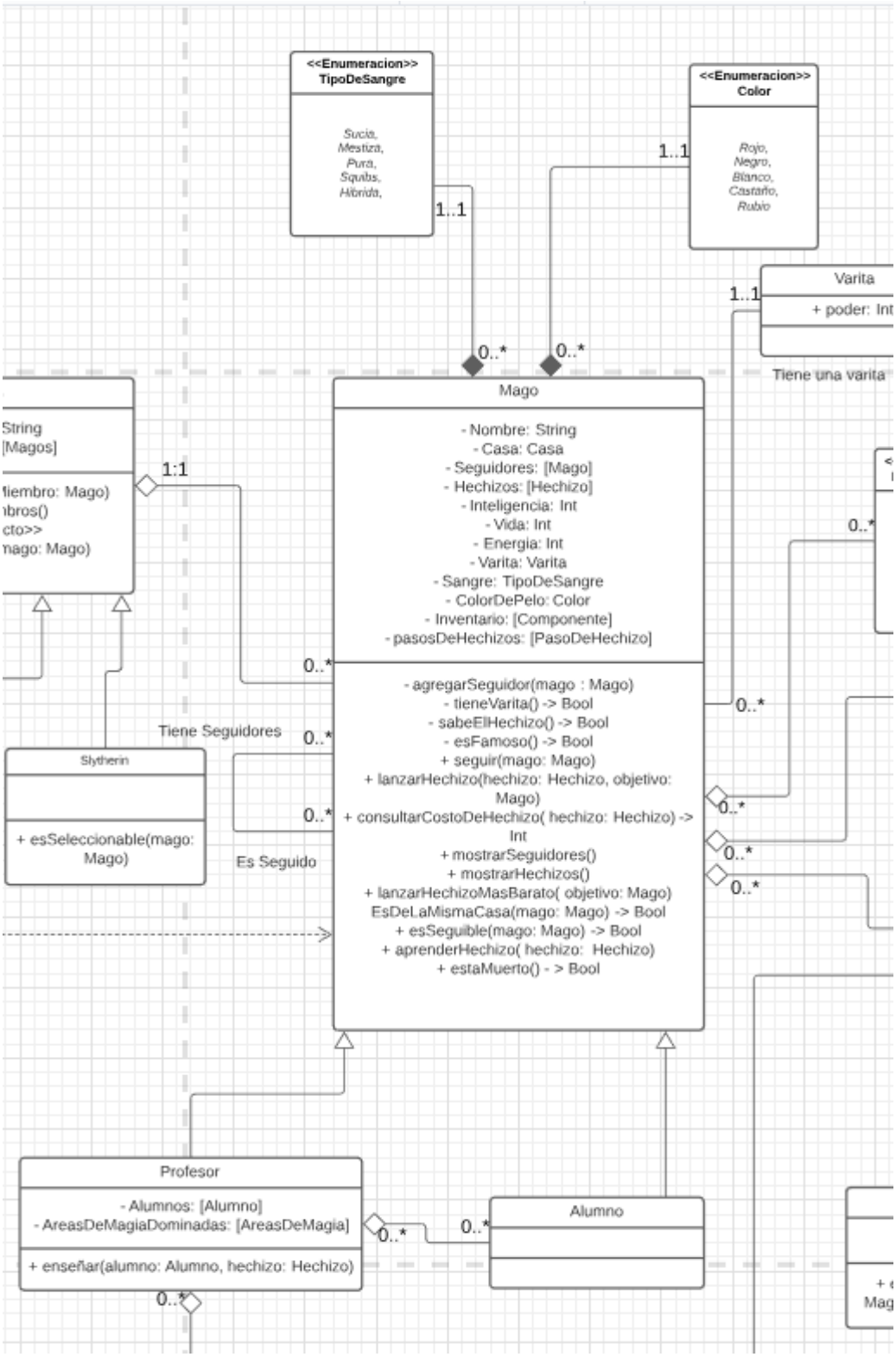
esto podría traer problemas de inconsistencia de los datos, de escalabilidad y es un uso ineficiente de la memoria.

Toda escuela de magos debe poseer un Sombrero Seleccionador, el cual es inicializado mediante las casas pertenecientes a la escuela. El sombrero es el mecanismo por el cual la escuela asigna a sus alumnos en casas, es por esto que, la responsabilidad de este objeto se ve plasmada en sus 2 métodos que determinan y asignan la casa correspondiente al mago, según los criterios especificados por la lista de casas de la escuela.

Respecto a la entidad Casa, su función es contener a los Magos asignados por el sombrero, para esto posee comportamientos para agregar miembros y listar sus miembros, este último es llamado por la Escuela de magos para poder listar los miembros de la escuela completa.

La entidad Casa, tiene un método abstracto llamado esSeleccionable, que es el que le dice al sombrero seleccionador si el mago en selección, puede ir a la casa especificada, este método abstracto se implementa de forma polimórfica en las subclases Gryffindor, Hufflepuff, Ravenclaw, Slytherin ya que cada una tiene sus condiciones y su forma de decir cuando un miembro es apto de unirse a la casa.

Magos


<<Enumeracion>>
TipoDeSangre
Sucia,
Mestiza,
Pura,
Squibs,
Hibrida,
<<Enumeracion>>
Color
Rojo,
Negro,
Blanco,
Castaño,
Rubio
1..1
1..1
Varita
+ poder: Int
1..1
0..*
0..*
Tiene una varita
Mago
- Nombre: String
- Casa: Casa
- Seguidores: [Mago]
- Hechizos: [Hechizo]
- Inteligencia: Int
- Vida: Int
- Energia: Int
- Varita: Varita
- Sangre: TipoDeSangre
- ColorDePelo: Color
- Inventario: [Componente]
- pasosDeHechizos: [PasoDeHechizo]
- agregarSeguidor(mago : Mago)
- tieneVarita() -> Bool
- sabeElHechizo() -> Bool
- esFamoso() -> Bool
+ seguir(mago: Mago)
+ lanzarHechizo(hechizo: Hechizo, objetivo: Mago)
+ consultarCostoDeHechizo(hechizo: Hechizo) -> Int
+ mostrarSeguidores()
+ mostrarHechizos()
+ lanzarHechizoMasBarato(objetivo: Mago)
EsDeLaMismaCasa(mago: Mago) -> Bool
+ esSeguible(mago: Mago) -> Bool
+ aprenderHechizo(hechizo: Hechizo)
+ estaMuerto() - > Bool
String
[Magos]
iembro: Mago)
bros()
cto>>
nago: Mago)
1:1
0..*
0..*
0..*
0..*
0..*
0..*
0..*
Tiene Seguidores
Es Seguido
Slytherin
+ esSeleccionable(mago: Mago)
Profesor
- Alumnos: [Alumno]
- AreasDeMagiaDominadas: [AreasDeMagia]
+ enseñar(alumno: Alumno, hechizo: Hechizo)
0..*
0..*
0..*
Alumno

Los magos son la clase central de todo el proyecto y la que más funcionalidad tiene, además de que la mayoría de las relaciones convergen en la entidad Mago.

Un mago se compone de distintos atributos, dos de los principales que son entidades externas son el Tipo de Sangre y el Color de Pelo, estos atributos fueron modelados mediante enumeraciones para una mejor legibilidad del código y consistencia de tipos, ya que son atributos a evaluar por el sombrero seleccionador.

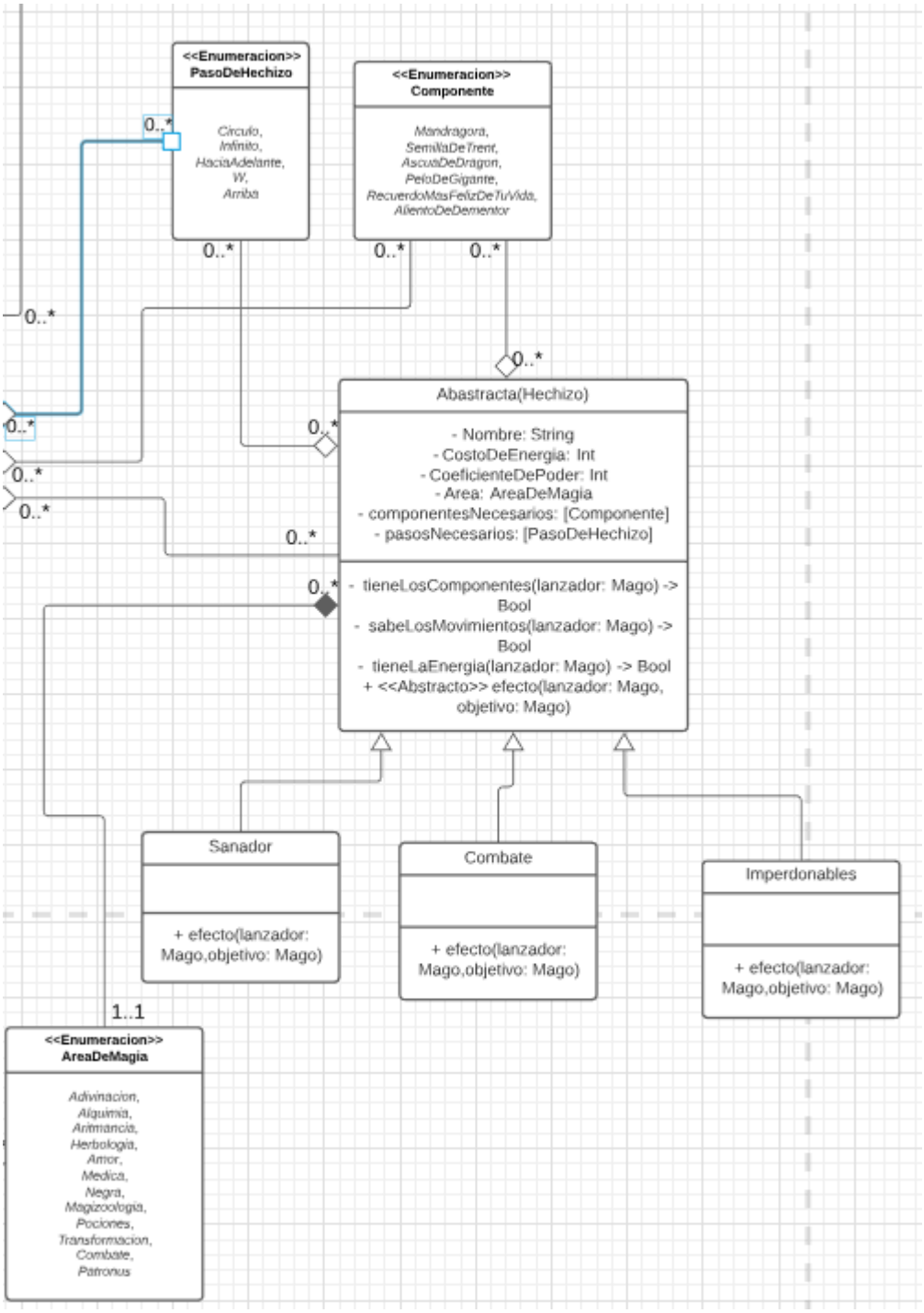
El mago, también hace uso de una entidad llamada Varita, la cual podría haber sido un simple entero, pero se modelo a propósito como un objeto aparte con el objetivo de hacerlo un poco más escalable a futuro, ya que es fácil añadir funcionalidad o atributos a la clase varita con el fin de modificar el funcionamiento en el futuro.

La entidad mago, también tiene una lista de magos llamados seguidores, un mago puede seguir a otro bajo ciertas condiciones detalladas en los requerimientos del proyecto, para eso se hace uso de los metodos, esFamoso() y agregarSeguidor(), esSeguible() y seguir()

Existen 2 subclases asociadas a la entidad Mago, que son Profesor y Alumno, el profesor tiene áreas de magia dominadas y una relación de agregación la entidad alumno, además de un método para poder enseñar Hechizos, la entidad Mago no se modeló como clase abstracta ya que se puede dar el caso de un Mago que no sea ni profesor ni alumno, por ejemplo, un Mago renegado, que también tendría el atributo casa sin asignar.

El mago también posee comportamientos asociado a los hechizos, además listas de una entidad llamada Componente y de otra entidad llamada Paso de Hechizo, pero serán explicadas en la siguiente sección.

Hechizo


<<Enumeracion>>
PasoDeHechizo
Circulo,
Infinito,
HaciaAdelante,
W,
Arriba
<<Enumeracion>>
Componente
Mandragora,
SemillaDeTrent,
AscuasDeDragon,
PeloDeGigante,
RecuerdoMasFelizDeTuVida,
AlientoDeDementor
Abastracta(Hechizo)
- Nombre: String
- CostoDeEnergia: Int
- CoeficienteDePoder: Int
- Area: AreaDeMagia
- componentesNecesarios: [Componente]
- pasosNecesarios: [PasoDeHechizo]
- tieneLosComponentes(lanzador: Mago) -> Bool
- sabeLosMovimientos(lanzador: Mago) -> Bool
- tieneLaEnergia(lanzador: Mago) -> Bool
+ <<Abstracto>> efecto(lanzador: Mago, objetivo: Mago)
Sanador
+ efecto(lanzador: Mago,objetivo: Mago)
Combate
+ efecto(lanzador: Mago,objetivo: Mago)
Imperdonables
+ efecto(lanzador: Mago,objetivo: Mago)
<<Enumeracion>>
AreaDeMagia
Adivinacion,
Alquimia,
Aritmancia,
Herbologia,
Amor,
Medica,
Negra,
Magizoologia,
Pociones,
Transformacion,
Combate,
Patronus
0..*
1..1

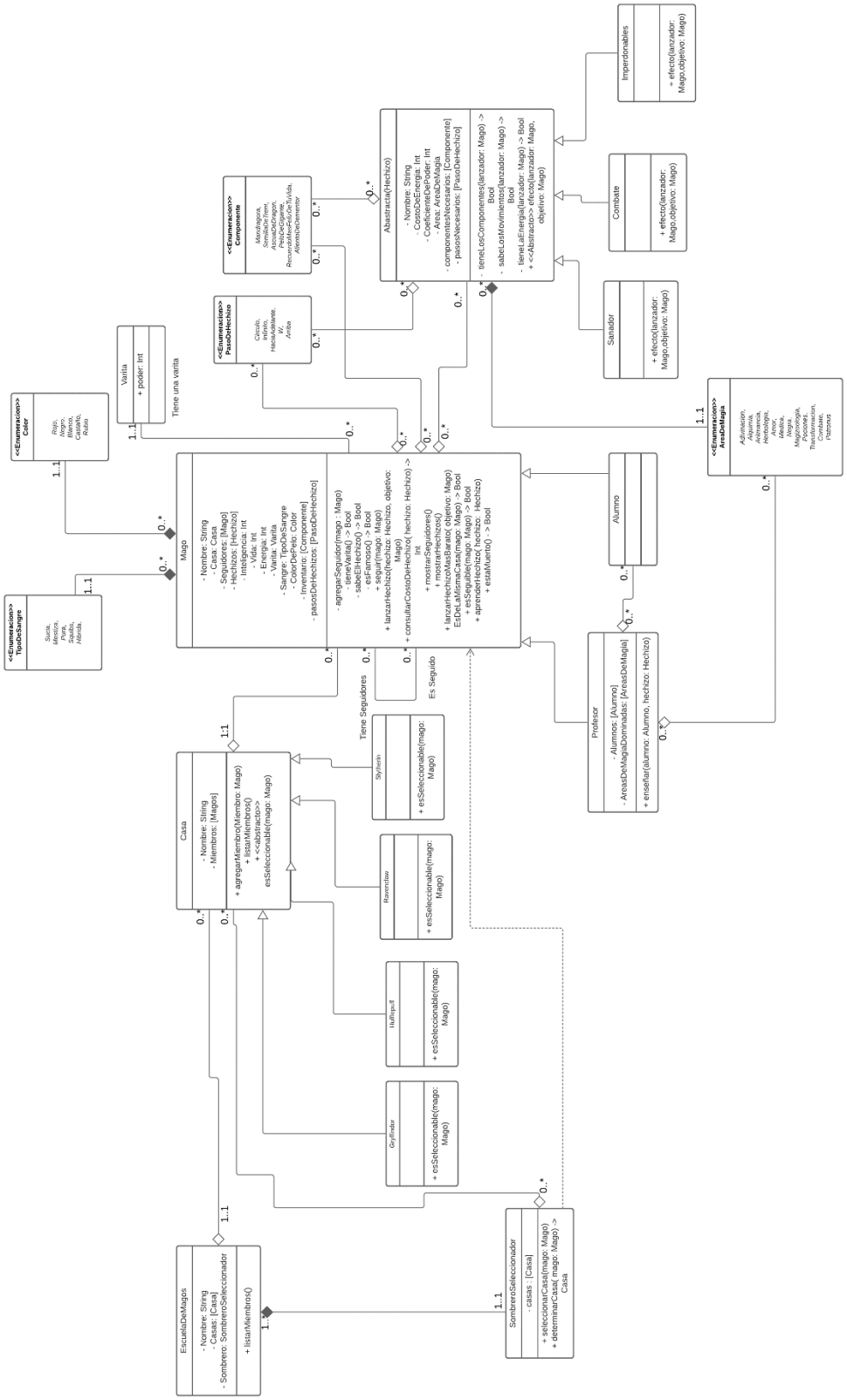
La sección de Hechizos se centra en los tipos de hechizos modelados y las entidades que los componen, para esto se hizo uso de dos enumeraciones llamadas Componente y Paso de Hechizo, que también son parte de la entidad Mago, estas enumeraciones son parte de los requisitos para poder lanzar un hechizo y son verificadas al momento de intentar lanzarlo, por los métodos tieneLosComponentes() y sabeLosMovimientos(), además de un tercer requisito que es verificar la energía necesaria a través del método tieneLaEnergia().

Se tomó la decisión de que es responsabilidad de la entidad Mago el hecho de lanzar el hechizo, pero el hechizo debe aplicar el efecto correspondiente, para eso se implementó un método abstracto efecto() que recibe como parámetro un mago lanzador y un mago objetivo, este método se aplica de manera polimórfica a través tres subclases llamadas HechizoSanador, HechizoDeCombate y HechizoImperdonable, cada una de ellas tiene su forma de calcular su efecto en base a las características del mago lanzador y se encargan de modificar el resultado en el mago objetivo.

Se modeló la clase Hechizo como una clase abstracta para poder exigir que todos los hechizos deban pertenecer sin excepciones a una subclase para poder ser instanciados.

Por último, cada hechizo tiene una Área de magia especifica la cual es inicializada al momento de crear el hechizo.

A continuación, se presenta una imagen del diagrama de clases completo de manera apaisada, que también será adjuntado con la entrega de este informe en formato PDF y PNG.


Imperdonables
+ efecto(lanzador: Mago,objetivo: Mago)
Abastracta(Hechizo)
- Nombre: String
- CostoDeEnergia: Int
- CostoDeHechizo: Int
- Area: AreaDeMagia
- componentesNecesarios: [Componente]
- pasosNecesarios: [PasoDeHechizo]
- tieneLosComponentes(lanzador: Mago) -> Bool
- sabeLosMovimientos(lanzador: Mago) -> Bool
- tieneLaEnergia(lanzador: Mago) -> Bool
+ <<Abstracto>> efecto(lanzador: Mago, objetivo: Mago)
Combate
+ efecto(lanzador: Mago,objetivo: Mago)
Sanador
+ efecto(lanzador: Mago,objetivo: Mago)
<<Enumeracion>> Componente
Varita
Caldero, Libro, Hierba, Piedra, Ala, Ambar
<<Enumeracion>> PasoDeHechizo
Varita
+ poder: Int
Tiene una varita
<<Enumeracion>> Color
Rojo, Negro, Blanco, Castaño, Rubio
<<Enumeracion>> AreaDeMagia
Adivinacion, Alquimia, Astronomia, Herbologia, Amor, Historia, Vuelo, Magizoologia, Pociones, Transformacion, Combate, Defensa
<<Enumeracion>> TipoDeSangre
Sucia, Mestiza, Pura, Squib, Hibrida
Mago
- Nombre: String
- Casa: Casa
- Seguidores: [Mago]
- Hechizos: [Hechizo]
- Inteligencia: Int
- Vida: Int
- Energia: Int
- Varita: Varita
- Sangre: TipoDeSangre
- ColorDePelo: Color
- Inventario: [Componente]
- pasosDeHechizos: [PasoDeHechizo]
- agregarSeguidor(mago : Mago)
- tieneVarita() -> Bool
- sabeElHechizo() -> Bool
- esFamoso() -> Bool
+ agregar(mago: Mago)
+ lanzarHechizo(hechizo: Hechizo, objetivo: Mago)
+ consultarCostoDeHechizo(hechizo: Hechizo) -> Int
+ mostrarSeguidores()
+ mostrarHechizos()
+ lanzarHechizoMasBarato(objetivo: Mago)
EsDeLaMismaCasa(mago: Mago) -> Bool
+ esSeguible(mago: Mago) -> Bool
+ aprenderHechizo(hechizo: Hechizo)
+ estaMuerto() -> Bool
Alumno
Profesor
- Alumnos: [Alumno]
- AreasDeMagiaDominadas: [AreasDeMagia]
+ enseñar(alumno: Alumno, hechizo: Hechizo)
Tiene Seguidores
Es Seguido
Casa
- Nombre: String
- Miembros: [Magos]
+ agregarMiembro(Miembro: Mago)
+ listarMiembros()
+ <<abstracto>>
esSeleccionable(mago: Mago)
Slytherin
+ esSeleccionable(mago: Mago)
Ravenclaw
+ esSeleccionable(mago: Mago)
Hufflepuff
+ esSeleccionable(mago: Mago)
Gryffindor
+ esSeleccionable(mago: Mago)
EscuelaDeMagos
- Nombre: String
- Casas: [Casa]
- Sombrero: SombreroSeleccionador
+ listarMiembros()
SombreroSeleccionador
- casas : [Casa]
+ seleccionarCasa(mago: Mago)
+ determinarCasa(mago: Mago) -> Casa

Decisiones de diseño e implementación tomadas

Si bien las decisiones de diseño e implementación fueron mencionadas y detalladas en el apartado anterior, se mencionarán las más importantes a método de resumen y conclusión a continuación:

1. En la sección de Instituciones:
    1. Se implementó la escuela de magos de manera tal que los pertenecientes a la escuela fueran parte de las casas y que las casas fueran parte de la escuela.
    2. Las casas aplican un método de selección de manera polimórfica, para agregar nuevas casas deberían ser agregadas como entidad que hereden de la clase Casa, esto da una mejor escalabilidad a nivel futuro.
    3. La forma de agregar alumnos a la escuela es mediante el sombrero seleccionador, que agrega a los magos a las casas.
2. En la sección de Magos:
    1. Se estableció que la responsabilidad de lanzar un hechizo sea de la clase Mago, pero que la de aplicar el efecto sea de la clase hechizo, asi como que la clase hechizo le provee a la clase mago, todos los comportamientos para verificar la posibilidad de poder lanzar el hechizo.
    2. Se usaron muchas enumeraciones para modelar entidades que funcionaban como listas para una mejor legibilidad del código.
3. En la sección de Hechizos:
    1. Se implementó de manera polimórfica el hecho de lanzar un hechizo y se estableció que la clase Hechizo sea abstracta para no permitir hechizos sin un efecto asociado a las subclases.

Problemas Encontrados

Se encontraron 2 problemas en el modelado temprano del proyecto que exigieron cambios:

En una primera instancia, se trató de modelar la entidad casa como una sola clase a la cual se le asignara el nombre y su condición de selección, se planeaba usar una función en constructor de la clase Casa que funcionara como un *callback* que se terminara disparando para devolver un valor booleano para saber si el mago en cuestión era seleccionable, esto no se pudo realizar de esta forma por el lenguaje de programación Java, por eso se optó por el polimorfismo ya que era una solución más adecuada al paradigma orientado a objetos.

Otro problema que surgió fue en un comienzo, la responsabilidad total de lanzar el hechizo, era de la clase hechizo y no del propio mago, lo que indujo cambios en el modelado para poder dejar de forma correcta las responsabilidades y no tener problemas futuros de escalabilidad y mantenimiento del código.

Ejecución del .jar

El .jar presentado adjunto al informe, presenta todos los requerimientos dados por las especificaciones del proyecto, como así también una simulación de una batalla entre estudiantes para que se verifique el correcto funcionamiento del mismo, se tiene una clase Test donde desde el main se llaman todos los requerimientos, cuyos métodos se llaman "*requerimientoN"* y están indicados con un comentario en el código y al ejecutarse.

Al ejecutar .jar se presentan todos los requerimientos y al último la batalla, el único requerimiento que se deja implícito es el de crear las 20 instancias de objetos, que está presente en todo el desarrollo de la clase Test.

Conclusión

El desafío más grande del proyecto era hacer un uso debido de la programación orientada a objetos, para llegar a un modelo satisfactorio en el cual cada clase tuviera su responsabilidad adecuada y que tuviera sentido, con eso en mente, el proyecto ha ayudado mucho al aprendizaje de modelar situaciones más complejas y sobre todo de la abstracción necesaria para este tipo de trabajos.