Trabajo Práctico: Iniciación al Ajedrez

Darién J. Ramírez
Facultad de Ingeniería y Ciencias Hídricas, ianshalaga@gmail.com

En este trabajo se explicará una posible manera de modelar un juego de ajedrez mediante un modelo de programación orientado a objetos. El resultado sería un juego de ajedrez básico donde podrán jugar dos jugadores, no se entrará en el desarrollo de una inteligencia artificial.

I. Conceptos de la programación orientada a objetos

La programación orientada a objetos es una metodología que descansa en el concepto de objeto para imponer la estructura modular de los programas. Permite comprender el dominio del problema a resolver, al intentar construir un modelo del mundo real que envuelve nuestro sistema. Es decir, la representación de este mundo mediante la identificación de los objetos que constituyen el vocabulario del dominio del problema, su organización y la representación de sus responsabilidades.

En la POO existen tres conceptos básicos: Objetos, Clases y Mensajes.

Un objeto es una abstracción conceptual del mundo real que se puede traducir a un lenguaje computacional o de programación OO. También tiene un estado, que permite informar lo que éste representa y su comportamiento, es decir "lo que él sabe hacer". Hablando en términos computacionales, la identidad del objeto se puede interpretar como la referencia. El estado del objeto es una lista de variables conocidas como sus atributos, cuyos valores representan el estado que caracteriza al objeto. Los objetos también se conocen como instancias. Los objetos encapsulan datos y operaciones (o métodos). Es una buena práctica de ingeniería encapsular el estado de un Objeto, ocultar la representación del estado del objeto a sus clientes externos. Ningún objeto puede modificar el estado de otro objeto. El estado interno de un objeto sólo puede modificarse mediante el envío de algún mensaje válido. El comportamiento expresa de que forma el objeto actúa y reacciona en términos de su cambio de estado y solicitud de servicio. Representa las actividades visibles exteriormente. Los métodos asociados a un objeto implementan el comportamiento del objeto.

Una clase contiene la descripción de las características comunes de todos los objetos que pertenecen a ella:
– la especificación del comportamiento.
– la definición de la estructura interna.
– la implementación de los métodos.
También se puede ver una clase como un molde, esquema o un patrón que define la forma de sus objetos. Un objeto es una instancia de una clase. Tiene un estado, un comportamiento y una identidad. Una instancia es un individuo de la clase.

¿Qué sucede cuando los objetos desean comunicarse entre sí? Un objeto recibe un estímulo externo de solicitud de un servicio que se traduce en la invocación de un método de éste objeto. Al ejecutarse el método, el objeto puede solicitar servicios de otros objetos, enviándoles mensajes que implican a su vez la invocación de sus métodos, y dentro de estos, nuevamente invocar servicios de otros objetos y así sucesivamente. Este envío de mensajes en cascada representa el comportamiento dinámico del modelo de objetos. En el envío de mensajes interactúan dos objetos: el emisor del mensaje y el receptor. La interface pública es el protocolo o conjunto de mensajes a los que puede responder el objeto (Punto de Vista del Usuario). La representación privada son: los datos necesarios para describir el estado y la implementación de los métodos o procedimientos (Punto de Vista del Implementador).

Este modelo cuenta de las siguientes características:

Abstracción: Denota las características esenciales de un objeto las cuales lo distinguen de todos los otros tipos de objetos. Provee una definición conceptual relativa a la perspectiva del observador. Separaremos el comportamiento de la implementación Es más importante saber qué se hace en lugar de cómo se hace.

Encapsulamiento: Es la propiedad que asegura que la información de un módulo esta oculta al mundo exterior. Es el proceso de ocultar todos los secretos de un módulo que no contribuyen a sus características esenciales. Es una técnica de diseño para descomponer sistemas en módulos. Ninguna parte de un sistema complejo debe depender de los detalles internos de otra.

Modularidad: Consiste en separar el sistema en bloques poco ligados entre sí: módulos. Es una especie de encapsulamiento de más alto nivel. Difícil pero muy importante en sistemas grandes. Suele aplicarse refinando el sistema en sucesivas iteraciones.

Ventajas de POO:
Fomenta la reutilización y extensión del código.
Permite crear sistemas más complejos.
Relacionar el sistema al mundo real.
Facilita la creación de programas visuales.
Agiliza el desarrollo de software.
Facilita el trabajo en equipo.
Facilita el mantenimiento del software.

Desventajas de POO:
Hay que ser muy cuidadosos en la creación de los objetos, ya que de ello dependerá el éxito de nuestro proyecto. Un error en estas primeras definiciones podría resultar catastrófico. Precisamente el secreto de esta técnica está en la correcta definición inicial de los objetos.

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenido. Estos

diagramas son los más comunes del modelado de sistemas orientados a objetos. Direccionan la vista de diseño estático del sistema. Un diagrama de clases esta compuesto por los siguientes elementos:
• Clase: atributos, métodos y visibilidad.
• Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Las relaciones existentes entre las distintas clases nos indican cómo se comunican los objetos de esas clases entre sí:
Los mensajes “navegan” por las relaciones existentes entre las distintas clases. Tipos de Relaciones:
- Asociación (conexión entre clases). La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre si. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.
- Generalización/especialización (relaciones de herencia). Indica que una subclase hereda los métodos y atributos especificados por una Super Clase, por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase (public y protected).
- Realización (se utiliza para implementar una interfaz).

Características de la herencia:
• Anulación o sustitución: cuando redefino un Método heredado en la subclase, se dice que estoy anulando o sustituyendo dicho método. Sería deseable una "herencia selectiva": seleccionar lo que se requiere heredar es la mejor forma de anulación.
• Sobrecarga: Propiedad que puede darse también sin herencia. Es designar varios elementos (identificadores) con el mismo nombre. No es anulación.
• Polimorfismo (sobrecarga con anulación) es la forma de invocar a distintos métodos utilizando el mismo elemento de programa. Es invocar métodos distintos con el mismo mensaje (ligadura en tiempo de ejecución). Para ello es necesaria una jerarquía de herencia: una clase base que contenga un método polimórfico, que es redefinido en las clases derivadas (no anulado). Se permite que los métodos de los hijos puedan ser invocados mediante un mensaje que se envía al padre. Este tipo de clase que se usa para implementar el polimorfismo se conoce como clase abstracta.

II. El Juego

Con los conceptos expuestos en el apartado anterior es posible comenzar a diseñar en juego. Para empezar se definirán las clases del sistema que representarán al juego de ajedrez. Como bien sabemos un juego de ajedrez consta de 32 piezas, un tablero de 8x8 casillas y un reloj para establecer el tiempo de las partidas. En primer lugar esos tres aspectos son los más importantes y por ende los que deben ser modelados correctamente.

Comencemos con el tablero, se creará una clase tablero la cual estará relacionada con una clase casilla, es decir, un objeto tablero contiene 64 objetos del tipo casilla y cada se identificará con un valor alfanumérico único que representará la coordenada, ejemplo, a1, a2, b7, h8. Cada casilla conoce su color y su posición en el arreglo, además para facilitar cuestiones de programación y de integridad de los movimientos, cada casilla sabe si se encuentra o no en el borde del tablero y de no ser así sabe a cuantos cuadros de los bordes se encuentra en las 4 direcciones.

El reloj es simplemente un par de relojes que se configuran en cuenta regresiva, los tiempos podrán ser configurables hasta un máximo de una hora por jugador. Los tiempos deberán ser seteados en minutos y estarán restringidos a solo poder configurar minutos de tiempo enteros, es decir, 3 minutos, 5 minutos, 10 minutos, pero no 7,5 minutos. También será posible configurar incrementos por jugada. Esta función será más utilizada en partidas de pocos minutos y dichos incrementos serán configurados en segundos. El tiempo de un reloj corre cuando un jugador debe efectuar su movida mientras que el del otro permanece detenido, al jugar, automáticamente se da la situación opuesta, de modo que la clase principal tendrá un vínculo con la clase reloj para poder determinar cuando se realiza una movida. A diferencia del ajedrez en la vida real donde un jugador puede seguir jugando a pesar de haberse terminado su tiempo mientras el contrincante no se percate de ello, aquí la partida se dará por finalizada automáticamente cuando uno de los dos tiempos llegue a cero. El resultado de la partida será determinado en base a las piezas de los jugadores, es decir, el jugador al que se le termina el tiempo puede perder o empatar en caso de que el contrincante no tenga los recursos necesarios para dar jaque mate. Por esto la clase principal deberá estar relacionada con la clase pieza y la clase tablero.

Como sabemos, existen seis tipos de piezas distintas en ajedrez, peones, caballos, alfiles, torres, damas y reyes, pero todos son piezas, entonces se creará una clase pieza la cual será una generalización de seis clases con nombres correspondientes con el nombre de cada pieza en singular (Peón, Caballo, Alfil, Torre, Dama, Rey). Esta clase contendrá un método relacionado con el movimiento de una pieza, que luego es cada una de las clases hijas será modificado por la propiedad de polimorfismo para adecuarlo a cada pieza en particular. Este método deberá establecer dada una casilla donde se encuentra una pieza, a cuales casillas tiene posibilidad de moverse y a cuales no. También contemplará si existe una pieza dentro de su camino de movimiento y esta es enemiga, la posibilidad de cambio. En caso de ser una pieza propia será una defensa que servirá para ayuda a los principiantes en el juego. Existirá un método de coloración donde al seleccionar una pieza para mover se mostrará las posibles casillas de movimiento en un color, las casilla de las pieza que amenaza en otro y finalmente las casillas de las piezas que defiende en otro. Esta opción podrá activada o desactivada de acuerdo a la necesidad del jugador.

La clase principal será la encargada de regular el funcionamiento total del sistema, allí estará el método de movimiento que irá cambiando la disposición de las piezas en el tablero y el registro del tiempo. También se puede agregar en ella un registro de los movimientos que se han realizado durante la partida para ser guardados y poder analizarlos luego.

A grandes rasgos con los elementos descriptos anteriormente se tiene definido el juego de ajedrez. Para agregar algunas opciones de personalización se pueden agregar tres clases asociadas cada una con las clases pieza, tablero y reloj respectivamente. Estás son un mini base de datos que contienen modificadores gráficos del sistema parmitiendo cambiar el aspecto de las piezas, del tablero o del reloj. Cabe aclarar que en primera instancia este juego está pensado para ser bidimensional de modo que tanto el tablero como las piezas y el reloj cambiarán de aspecto pero siempre en un marco bidimensional. Existen

a disposición gratuita muchos modelos de piezas que se podrán usar para implementar esta opción, con los tableros ocurre los mismo. Con los relojes es un poco más complicado ya que suele ser simplemente un reloj digital al costado del tablero, la idea aquí es que se pueda seleccionar por ejemplo, uno con agujas.

También se puede implementa un clase sonido vinculada con la principal encargada de emitir sonidos ante las acciones en el tablero. Habría un sonido para el movimiento de las piezas, uno para el cambio de piezas, uno para el jaque y uno de finalización para el jaque mate, siempre siendo lo más discretos y respetuosos posible, no se deberán incluir sonidos llamativos y molestos que irriten a los jugadores. Además estos sonidos podrás ser configurables, existen mucho sonidos de disposición libre en la web que se podrán usar para este cometido.

Con esto se obtendría un juego decente de ajedrez para disputar partidas entre conocidos, ya que de momento no se está contemplando el hecho de poder jugar a través de internet. El código del juego puede ser implementado en cualquier lenguaje de programación orientado a objetos pero por mi parte elegiría Java por su simple lectura y entendimiento, y además por ser un lenguaje orientado a objetos puro.