Universidad Simón Bolívar
Departamento de Computación
y Tecnología de la Información
Laboratorio de Lenguajes de Programación
CI-3661

PIEDRA, PAPEL, TIJERAS, LAGARTO O SPOCK

Realizado Por:
Germán Robayo 14-10924
Gabriel Gutiérrez 13-10625

1. INTRODUCCIÓN

Ruby es un lenguaje de programación multipropósito, dinámico, interpretado, orientado a objetos y reflexivo. Los tipos de lenguaje multipropósitos, como Ruby, no están diseñados para atacar un tipo de problemas en específico (Como podría verse con SQL y su relación con manejo de bases de datos) por lo que pueden ser usados de manera versátil para resolver diferentes asuntos e igual proporcionar facilidad de resolución en ellos.

2. INSTRUCCIONES DE JUEGO

El juego puede ejecutarse de dos maneras, por el terminal haciendo uso únicamente de este y sin interfaz gráfica , o haciendo uso de la interfaz provista por **Ruby Shoes**.

- Terminal :En caso de correrse por el terminal se debe correr " Ruby pptls-terminal.rb**"** .

  Se desplegará un menú que hará que el usuario navegue para configurar las estrategias y luego pueda jugar bien sea usando solo una jugada, o se puede usar el método rondas o alcanzar. El método alcanzar(<entero>), para poder jugar hasta alcanzar la puntuación pasada como parámetro. El método rondas(<Entero>), que juega la cantidad de rondas pasadas como parámetro.

- Interfaz : Para obtener la interfaz es necesario cargar con shoes el archivo **PPTLS.rb.** Esto se puede lograr mediante la interfaz de shoes o por el terminal con "./shoes PPTLS.rb" . Se recomienda agregar el flag --console para más detalles de las rondas.

  Una vez abierta la interfaz se preguntan los datos principales del usuario al igual que su estrategia, posteriormente se lleva a una vista donde se termina de especificar lo necesario de la estrategia y se da la opción de jugar por rondas o por puntos.

3- DETALLES DE IMPLEMENTACIÓN

Para implementar el juego se crearon 4 clases principales: Jugada, Estrategia, Partida, Tablero, y una serie de subclases de las mismas para poder hacer uso de la herencia que nos brinda la programación orientada a objetos. Aún cuando Jugada y Estrategia no son clases abstractas, tienen un uso de este estilo ya que nunca son instanciadas, solo sus subclases.

- Jugada : Clase que engloba las jugadas Piedra, Papel, Tijeras, Spock, Lagarto. Apartando los métodos requeridos, se sobrescribe el método hash de esta clase para redefinir cuando se consideraban iguales dos objetos de la clase Jugada, al igual que se cambió el método inspect y to_s para obtener un output más explicativo del estándar de ruby. Para obtener qué tipo de jugada gana a quien, se implementó una lista de aquellos contra los que cada jugada pierde, y luego se verifica si el oponente está en esa lista usando la reflexión de clases.
- Estrategia: Clase que engloba Manual, Copiar, Uniforme, Sesgada, Pensar. Estas clases se implementan de la siguiente manera:

  Entendamos como jugadas a cualquiera de los símbolos " :Piedra , :Papel, :Tijeras, :Piedra, :Spock "

  - Estrategia Manual : Manual.new() o Manual.new(<Nombre Jugador>)

    Esta estrategia no es más que mapear la entrada del usuario a la jugada indicada.

  - Estrategia Copiar: Copiar.new(<Jugada>) o Copiar.new(<Jugada>, <Nombre Jugador>)

    Para implementar copiar solo se recibe la última jugada del oponente y se devuelve esta misma como la nueva jugada.

  - Estrategia Uniforme : Uniforme.new([<Jugadas>]) o Unifome,new([<Jugadas>], <Nombre Jugador>)

    La clase uniforme recibe la lista de jugadas, y con el método para arreglos "uniq" se eliminan los duplicados.Luego se obtiene un valor random sobre la longitud de la lista, y se devuelve la jugada ubicada en esa posición.

  - Estrategia Sesgada: Sesgada.new({<Jugada> => <Probabilidad>}) o Sesgada.new({ <Jugada> => <Probabilidad>}, <Nombre Jugador>)

    Al igual que la estrategia uniforme, sesgada también usa random. Esta estrategia pone en una lista las jugadas ingresadas por el usuario,

repetidas tantas veces como su probabilidad lo indicaba, para luego aplicar el mismo principio que en la uniforme.

- Estrategia Pensar: Pensar.new() o Pensar.new(<Nombre Jugador>) .

  La estrategia pensar está implementada con un case sobre rangos, donde los rangos son los valores dadas por las repeticiones de las jugadas del oponente.

- Partida: Esta clase no tiene subclases, se instancia para ser la que contiene toda la lógica de historial de jugadas, puntuaciones, rondas, y los métodos necesarios para el juego de piedra, papel, tijeras, spock
- Tablero: El tablero es usado para recolectar toda la información del usuario, es el que tiene los métodos que interactúan con el usuario y hacen las instanciaciones de Partida, Estrategia, Jugadas.

4- CONCLUSIONES

Mediante el uso de programación orientada a objetos podemos obtener un código con mayor modularidad, que nos trae beneficios como mayor facilidad a la hora de encontrar errores. Por otro lado también nos permite reutilizar el código, ya que al tener subclases hay código que simplemente se hereda y no hace falta reescribir en las subclases, cosa que incluso hace que el código sea más mantenible y factorizable.

Con el uso de la programación orientada a objetos y de la reflexión se logran cosas como comparar las clases de los objetos, chequear si una clase es ancestro de otro, entre otras cosas. Esto abre la puerta a poder realizar chequeos de tipo con gran facilidad ya que podemos saber con qué tipo de dato estamos trabajando con gran facilidad.

5- CONSIDERACIONES SOBRE EL PROGRAMA

Las siguientes son consideraciones a tomar con respecto a la ejecución del juego por medio de la interfaz:

- Al usar la estrategia manual en alguno de los jugadores, las opciones de rondas, próximo y alcanzar no están funcionales. En esta modalidad hay que jugar partida por

partida. Al seleccionar la posición de manos se invoca la llamada de la próxima jugada del adversario, o en su defecto, se espera a que el otro usuario seleccione una opción.

Consideraciones generales:

Debido a la lógica de las estrategias, hay combinaciones que hacen que el juego se quede colgado, o entre en un ciclo infinito, por lo que hay que ser cuidadoso al realizar cosas como: Estrategia de ambos jugadores : Copiar con jugada inicial Piedra. Teniendo esta estrategia y corriendo Alcanzar(n) se entrara en un ciclo infinito ya que nunca se suman puntos.