Autoevaluación

Consideramos que cumplimos con todos los requerimientos solicitados en el sistema, habiendo tenido especial cuidado con la aplicación del GRASP Experto para la delegación de responsabilidades, así como en el correcto uso de una arquitectura M.V.C.

En cuanto a la división física del código, se realizó en cuatro paquetes, según de detallan a continuación:

1. controladores: Posee todas las clases controladoras de la parte de interfaz de usuario, así como las interfaces contra las que trabajan los mismos.
2. logica: Posee toda las clases del dominio, así como los sub sistemas y la fachada
3. utilidades: Contiene la clases ObligatorioException, la cual hereda de Exception y es utilizada para lanzar excepciones personalizadas, así como también contiene la clase interface necesarias para la implementación del patrón del Observador
4. vista: Posee todo lo relacionado con lo que el usuario visualiza, incluso el botón personalizado (Clase BotonFicha que hereda de JButton), así como la clase Inicio que contiene el main.

Autocalificación

(Acá va la tabla del Excel)

Datos de Prueba

(Acá va al tabla del Excel)

Nota: EN nuestra solución, la apuesta inicial de todas las partidas es de 100 por jugador, por lo que todos los que poseen saldo menor a 100, no podrán ingresar al juego.

Diagrama Conceptual del Dominio del Problema

(Pegar Diagrama)

Diagrama de Diseño

Notas:

Entendimos que el diagrama a presentar es completo, pero solamente debe incluir las clases correspondiente a la lógica.

Para que el diagrama sea más eficiente arrojando información más concreta, en el mismo se representan las relaciones entre las clases, detallando los atributos y métodos de las mismas en forma de lista.

No están los Enum (Ver si hay que ponerlos de alguna forma)

Ver si hay que aclarar de alguna forma los métodos que lanzan excepciones personalizadas (para identificarlos, actualmente tienen escrito el throws ObligatorioException. Si no es necesario, borrar eso de aquellos que lo tengan)

Averiguar si los nombres de los parámetros en el diagrama (En este caso, en lo que está a continuación) tienen que ser exactamente los mismos que en el código, o si acá pueden ser un poco más nemotécnicos

(Pegar diagrama)

Atributos y Métodos de las Clases

CLASE: Usuario

ATRIBUTOS

- nombre : String

- password : String

- nombreCompleto : String

METODOS

+ getNombre() : String

+ getPassword() : String

+ Usuario(n : String, p : String, nc : String)

+ getNombreCompleto() : String

+ equals(o : Object) : boolean

+ Usuario()

CLASE: Administrador

METODOS

+ Administrador(n : String, p : String, np : String)

CLASE: Ficha

ATRIBUTOS

- valorDerecha : int

- valorIzquierda : int

METODOS

- rotar() : void

+ getValorDerecha() : int

+ getValorIzquierda() : int

+ sePuedeUnir(lado : String, fichaDescartada : Ficha) : void throws ObligatorioException

+ Ficha(valorIzquierda : int, valorDerecha : int)

CLASE: Apuesta

ATRIBUTOS

- valor : double

- jugador : Jugador

METODOS

+ getValor() : double

+ getJugador() : Jugador

+ setValor(valor : double) : void

+ setJugador(jugador : Jugador) : void

+ Apuesta(valor : double)

CLASE: Movimiento

ATRIBUTOS

- ganador : Jugador

- fechaHora : Date

- pozoApuestas : double

- jugador : Jugador

- tablero : ArrayList<Ficha>

METODOS

+ getGanador() : Jugador

+ getFechaHora() : Date

+ getPozoApuestas() : double

+ getJugador() : Jugador

+ getTablero() : ArrayList<Ficha>

+ setGanador(ganador : Jugador) : void

+ Movimiento(ganador : Jugador, pozoApuestas : double, jugador : Jugador, tablero : ArrayList<Ficha>)

CLASE: Jugador

ATRIBUTOS

- saldo : double

- mano : ArrayList<Ficha>

- partidaJugando : Partida

METODOS

+ setSaldo(saldo : double) : void

+ getSaldo() : double

+ verificarSaldo(monto : double) : void throws ObligatorioException

+ agregarFicha(f : Ficha) : void

+ getFichas() : ArrayList<Ficha>

+ quitarApuesta(valor : double) : void throws ObligatorioException

+ eliminarFicha(ficha : Ficha) : void

+ vaciarMano() : void

+ Jugador(n : String, p : String, np : String, s : double)

CLASE: Partida

ATRIBUTOS

- partidaActiva : boolean

- pozoApuestas : double

- jugadores : ArrayList<Jugador>

- turno : Jugador

- ganador : Jugador

- movimientos : ArrayList<Movimiento>

- ultimaApuesta : Apuesta

- libres : ArrayList<Ficha>

- tablero : ArrayList<Ficha>

- primera : Ficha

- ultima : Ficha

METODOS

+ jugadorAbandonando(jugador : Jugador) : void

+ getLibres() : ArrayList<Ficha>

+ getTurno() : Jugador

+ getTablero() : ArrayList<Ficha>

+ getGanador() : Jugador

+ getPozoApuestas() : double

+ getJugadores() : ArrayList<Jugador>

+ getUltimaApuesta() : Apuesta

+ getMovimientos() : ArrayList<Movimiento>

+ agregarJugador(jugador : Jugador) : void throws ObligatorioException

+ mover(jugador : Jugador, fichaTablero : Ficha, fichaDescartada : Ficha) : void throws ObligatorioException

- controlesAntesJugar(jugador : Jugador) : void throws ObligatorioException

+ robar(j : Jugador) : void throws ObligatorioException

+ apostar(apostador : Jugador, monto : double) : void throws ObligatorioException

+ confirmarApuesta(confirmación : boolean) : void throws ObligatorioException

+ primerJugada(fichaDescartada : Ficha) : void throws ObligatorioException

+ segundaJugada(fichaTablero : Ficha, fichaDescartada : Ficha) : void throws ObligatorioException

- unirFicha(fichaTablero : Ficha, fichaDescartada : Ficha) : void throws ObligatorioException

- restarMontoJugadoresSumarApuestaEnPartida() : void throws ObligatorioException

- repartirFichas() : void

- mezclarFichas() : void

- cambiarTurno() : void

- finalizarPartida(ganador : Jugador) : void

- verificarTurno(jugador : Jugador) : void throws ObligatorioException

- verificarSiSeDescartoTodas() : void

- verificarSiTieneMovimientos(j : Jugador) : boolean

- verificarApuesta(monto : double) : void throws ObligatorioException

- verificarUltimoEnApostar(apostador : Jugador) : void throws ObligatorioException

- sePuedeJugar() : void throws ObligatorioException

+ enEspera() : boolean

- agregarMovimiento() : void

- crearFichas() : void

+ Partida()

CLASE: SubSistemaUsuario

ATRIBUTOS

- administradores : ArrayList<Administrador>

- jugadores : ArrayList<Jugador>

- jugadoresLogueados : ArrayList<Jugador>

METODOS

+ loginJugador(n : String, p : String) : Jugador throws ObligatorioException

+ loginAdministrador(n : String, p : String) : Administrador throws ObligatorioException

+ logoutJugador(jug : Jugador) : void

- boolean estaLogueado(n : String) : boolean

- cargarUsuarios() : void

+ SubSistemaUsuario()

CLASE: SubSistemaPartida

ATRIBUTOS

- partidas : ArrayList<Partida>

- proximaPartida : Partida

METODOS

+ partidaParaJugar() : Partida

+ partidasFinalizadas() : void throws ObligatorioException

+ getPartidas() : ArrayList<Partida>

+ agregarJugador(j : Jugador) : void throws ObligatorioException

+ SubSistemaPartida()

CLASE: Sistema

ATRIBUTOS

- ssu : SubSistemaUsuario

- ssp : SubSistemaPartida

- instancia : Sistema <<static>>

METODOS

+ getInstancia() : Sistema <<static>

+ loginJugador(nombre : String, pass : String) : Jugador throws ObligatorioException

+ loginAdministrador(nombre : String, pass : String) : Administrador throws ObligatorioException

+ partidasFinalizadas() : void throws ObligatorioException

+ getPartidas() : ArrayList<Partida>

+ agregarJugador(j : Jugador) : void throws ObligatorioException

+ getPartidaParaJugar() : Partida

+ logoutJugador(jug : Jugador) : void

- Sistema()