Programación O. Objetos
2016

Universidad Nacional de Lujan

Carrera: Licenciatura en sistemas de información

Profesor: Walter Panessi

Alumno: Juan I. Sacco Petras

Legajo: 129.908

Indice

Problema(TP Nº7)

1.Introducción Tiempo de juego: de 1 a 5 minutos.

Número de jugadores: sin límite. Los espectadores también pueden apostar.

Número de dados: dos. Elementos: dos cubiletes -
uno para el tirador y otro para el apostador- aunque también se puede jugar con uno solo.
Juego de azar interesante en el cual tanto el que tira como el que apuesta tienen las mismas oportunidades de ganar.

Cuando se juega en el casino, el empleado se lleva el 2,5% de las apuestas recogidas por el ganador en cada vuelta. Se lo llama también barbooth.

2.Objetivo: Lograr un tiro ganador (3-3, 5-5, 6-6, ó 6-5) y evitar los perdedores(1-2, 1-1, 2-2, ó 4-4).

3.Comienzo
Cada jugador tira los dados y el que saca la mayor suma será el tirador. El jugador a la derecha del tirador será el apostador. (En el Barbudi, tradicionalmente se juega en dirección contraria a las agujas del reloj.)

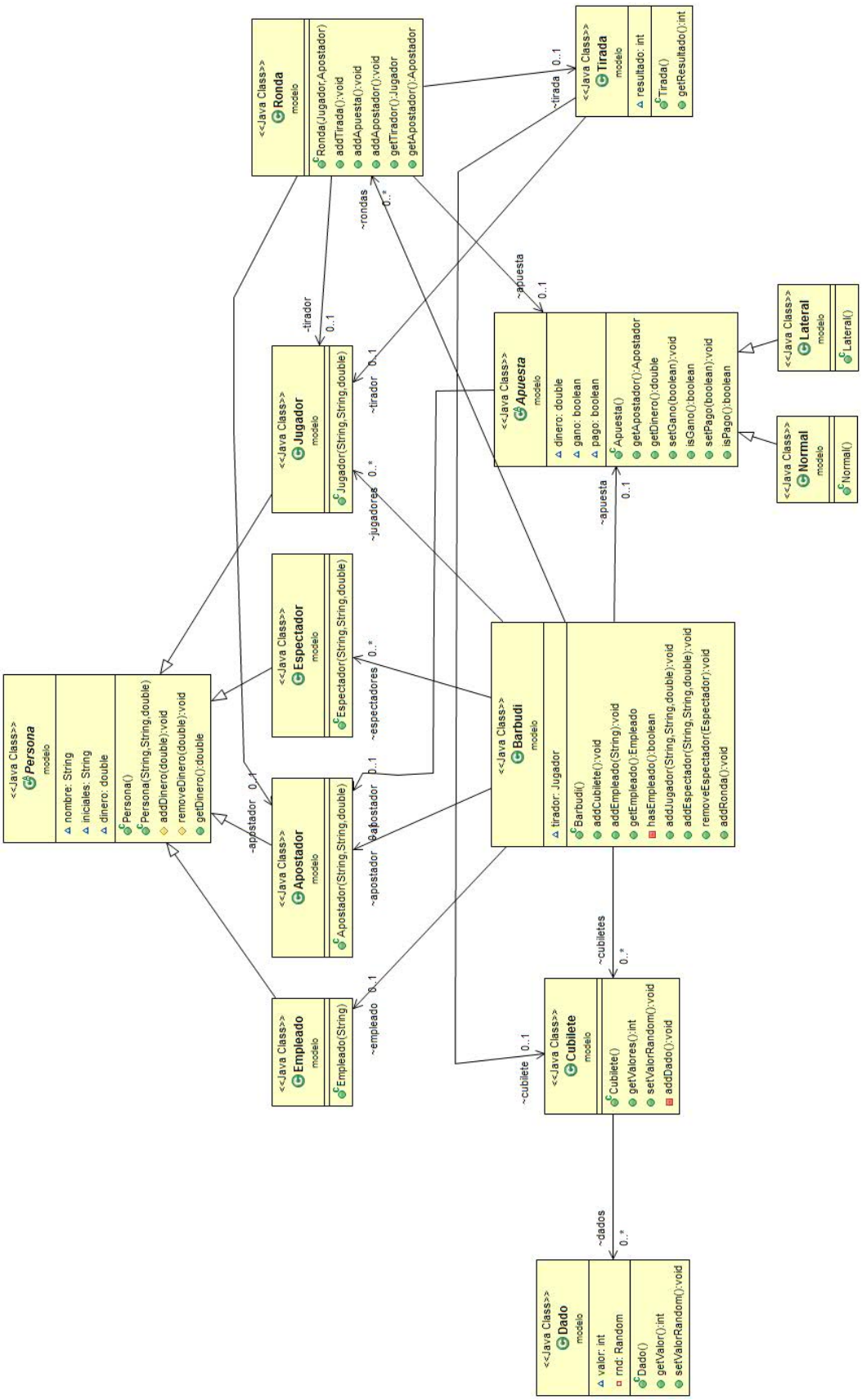
Apuesta primero el apostador y no el tirador. El apostador coloca su apuesta en el centro del área de juego. (El casino puede establecer un límite en el monto de ésta.) Luego el tirador cubre la apuesta total o parcialmente y deja que los demás jugadores cubran todo el resto (o parte de éste). No hay un orden determinado para hacer las apuestas, pero se puede fijar que el jugador a la derecha del apostador sea el primero en cubrir una parte de la apuesta. Se sigue apostando a la derecha hasta que la apuesta haya sido cubierta o hasta que nadie quiera apostar más. Se retira cualquier parte de la apuesta que haya quedado sin cubrir.
Si el tirador lo desea, puede rechazar la apuesta del apostador y pasar los dados hacia su derecha. De la misma manera, el apostador puede negarse a hacer apuestas y pasar la oportunidad de hacerlo al primer jugador de su derecha. Otros jugadores y apostadores pueden hacer apuestas laterales a favor o en contra del tirador.

4.Desarrollo:

Una vez hechas las apuestas, el tirador toma los dados y arroja. Si saca 3-3, 5-5, 6-6, ó 6-5, gana . Si saca 1-2, 1-1, 2-2, o 4-4, pierde.
Si no saca ninguna de estas combinaciones decisivas, pasa los dados al apostador. Éste tira y para ganar o perder, se aplican las mismas combinaciones que para el tirador.
Si no obtiene ninguna de estas combinaciones, le devuelve los dados al tirador.
El tirador y el apostador tiran alternadamente hasta que aparece una de dichas combinaciones. Entonces se fijan las apuestas y comienza una nueva vuelta.

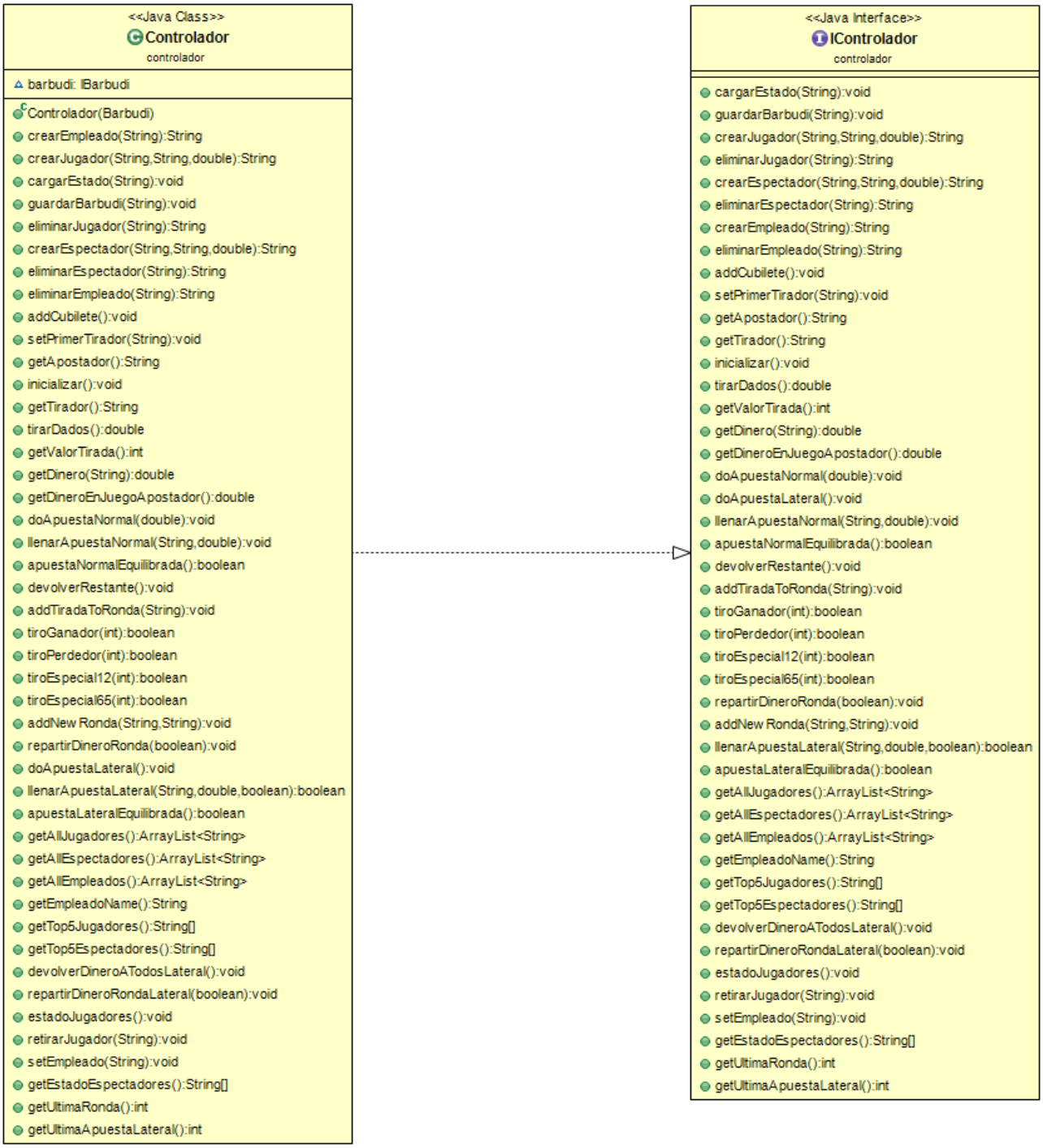
El jugador puede continuar como tirador mientras gane. Cuando el tirador pierde o cuando el apostador gana, este último pasa a ser tirador. Sin embargo, si el tirador pierde al sacar 1-2, o si el apostador gana al sacar 6-5, aquél puede retener los dados si lo desea

Primer UML – Antes de empezar

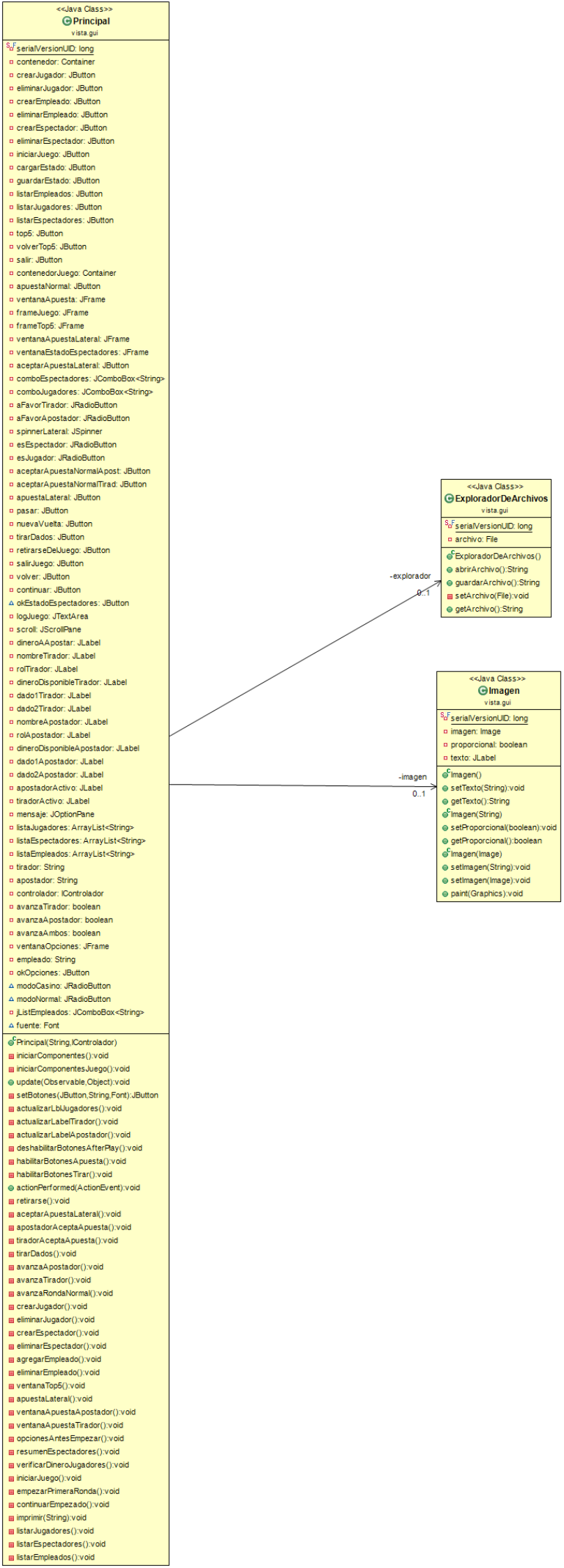


UML Final - Modelo

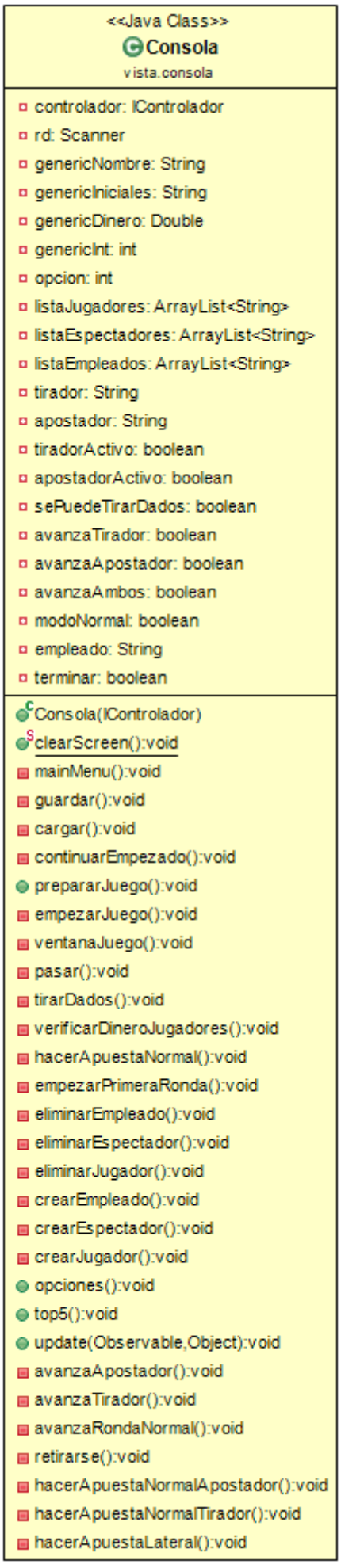
UML – Controlador



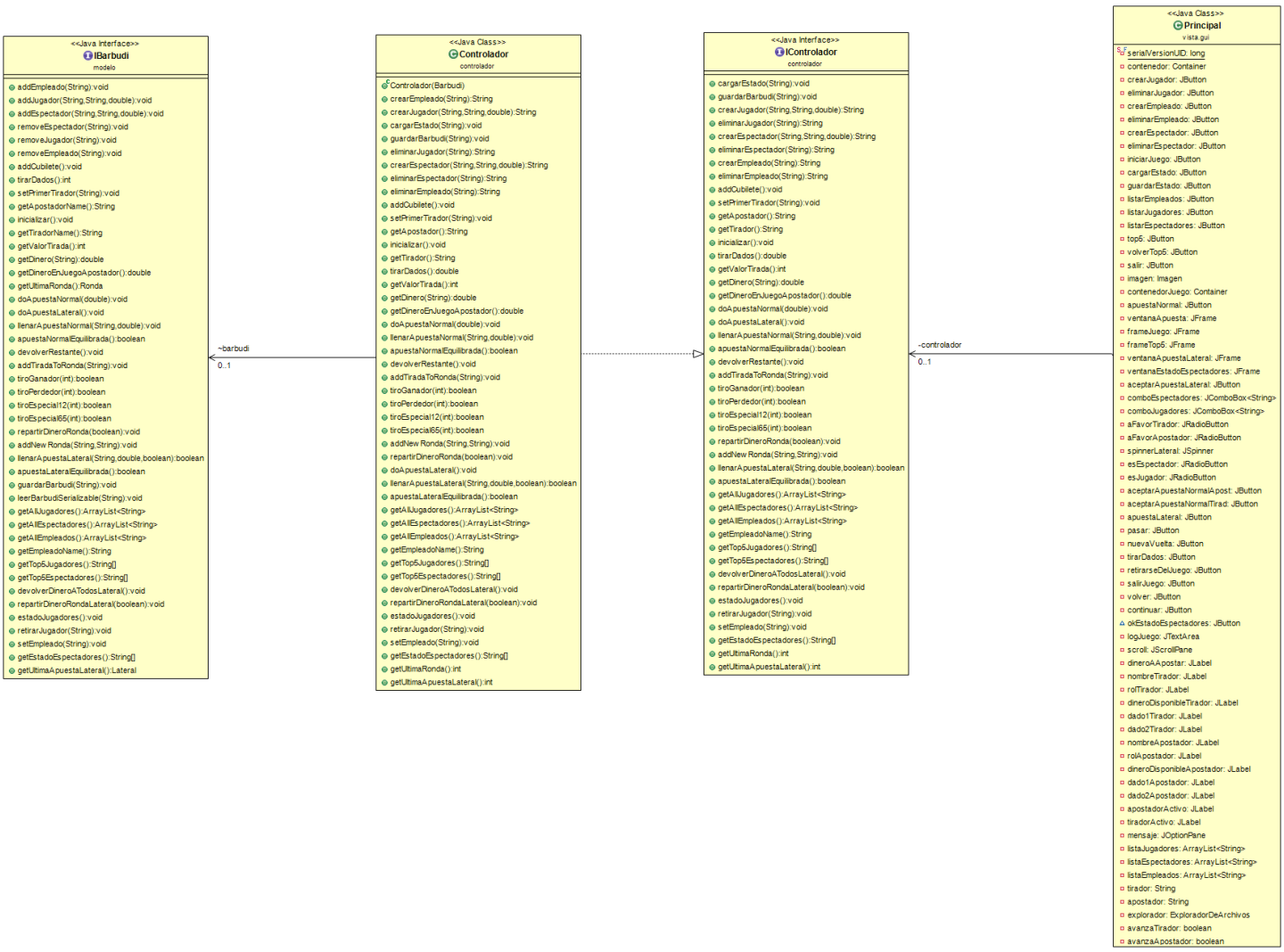
UML – Vista (GUI)



UML Vista – Consola



Relación entre M – V – C



Modificaciones: Respecto al primer UML.

Entidad	Primer UML	Ultimo UML
Persona	Se considero la posibilidad de que las entidades hijas funcionen como wrappers	Solo se le agregan 2 metodos getDineroEnJuego y setDineroEnJuego
Empleado	Representa al Empleado que esta presente en el juego	Esta entidad no es realmente necesaria, ya que el juego podria funcionar sin ella. En este ultimo modelo se utiliza para representar al empleado en “Modo Casino”. Si bien no tiene mucha funcionalidad, podría agregarse un atributo “Ganancia” y el casino así saber cuanto gano el empleado durante su trabajo.
Apostador	Esta entidad se utilizaría para discriminar a un tirador de un apostador.	En la actualidad fue removida del programa, ya que no es necesaria para distinguir un tirador de un apostador. Con la entidad Jugador es suficiente
Espectador	Representa a los espectadores que estan mirando el juego pero no jugando	Esta entidad no tuvo cambios significativos respecto un UML a otro
Jugador	En un principio, representaría a los jugadores que no eran “apostador”	Actualmente esta entidad representa a todos los jugadores que estan en la mesa del barbudi
Ronda	Guardaría toda la información referida a una ronda en Barbudi (Tirada,Apuesta,Apostador,Tirador)	Sigue la idea de guardar la información de la ronda, pero a diferencia del UML anterior, no guarda las apuestas, ya que, son guardadas en Barbudi
Apuesta	Una clase abstracta que representa las apuestas, sus hijos Normal y Lateral pensados como wrappers	Al final fue removida porque no tenia uso *practico* alguno,si bien teóricamente “Normal” y “Lateral” son Apuestas(cumple la condición teórica es-un) tienen comportamientos y atributos totalmente diferentes.
Normal	Wrapper de Apuesta	Normal tiene la capacidad de guardar a los jugadores a favor del tirador(incluido este), repartir las apuestas, devolver el dinero que sobra y auto-chequearse si la apuesta esta equilibrada
Lateral	Wrapper de apuesta	En la ultima versión se ve modificada tanto como “Normal”. Guarda las “Persona” a favor y en contra del apostador. Puede devolver el dinero a todos(en caso de que la ronda se cancele porque alguien “Paso”), verificarse si esta equilibrada, y

		reparte las apuestas a quien haya ganado
Tirada	Tiene el resultado que salio en el cubilete	Sabe que jugador y que cubilete la usa. Ademas, otorga el resultado de los dados(igual que antes).
Cubilete	Representa al cubilete que tiene 2 dados	No sufrió modificaciones
Dado	Representa al dado	No sufrió modificaciones.
Barbudi	Este es la clase principal del modelo, en este UML le falta un montón de métodos y algunos atributos. Tenga en cuenta que en este UML aun no se había escrito codigo	Esta clase sigue siendo el core de la aplicación. Se sumaron un montón de métodos entre ellos(Setters, Getters, Removers, Métodos generales, Persistencia y Reglas del juego). En atributos se agregaron ArrayList de Jugadores, Espectadores, Rondas, Cubiletes, Historial de Apuestas normales y laterales, y jugadores retirados(Esta ultima se implemento para cuando se quiere continuar con una partida). Entre otros atributos se agrego un booleano “modoNormal” que se utiliza para controlar que modo de juego se esta realizando

Conclusión

Este trabajo sumo muchos conocimientos entre ellos mucho del lenguaje Java: sus sintaxis, sus formas de ocultar/mostrar atributos y/o métodos (entre las diferentes clases implementadas) y las facilidades de poder implementar herencia de clases.

Ademas de lo aprendido en el lenguaje también sirvió para aprender de las diferentes formas de buscar una solución a un problema determinado mediante patrones genéricos, entre ellos, uno muy practico el cual fue implementado: el patrón Observer, en problemas como el que había que resolver, respetando el Modelo-Vista-Controlador, fue clave la implementación de este patrón ya que ante una modificación en el modelo (el cual estaba siendo observado) la vista era actualizada por si sola generando muchas facilidades y con una implementación muy simple.

En el modelo-vista-controlador se pudo comprobar ademas, que un Modelo y controlador puede servir para varias vistas.

Luego de muchas horas de trabajo, otras tantas de lectura y razonamiento sobre el tema se obtiene una conclusión muy positiva sobre este trabajo.