Diseño del Sistema

Ingeniería del Software

Nombre del grupo: The Moth

Nombre del proyecto: La Mosca

Nombre del documento: Documento de Diseño del Sistema

Integrantes: - Agustín Carranza
- Bruno García
- Emanuel Echazú

Fecha: 23/06/2020

Versión del documento: 1.0.0

Historial de Cambios

Versión	Fecha	Resumen del Cambio	Autores
0.1.0	16/06/2020	Creación del documento, agregación de diagramas UML.	Agustín Carranza
0.2.0	19/06/2020	Agregación de diagramas UML, ajustes en diseño de patrones, definición de pruebas unitarias, actualización de matriz de trazabilidad.	Agustín Carranza Bruno García Emanuel Echazú
1.0.0	23/06/2020	Corrección del diagrama de clases “Strategy”. Corrección de las pruebas unitarias. Se realizaron las correcciones en base a la devolución de Martín B.	Bruno García Emanuel Echazú

Índice

Consigna

Se debe presentar un documento de diseño en el cual se incluyan diagramas de paquetes, diagramas de clases y objetos, diagramas de secuencia y todo aquel diagrama que sirva para explicar el diseño del software a construir.

Se debe mostrar también la aplicación de patrones de diseño como el Singleton, Observer, Strategy, etcétera usando diagramas de clases e indicar por qué se utilizan y qué problemas se solucionan con ellos.

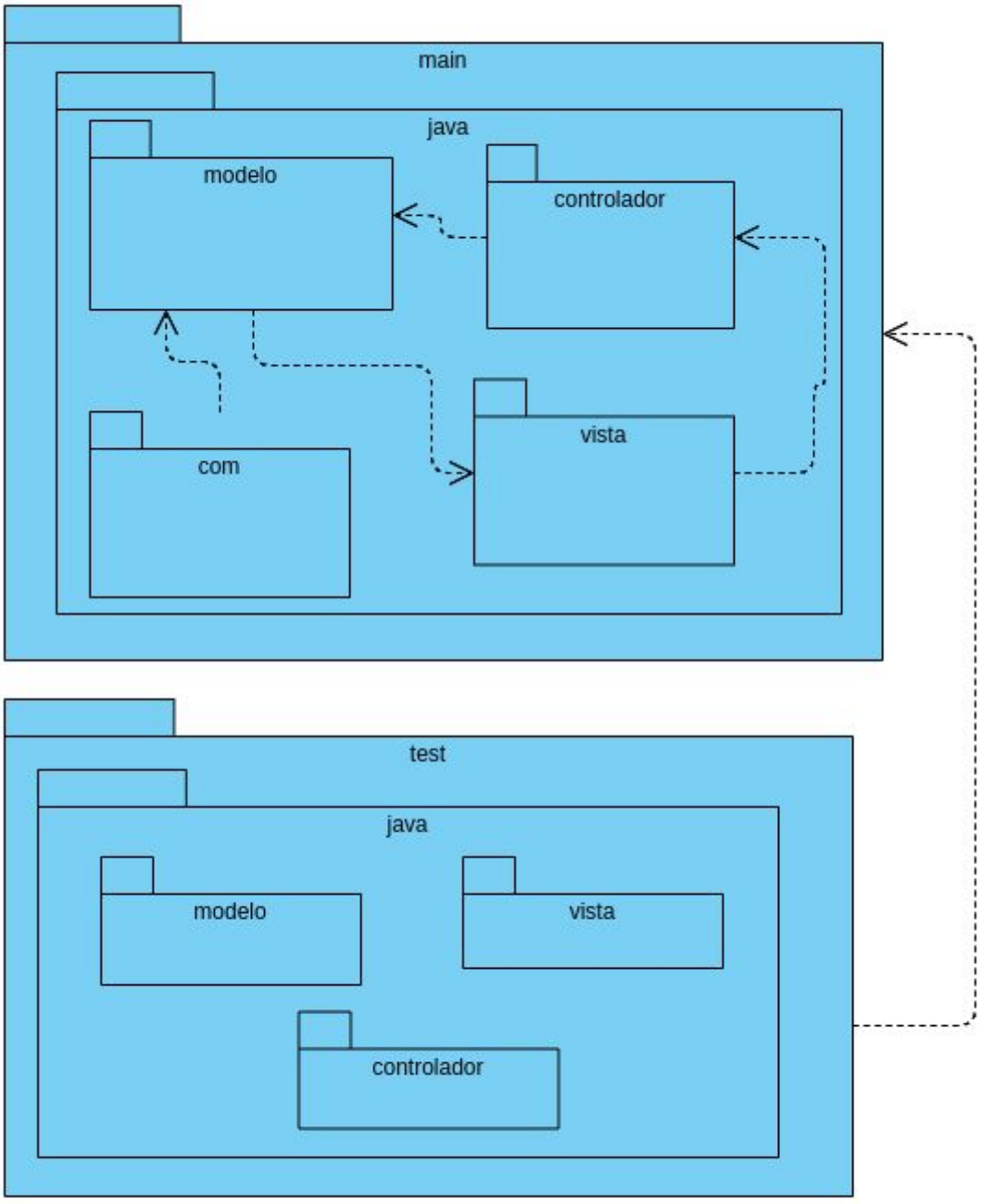
Por otro lado, se deben generar pruebas unitarias automáticas para el código, explicando cómo correrlas y verificar su estado. Actualizar la Matriz de trazabilidad para incluir módulos o clases y casos de pruebas unitarias.

Opcionalmente, se puede incluir cualquier otra herramienta de gestión de la calidad de software que haya sido usada FindBugs, CheckStyle, PMD, Sonar, etcétera. Mostrando ejemplos de su uso. En el caso de Sonar o herramientas similares, se debe agregar al Plan de Gestión de las Configuraciones la dirección y forma de acceso a la herramienta.

Diagramas UML

Diagramas de Paquetes

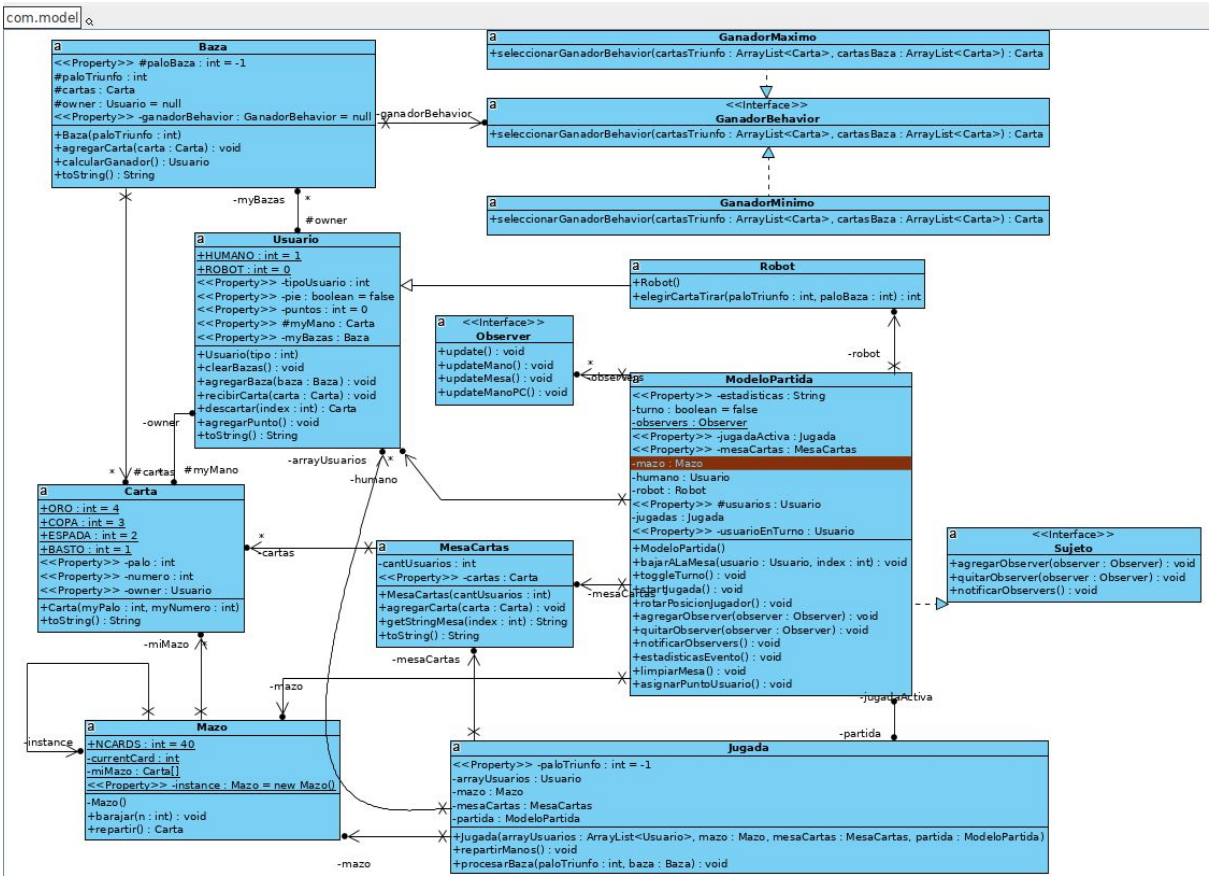
La distribución de paquetes refleja la arquitectura del patrón MVC.



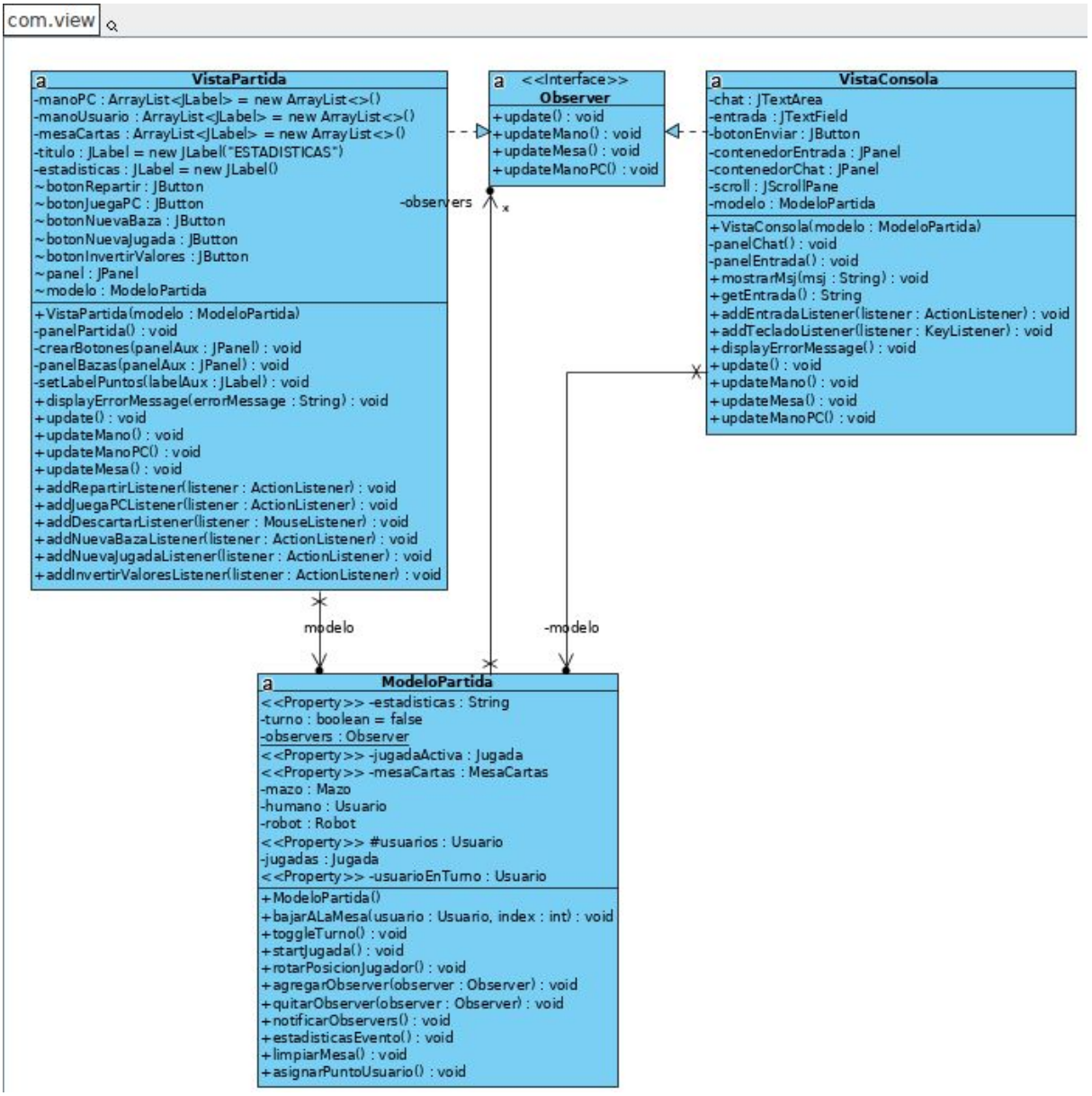
Paquetes

Diagramas de Clases

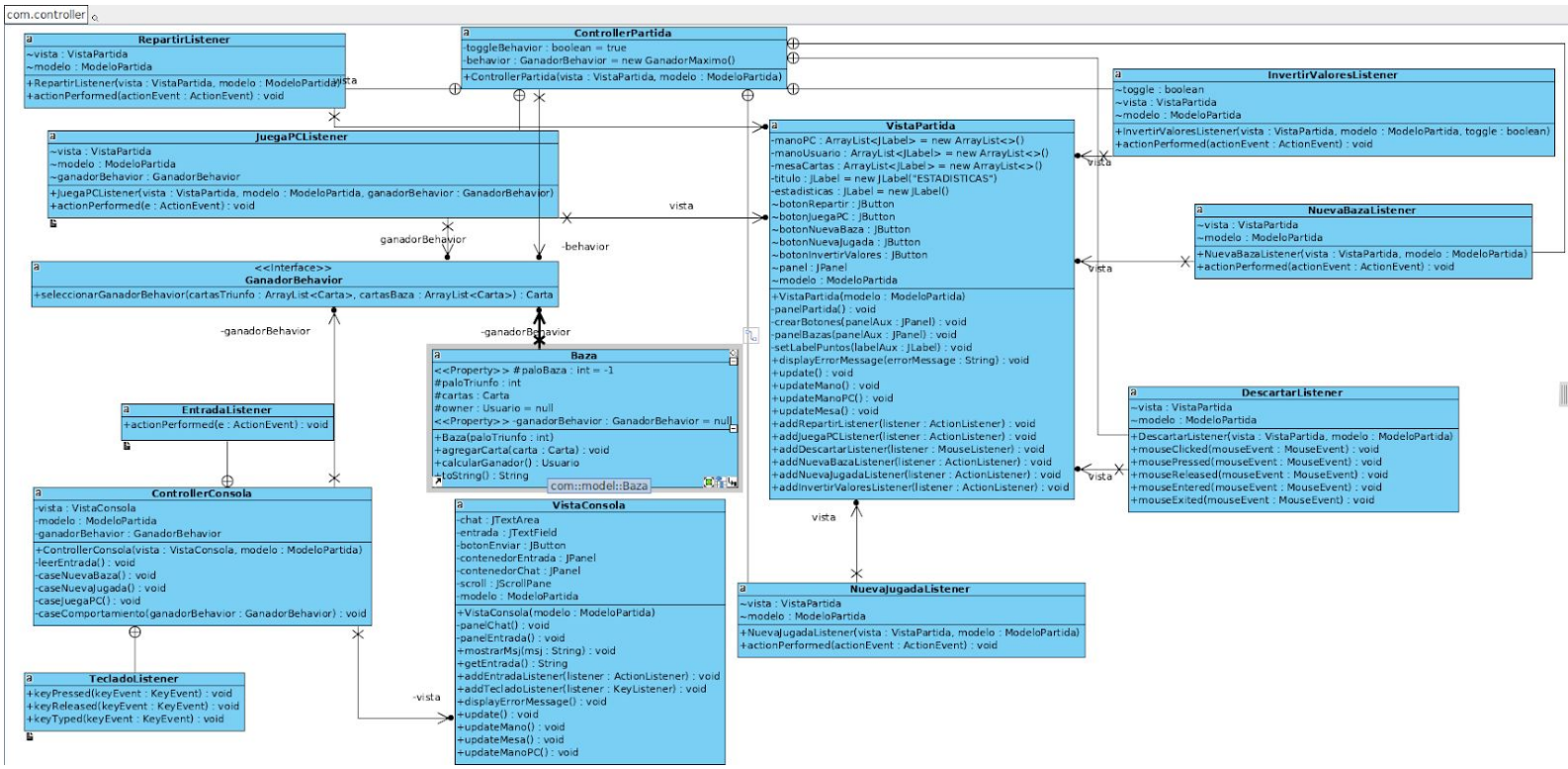
Se presentan diagramas de clase, separados por paquetes. Estos coinciden con los componentes del patrón MVC.



Clic en la imagen para ver más grande.

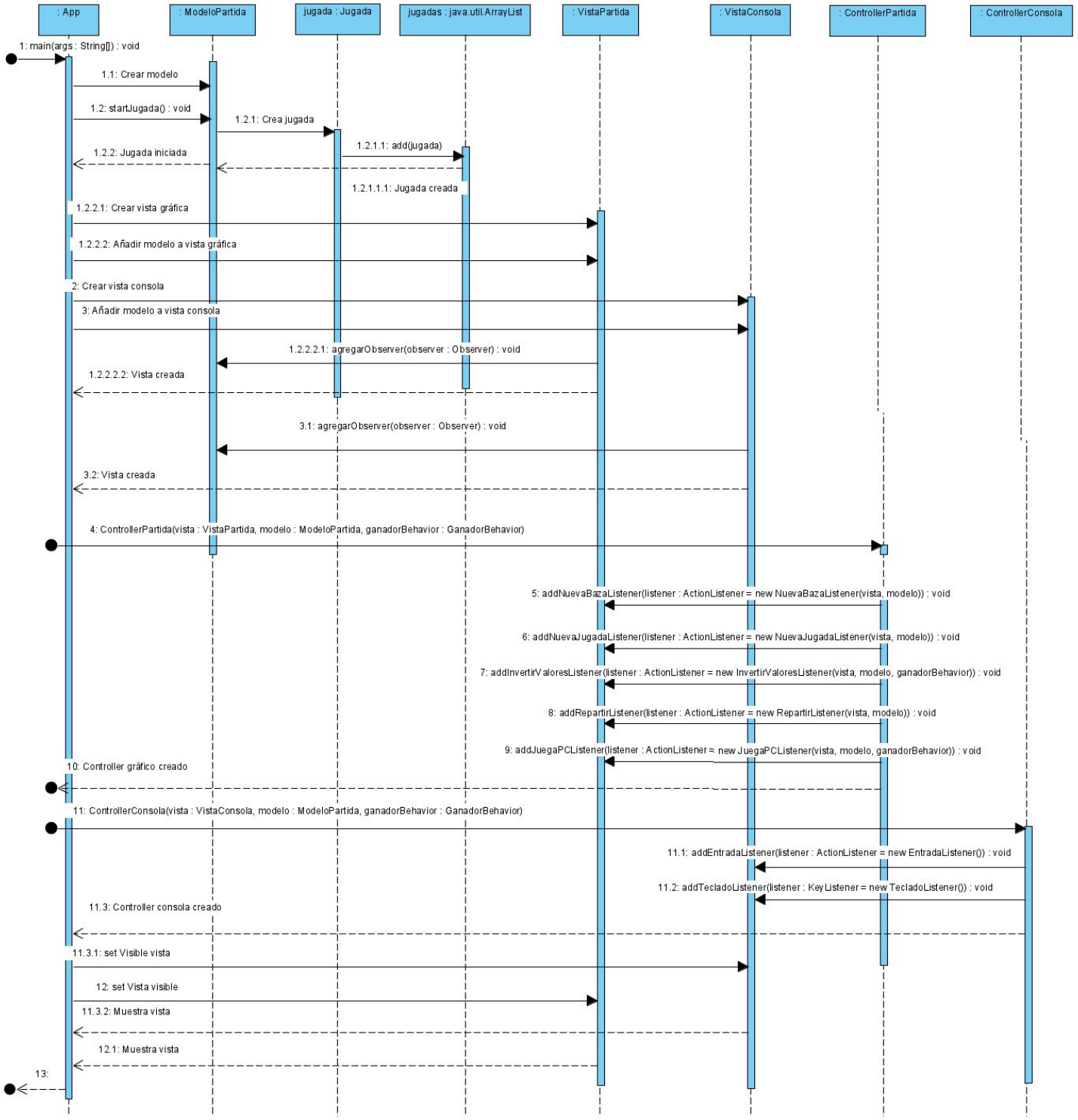


Clic en la imagen para ver más grande.



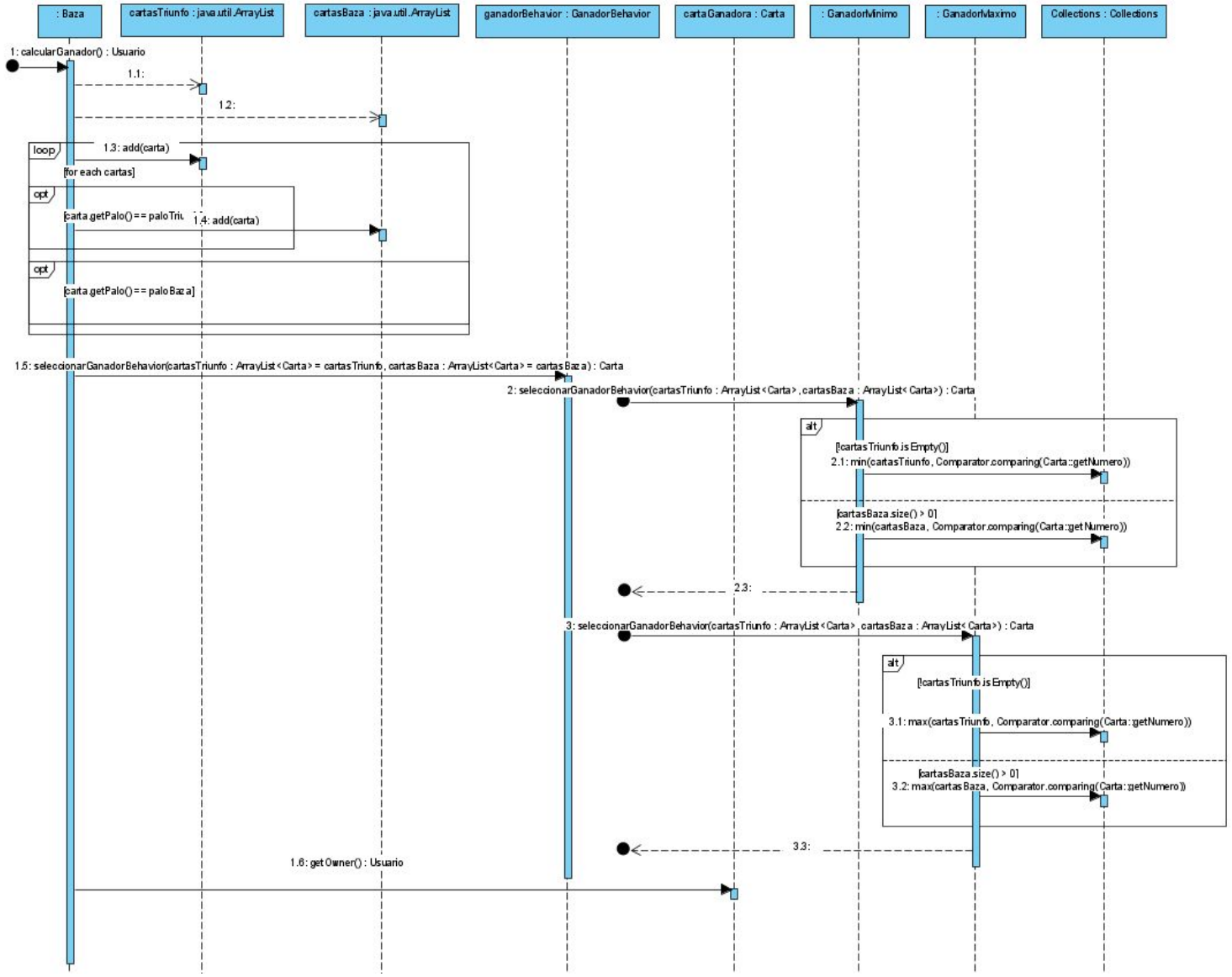
Clic en la imagen para ver más grande.

Diagramas de Secuencia



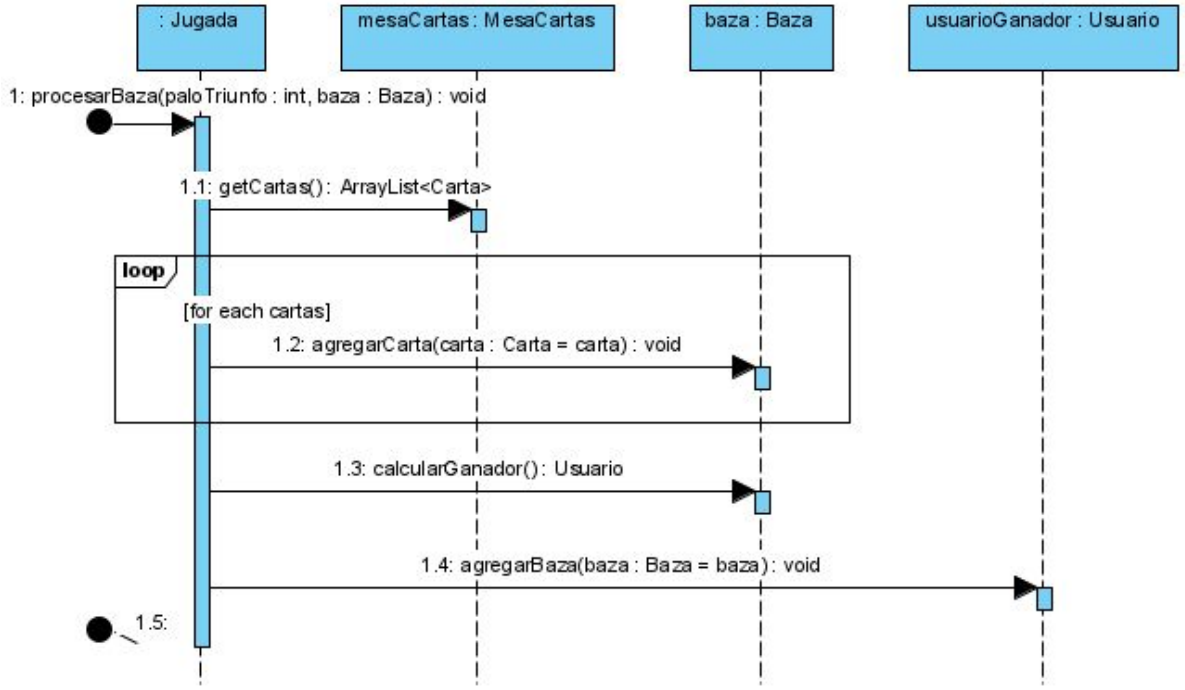
[Enlace de la imagen para ver más grande.]

Diagrama de secuencia: *calcularGanador*



Enlace de la imagen para ver más grande.

Diagrama de secuencia: ***procesarBaza***



[Enlace de la imagen para ver más grande.]

Diagrama de secuencia: ***startJugada***

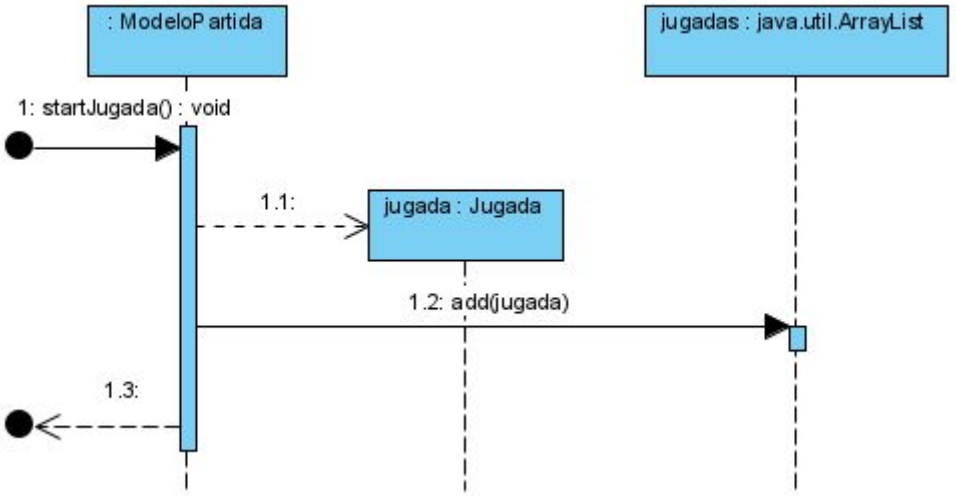


Diagrama de secuencia: ***bajarALaMesa***

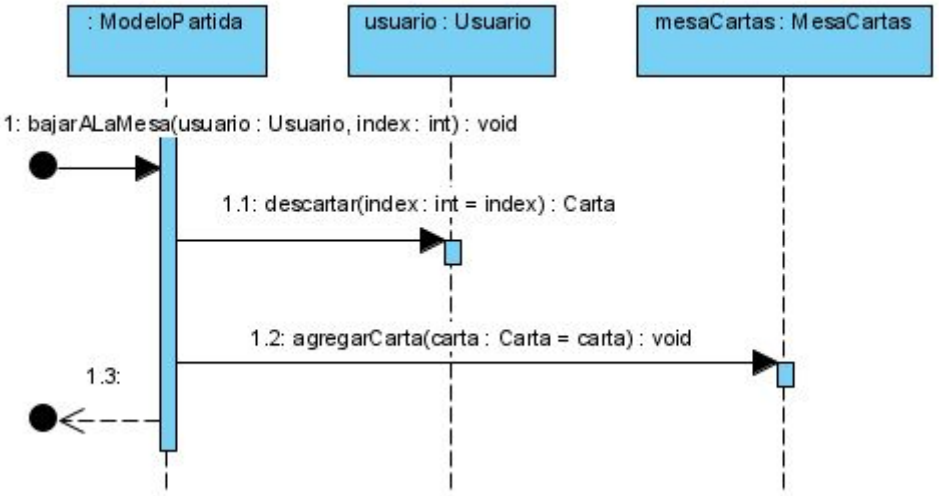
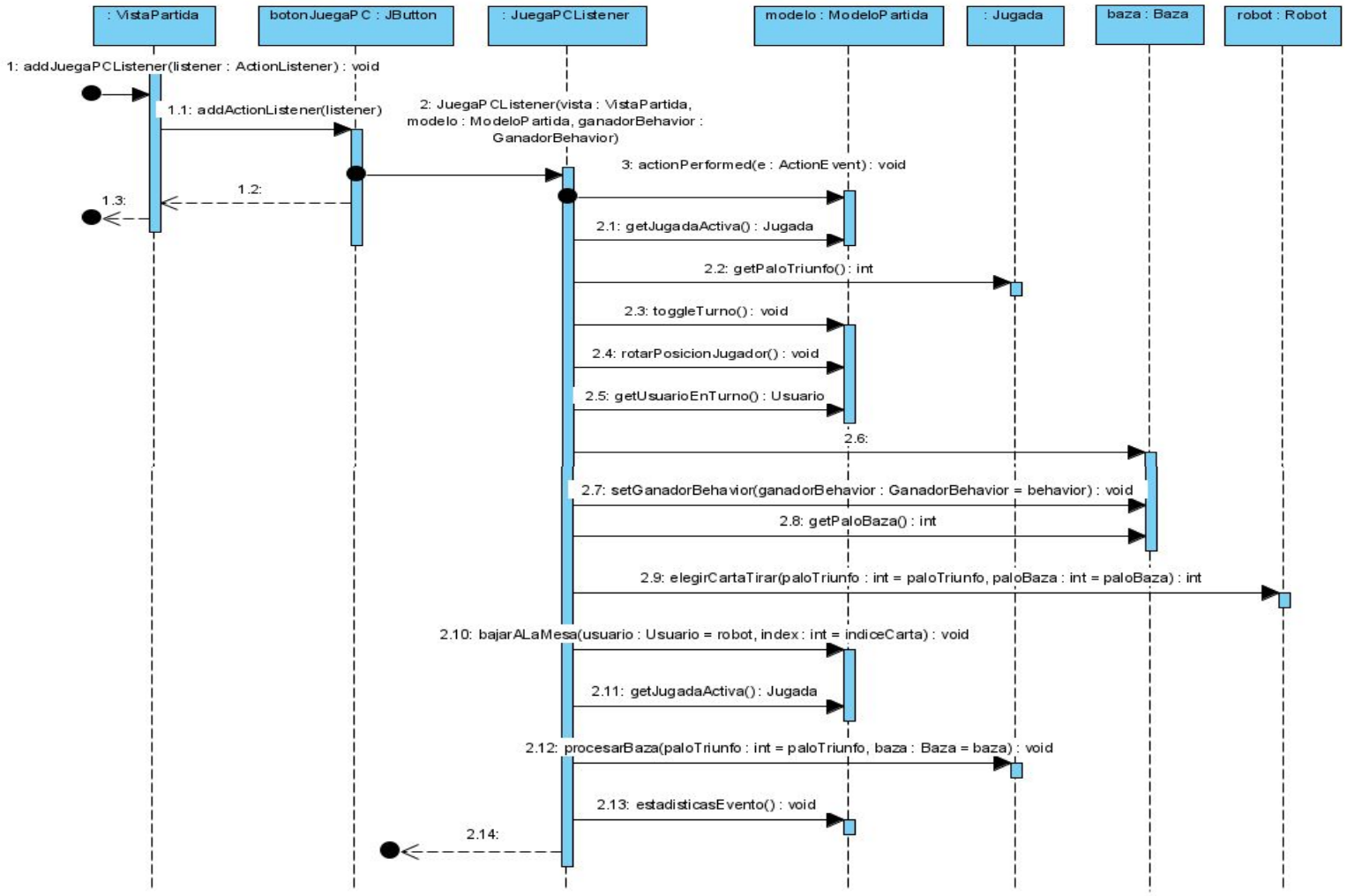


Diagrama de secuencia: ***addJuegaPCListener***

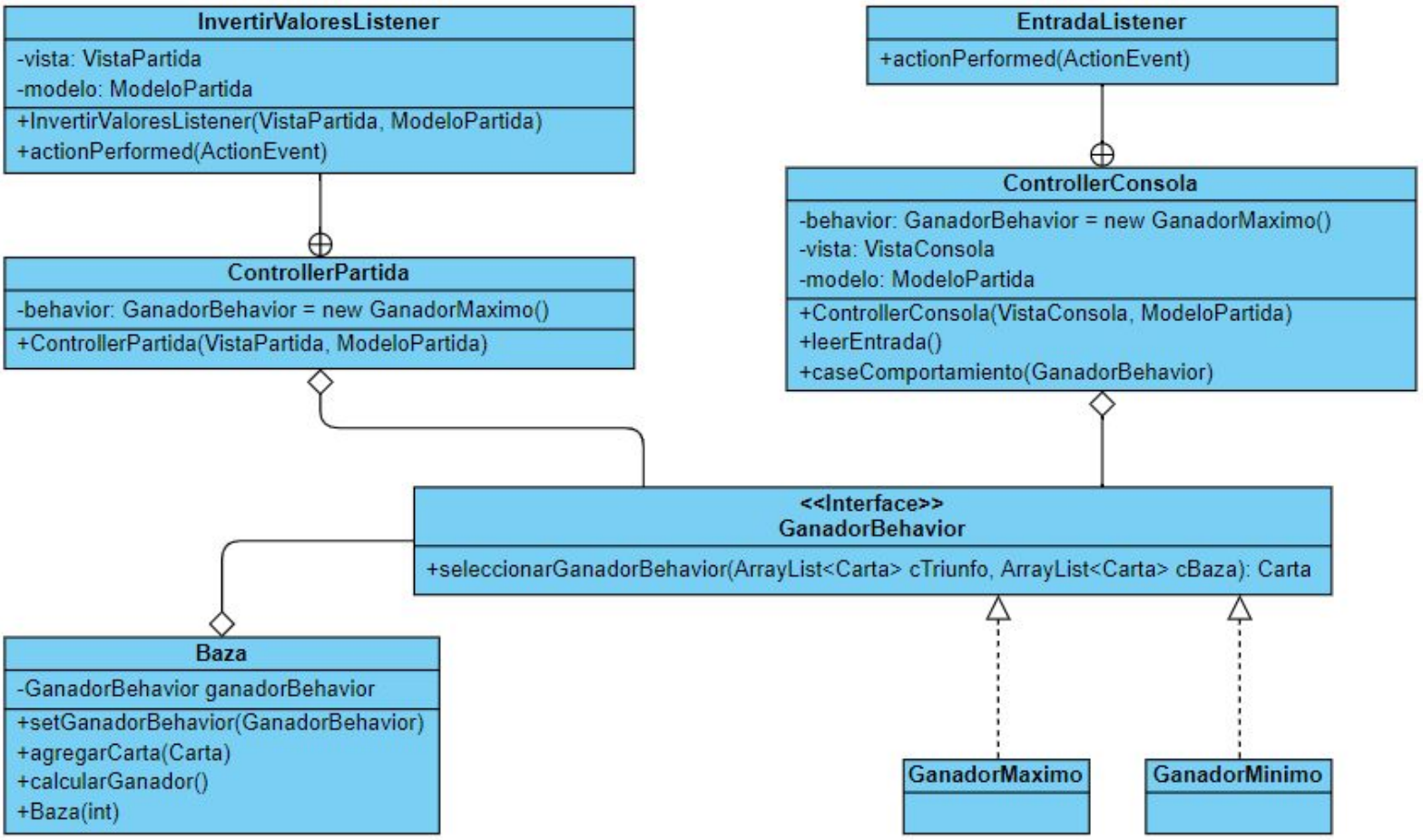


Enlace de la imagen para ver más grande.

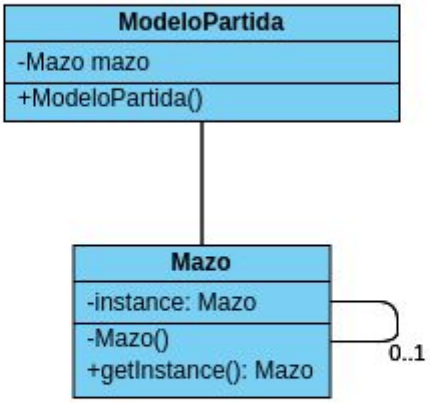
Patrones de Diseño

Diagramas de Clases

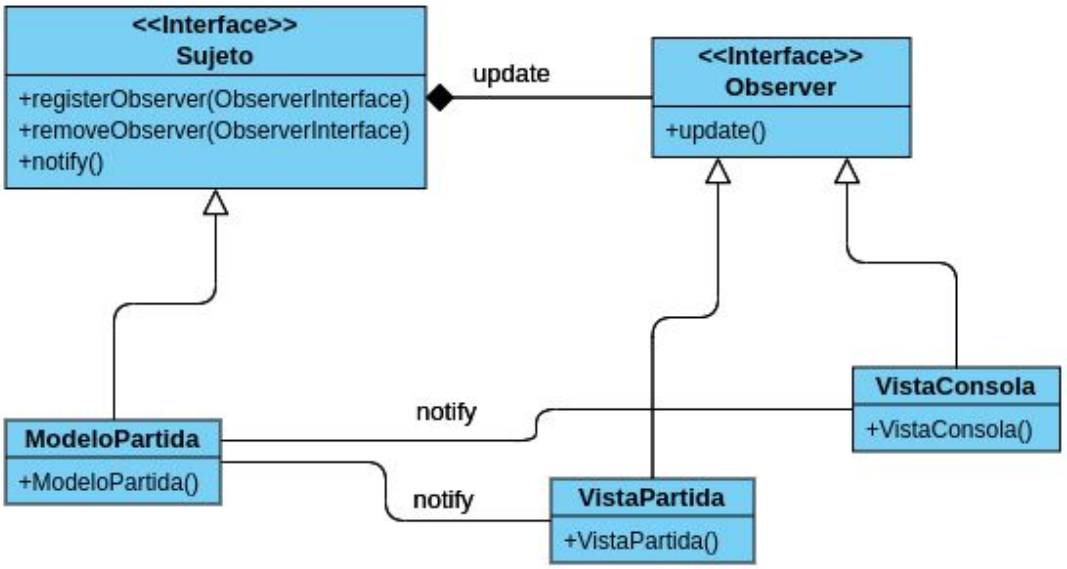
Strategy



Singleton



Observer



Justificación

El patrón singleton es apropiado para garantizar que una clase sólo tenga una instancia en toda la ejecución del software. En la vida real, en un juego de cartas, siempre hay un mazo único. Sería un error grave si en algún momento de la ejecución hay dos mazos instanciados.

El patrón strategy nos ayuda a encapsular los comportamientos de las bazas a la hora de decidir qué carta es la ganadora. Ocurre que en diferentes regiones del país, el juego de la mosca posee un orden descendente en el valor de las cartas. Esto quiere decir que las cartas numéricamente más bajas tienen valor más alto en las bazas del juego. En otros lugares ocurre exactamente lo contrario. Usar el patrón strategy permite elegir en tiempo de ejecución el comportamiento de los métodos encargados de calcular y asignar las bazas al ganador.

El patrón de diseño observer es una parte fundamental del patrón de arquitectura MVC. El hecho de tener las vistas desacopladas del modelo, y en el caso de este proyecto tener dos vistas disponibles, se aprovecha el patrón para informar a todas las vistas de un cambio de estado en el modelo. Esto facilita el desarrollo y la mantenibilidad a la hora de realizar cambios tanto en las vistas (como agregar o reemplazar alguna, por ejemplo), como en el modelo.

Pruebas Unitarias

Para correr el set de pruebas automáticas se deben correr los siguientes comandos en una consola:

```
$ git clone https://github.com/agucarranza/juego-maven
$ cd juego-maven
$ mvn test
```

Paralelamente, luego de realizar push a la rama master, la herramienta de integración continua correrá la pruebas unitarias y sólo aceptará el cambio si se pasan todas las pruebas.

Definiciones

- ➔ UT1: Verificar que el mazo no cree cartas repetidas.
- ➔ UT2: Verificar el turno de juego de cada usuario y luego realizar el cambio para pasar el turno al usuario contrario.
- ➔ UT3: Verificar luego de repartidas las cartas que cada usuario tenga en su mano 5 cartas.
- ➔ UT4: Verificar obtener distintos órdenes al momento de barajar el mazo, repitiendo el proceso en varias instancias.
- ➔ UT5: Verificar que al presionar con el teclado ENTER en la vista de la consola lea el texto ingresado en la caja de texto y esta quede vacía.
- ➔ UT6: Verificar que al presionar con el mouse el botón ENVIAR en la vista de la consola, ésta lea el texto ingresado en la caja de texto y quede vacía.
- ➔ UT7: Verificar que los componentes que conforman la vista figuren en el panel con los nombres asignados al ser creados.
- ➔ UT8: Verifica que la cantidad de jugadores al comenzar la partida sean 2.
- ➔ UT9: Verifica que el jugador robot, al tener una carta con el palo de la baza, tira esa.
- ➔ UT10: Verifica que una mano tenga como máximo 5 bazas.
- ➔ UT11: Verificar que el texto del nombre ingresado por el usuario no admite caracteres especiales.
- ➔ UT12: Verificar que existan 3 niveles de dificultad.

- ➔ UT13: Verificar que el jugador que tenga más puntos gane el contenido del plato una vez finalizada la partida.
- ➔ UT14: Verificar que la partida termina luego de transcurrir la cantidad de jugadas ingresadas al comienzo.
- ➔ UT15: Verificar al presionar el botón comienza primero que se seleccione el usuario mano. Verificar que el número elegido corresponda al usuario mano.
- ➔ UT16: Verificar que al presionar el botón comenzar de nuevo se cargue un nuevo panel y se cree una nueva jugada.

Mapeo de pruebas unitarias

Módulos	Casos de Pruebas Unitarias
Modelo	UT1
Modelo	UT2
Modelo	UT3
Modelo	UT4
Vista	UT5
Vista	UT6
Vista	UT7
Modelo	UT8
Modelo	UT9
Modelo	UT10
Modelo	UT11
Modelo	UT12
Modelo	UT13
Modelo	UT14
Modelo	UT15
Vista	UT16