UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE INGENIERÍA Y CIENCIAS
CC3002 - 1 Metodologías de Diseño y Programación

TAREA 3
Resumen

Daniela Campos
danielacamposfischer@gmail.com
https://github.com/DaniCampos/cc3002-tarea3

Profesor:
Alexandre Bergel

Fecha:
25 de Octubre del 2017

1 Diagramas UML

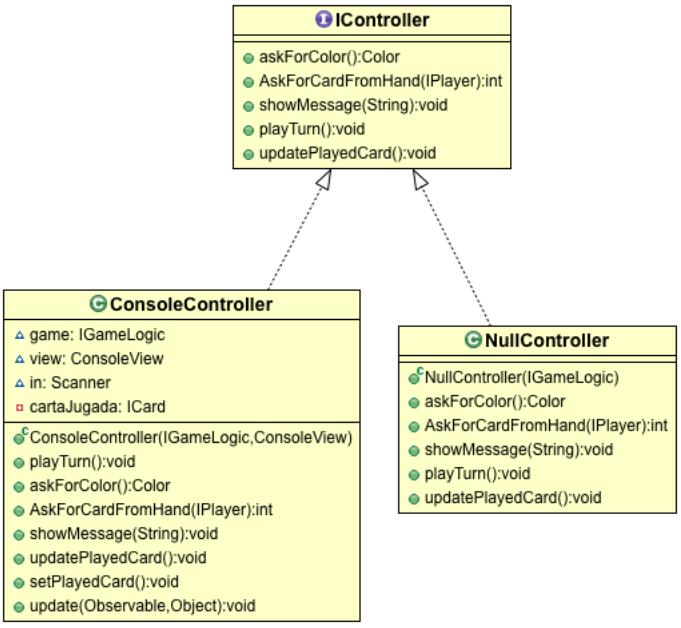

Figura 1: UML controller.package

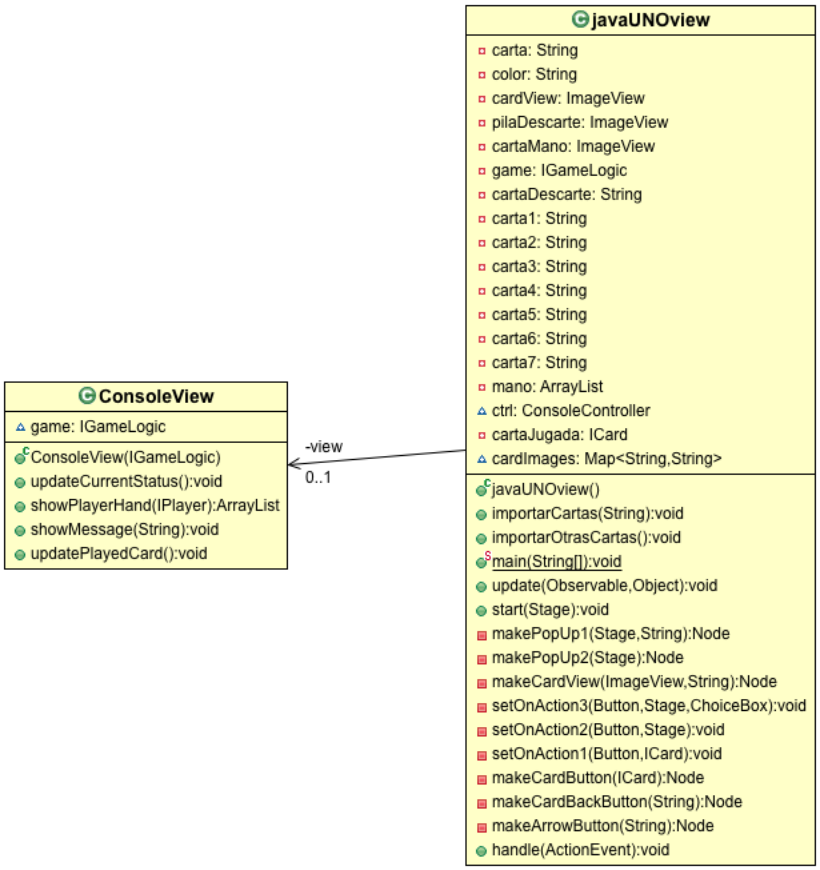

Figura 2: UML view.package

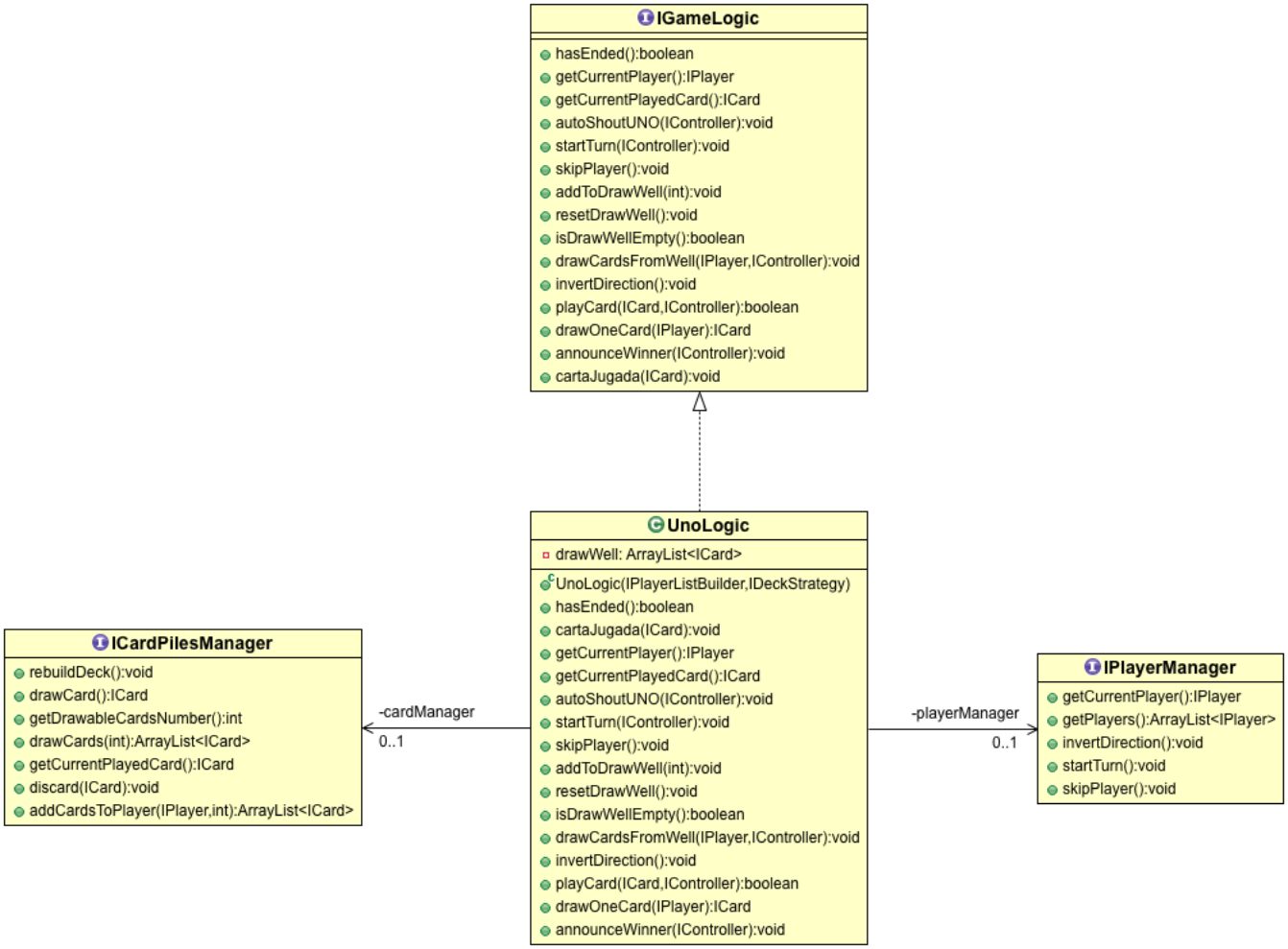

Figura 3: UML model.package

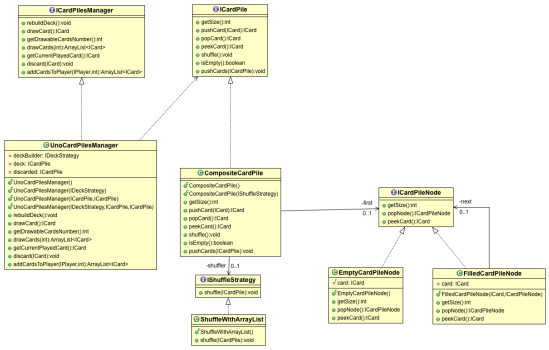

Figura 4: UML modelCard.package

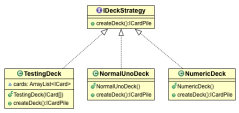

Figura 5: UML modelcarddeck.package

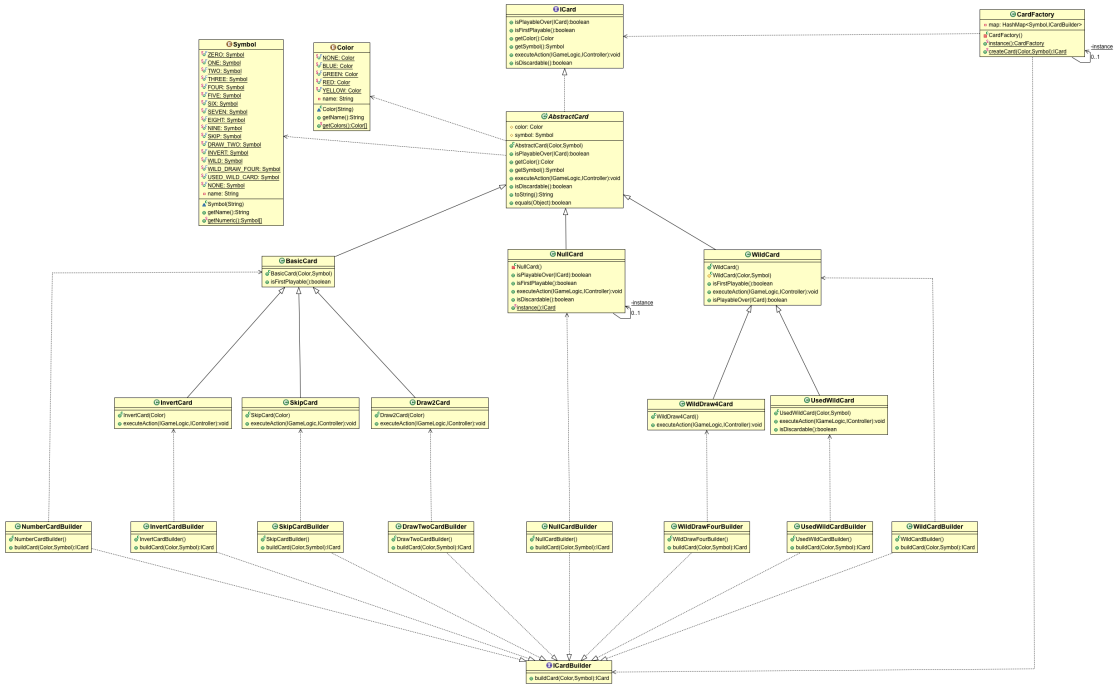

Figura 6: UML modelcardtype.package

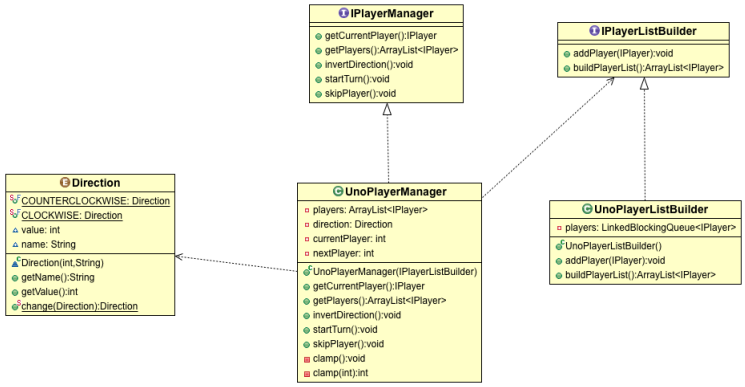

Figura 7: UML modelPlayer.package

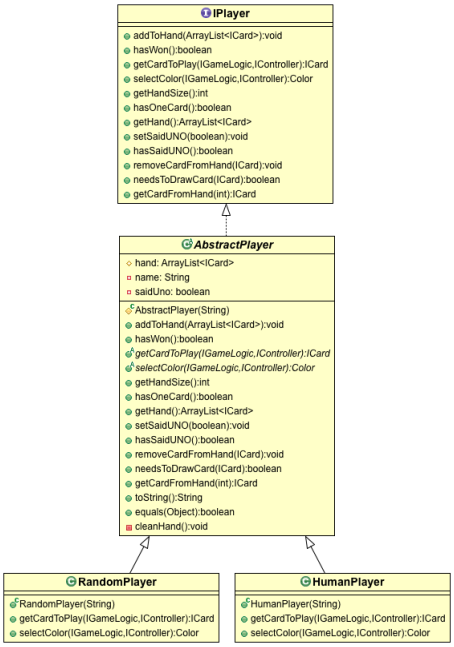

Figura 8: UML modelPlayertype.package

2 Patrones de Diseño

Los patrones de diseño utilizados en la Tarea 2 fueron:

1. Adapter: Se adaptó la clase Stack de Java, utlizando la clase ICardPile.

2. Template: Se implementó la clase DeckStrategy, la cual podría ser utilizada para crear un mazo para cualquier juego de cartas, debido a que dentro de esta se implemetaron los métodos createColorDeck y numericDeck, los cuales permiten crear un mazo de cartas de un color y uno con todas las cartas numéricas respectivamente.

3. NullObject: Se implementó la clase NullCard, con el objetivo de utilizarlo cuando fuera necesario sacar una carta, para así no tener que verificar que el arreglo que contuviera las cartas apropiadas para ser jugadas fuera nulo o estuviera vacío. También se implementó la clase NullController para testear el juego con un Human Player.

4. Factory: Se utilizó este patrón a la hora de crear la lista de jugadores y el mazo. Para el caso del mazo, se utilizó la interfaz IDeckStrategy para crear un mazo(Product) utilizando DeckStrategy(Factory), el cliente correspondía a CardPilesManager. Para crear la lista de jugadores(Producto), se utilizó la interfaz PlayerListBuilder(Factory) utilizando PlayerListBuilder, la cual era instanciada por PlayerManager(Cliente).

5. Observer: Se trató de implementar un Observer Pattern a la hora de crear la GUI, con el objetivo de que esta fuera interactiva y que el juego notificara automáticamente a la GUI de lo que estaba pasando, de tal manera de que esta se actualizara cuando fuera necesario.

6. Composite: Se implementó la clase CompositeCardPile para la implementación de la pila de cartas.

3 Cartas Extra

No se implementaron Cartas Extras.

4 Otros

Se utilizó un HashMap para albergar las imágenes de las cartas, debido a que de esta manera se podía aprovechar el método toString de las cartas para poder acceder a su imágen.
Se decidió hacer funciones del estilo makePopUp o MakeButton debido a que para muchos botones se repetía la funcionalidad que estos debían tener, por lo que era mejor crear métodos que hicieran esto, evitando así repetición de código.