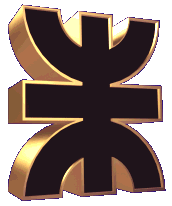
|  |
| --- |
| **Nombre y Apellido** |
| Cappelini Dario |
| Carella German |
| Centurion Damian |
| Fage Victor |
| Gonzalez Santiago |

|  |  |
| --- | --- |
| **Fecha De Presentación:** | 30/04 |
| **Fecha de Devolución:** |  |
| **Calificación** |  |
| **Firma Profesor** |  |



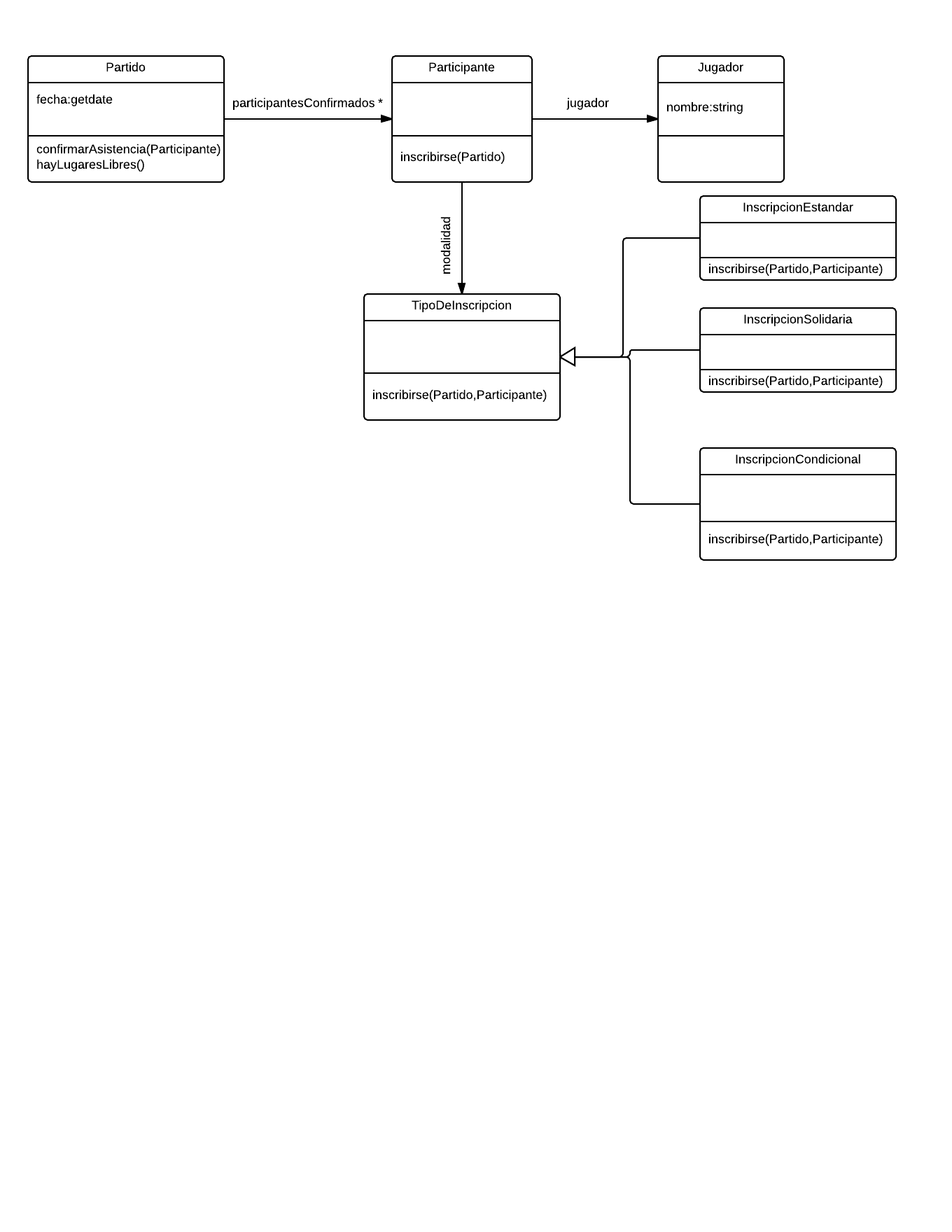
**GRUPO N° 7**

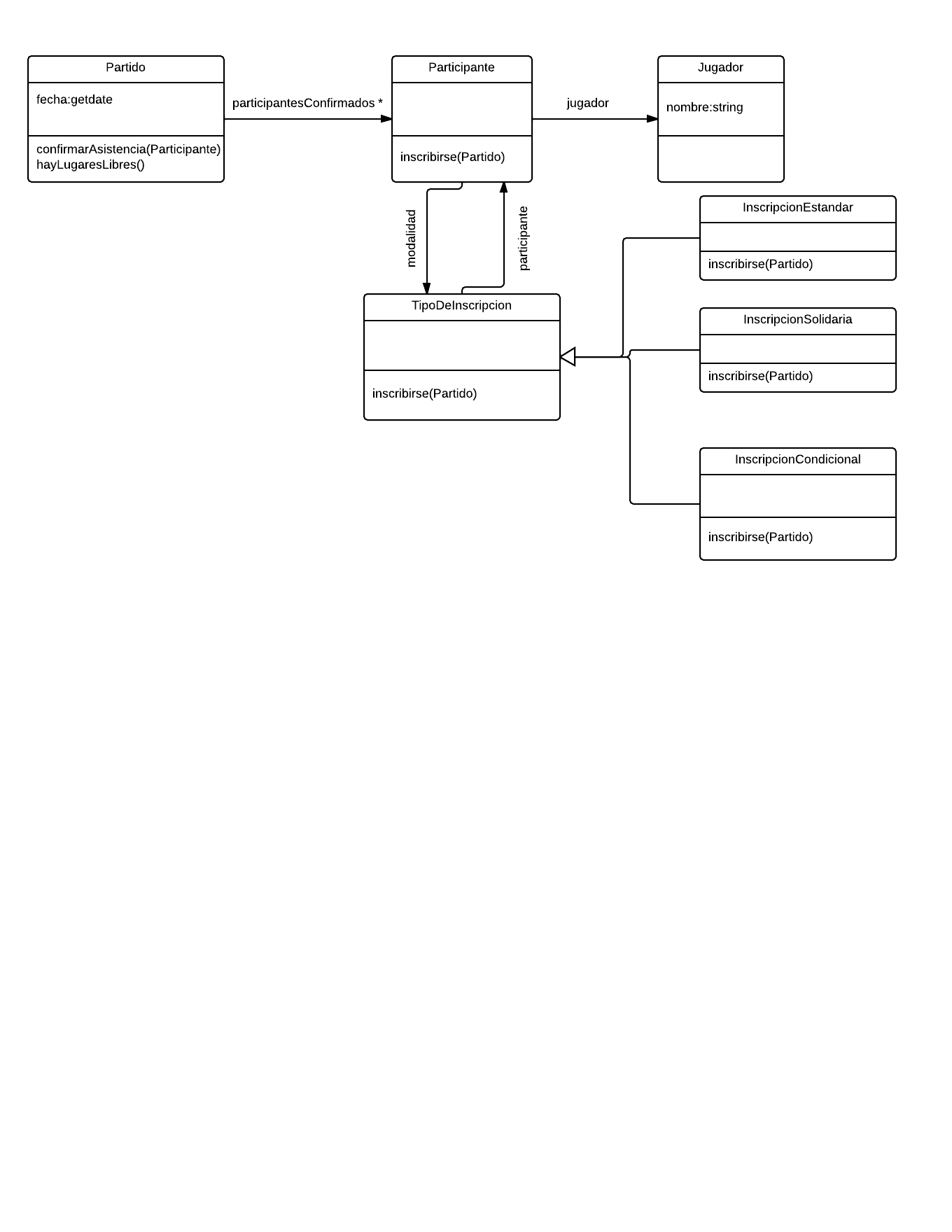
TP Anual: 1ª Entrega

***Diseño de Sistemas 2014***

***Miercoles mañana***

2. Las alternativas propuestas son las siguientes:

Utilizando un strategy sin estado:



Utilizando un strategy con estado:

La diferencia está en cómo la estrategia obtiene los datos necesarios para realizar la operación que se le pide: si conociendo a su “dueño” (stateful) o si recibiéndolo por parámetro (stateless). Elegimos la solución stateless por ahora, porque no estamos seguros de que sea realmente necesario en este momento agregarle estado a esos objetos estrategia, para poder cumplir con el requerimiento.

4. La dificultad de este problema está en el tipo de inscripción, ya que se relaciona directamente con el caso de uso que se nos pidió implementar y testear. Considerar los siguientes diagramas de secuencia, con un par de ejemplos de cómo se lleva a cabo esto:

actividades.tif

Como vemos en el diagrama de secuencia, para poder inscribir un jugador, Participante usa el método inscribirse(partido), el cual le delega al TipoDeInscripcion la forma en que lo hace. Acá usamos un strategy, el cual se implementa por herencia ya que todas las estrategias se comportarían igual si “hay lugar” en el partido, es decir, si hay menos de 10 inscriptos. El razonamiento es, en definitiva, que al haber comportamiento común vale la pena compartirlo en una superclase. En caso de no haber lugar, las inscripciones Estándar y Solidaria consultan al partido por inscripciones de “menor rango” y de haberlas, las desplaza. El mensaje que permite confirmar definitivamente a un participante es confirmarAsistencia(participante). El strategy lo llama directamente si no hay que desplazar a nadie, pero si hay que desplazar a alguien envía un mensaje específico para ese propósito.

5. Teniendo en cuenta los atributos mencionados, podemos decir que la solución que implementa el strategy acopla menos fuertemente a los componentes (ya que la estrategia no conoce a quien la usa sino que lo recibe por parámetro). Otra consecuencia es que, al no haber estado, si no es necesario ni siquiera necesito instanciar un objeto estrategia por participante, sino que puedo instanciar un objeto por cada implementación y utilizarlos indistintamente para N participantes.

Por otro lado, la solución stateful parece ser más simple ya que las interfaces de los métodos que implementan (ahora es uno, pero en un futuro podrían ser más) se simplifican por el sólo hecho de no tener que pasar al participante por parámetro. Además, suponiendo que una solución necesite algunos atributos para funcionar y otras no (como cuando haya que implementar la condición), tener estado va a permitir que sólo la estrategia que necesita ese atributo (en este caso estrategia condicional y atributo condición) lo tenga. Además, si fuera stateless, la condición la debería saber el participante, y si el participante no da la casualidad de que es condicional, va a tener un atributo condición que no va a usar.