Diseño de Software

Lenguaje de restricciones de objetos (OCL)

Contratos

PhD Oscar Franco-Bedoya

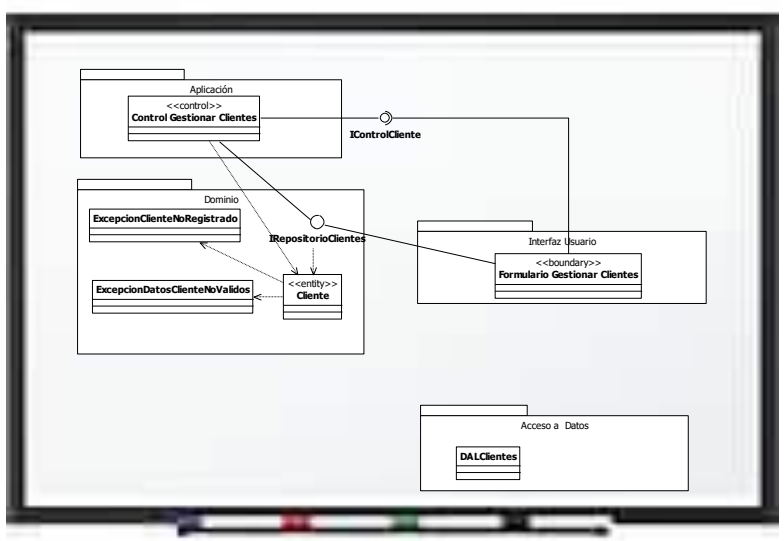
Software Engineering Coach

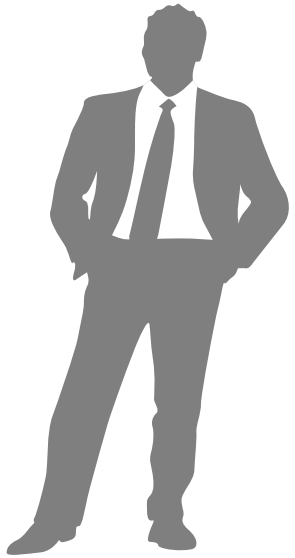
oscar.franco@ucaldas.edu.co

ohfrancob@unal.edu.co

¿No son suficientes los modelos UML?

+ precisión -ambigüedad





¿Qué otros artefactos ofrece UML?

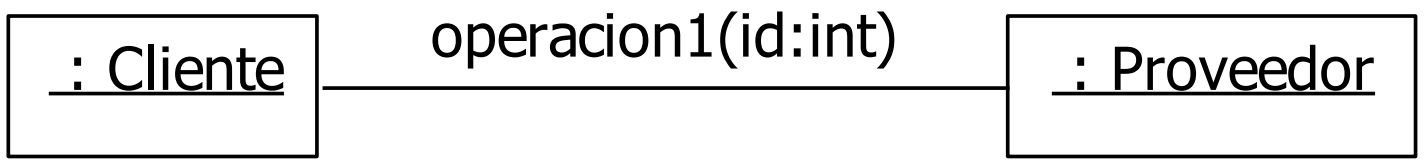
Contratos

Son **restricciones** sobre una clase *(o sus componentes)* que permiten que usuarios/clientes de la clase y desarrolladores **compartan** sus **suposiciones** sobre **la clase.**

[Meyer, 1997].

¿Qué otros artefactos ofrece UML?

Contratos



Estructura de los Contratos UML

Invariante

Es un **predicado** que es siempre verdadero para **todas las instancias** (objetos) de la clase

Precondición

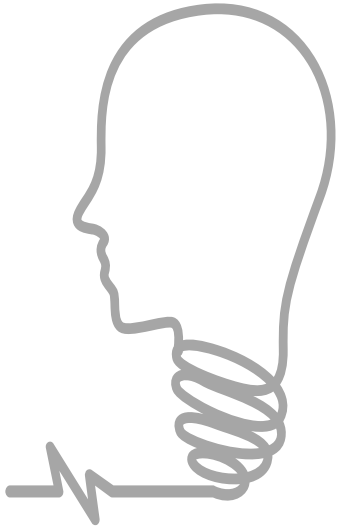
Es un **predicado** que debe ser verdadero antes que un **método** (operación) de la clase sea invocado

Postcondición

Es un **predicado** que debe ser verdadero después de que se ejecute un **método** (operación) de la clase.

Especificación de Contratos UML

Lenguaje Natural



Lenguaje Formal
OCL

Lenguaje de restricciones de objetos (OCL)

- Lenguaje formal
- No es un lenguaje de programación
- Lenguaje de especificación
  - Contratos UML
- Fuertemente tipeado

Lenguaje de restricciones de objetos (OCL)

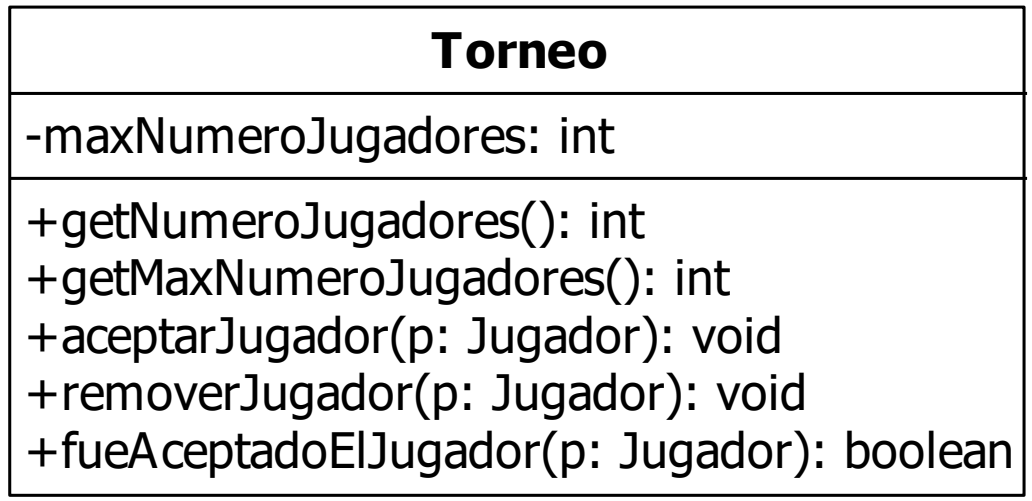
¿Qué es una restricción?

- Es una expresión que puede ser evaluada como verdadera o falsa (predicado)
- Puede especificarse en un documento, en el modelo UML o incluso en el código fuente

Especificando invariantes

Ejemplo

(Bruegge & Dutoit, 2010)



Entidad a la cual se aplica la expresión

Denota a todas las instancias de la clase

context Torneo **inv**: **self**.getMaxNumeroJugadores() > 0

Estereotipo UML
(inv, pre, post)

Especificando pre y post condiciones

Ejemplo

Torneo
-maxNumeroJugadores: int
+getNumeroJugadores(): int +getMaxNumeroJugadores(): int +aceptarJugador(p: Jugador): void +removerJugador(p: Jugador): void +fueAceptadoElJugador(p: Jugador): boolean

El contexto en las pre y post condiciones es el método

context Torneo::aceptarJugador(p:Jugador)

Especificando pre y post condiciones

Ejemplo

Torneo
-maxNumeroJugadores: int
+getNumeroJugadores(): int +getMaxNumeroJugadores(): int +aceptarJugador(p: Jugador): void +removerJugador(p: Jugador): void +fueAceptadoElJugador(p: Jugador): boolean

context Torneo::aceptarJugador(p:Jugador)
pre: !fueAceptadoElJugador(p)

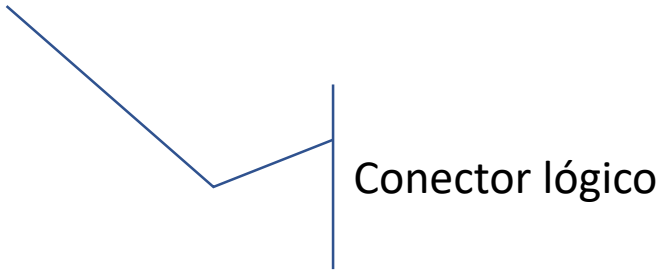
Negar/Invertir el resultado del método

Especificando pre y post condiciones

Ejemplo

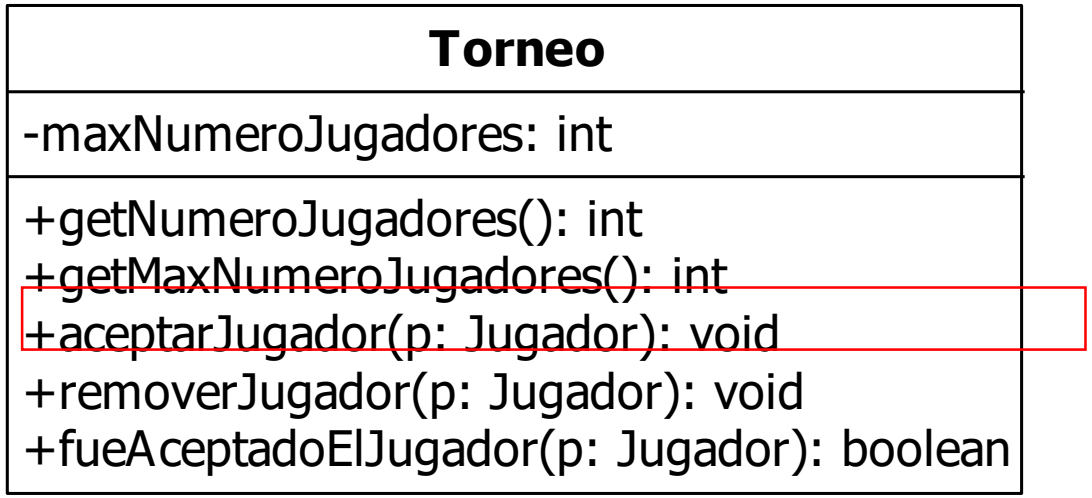
Torneo
-maxNumeroJugadores: int
+getNumeroJugadores(): int +getMaxNumeroJugadores(): int +aceptarJugador(p: Jugador): void +removerJugador(p: Jugador): void +fueAceptadoElJugador(p: Jugador): boolean

context Torneo::aceptarJugador(p:Jugador)
pre: !fueAceptadoElJugador(p) **and** getNumeroJugadores() < getMaxNumeroJugadores()



Especificando pre y post condiciones

Ejemplo

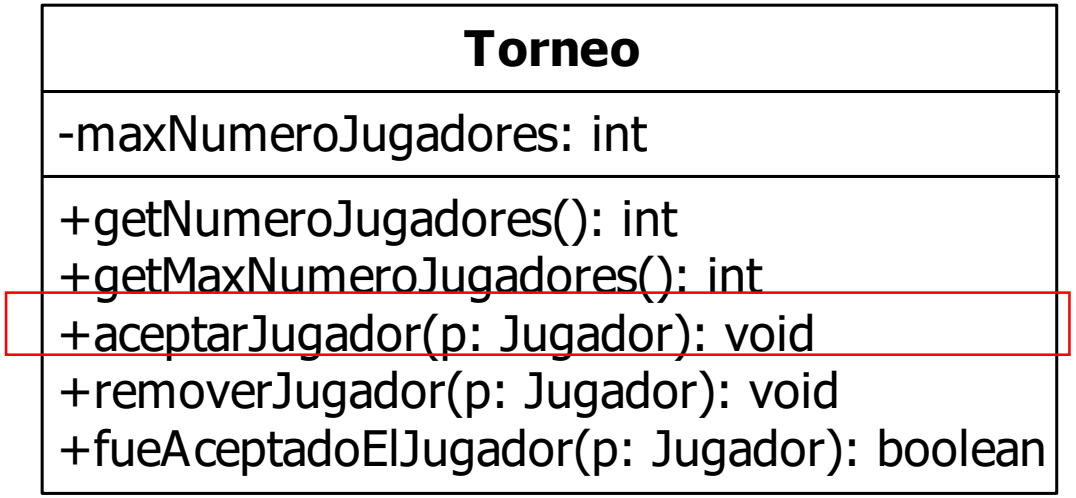


context Torneo::aceptarJugador(p:Jugador)
pre: !fueAceptadoElJugador(p) **and** getNumeroJugadores() < getMaxNumeroJugadores()

Aceptar jugador asume que el jugador no ha sido ya aceptado antes y que el numero de jugadores aceptados no ha llegado al máximo

Especificando pre y post condiciones

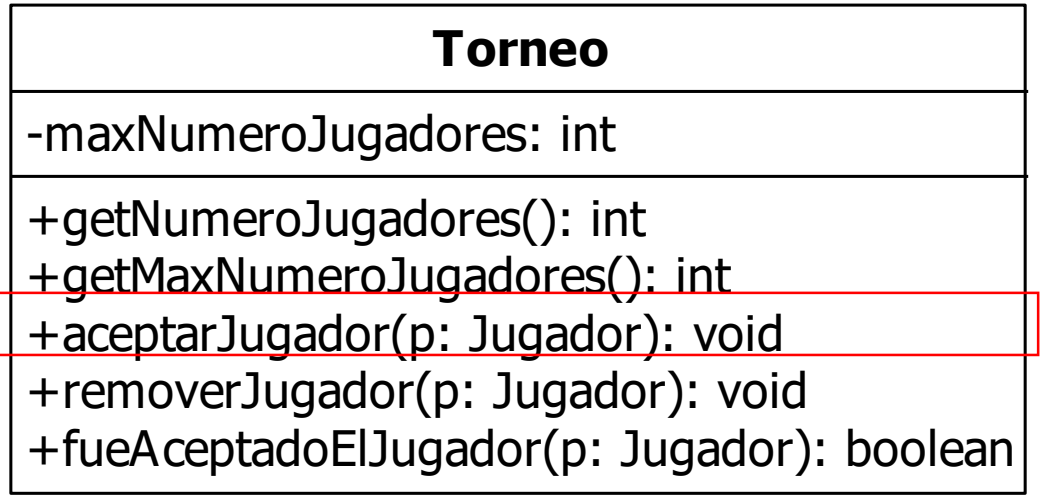
Ejemplo



context Torneo::aceptarJugador(p:Jugador)
pre: !fueAceptadoElJugador(p) **and** getNumeroJugadores() < getMaxNumeroJugadores()
post: fueAceptadoElJugador(p)

Especificando pre y post condiciones

Ejemplo

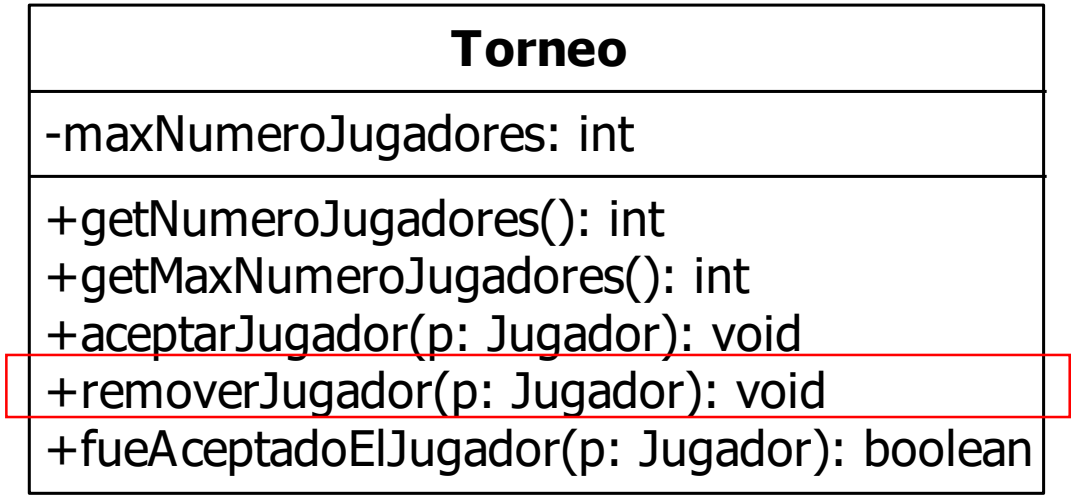


context Torneo::aceptarJugador(p:Jugador)
pre: !fueAceptadoElJugador(p) **and** getNumeroJugadores() < getMaxNumeroJugadores()
post: fueAceptadoElJugador(p) **and**
getNumeroJugadores() = **self@pre.**getNumeroJugadores() + 1

Hace referencia al valor retornado por el método getNumeroJugadores() antes de la invocación del método aceptarJugador()

Especificando pre y post condiciones

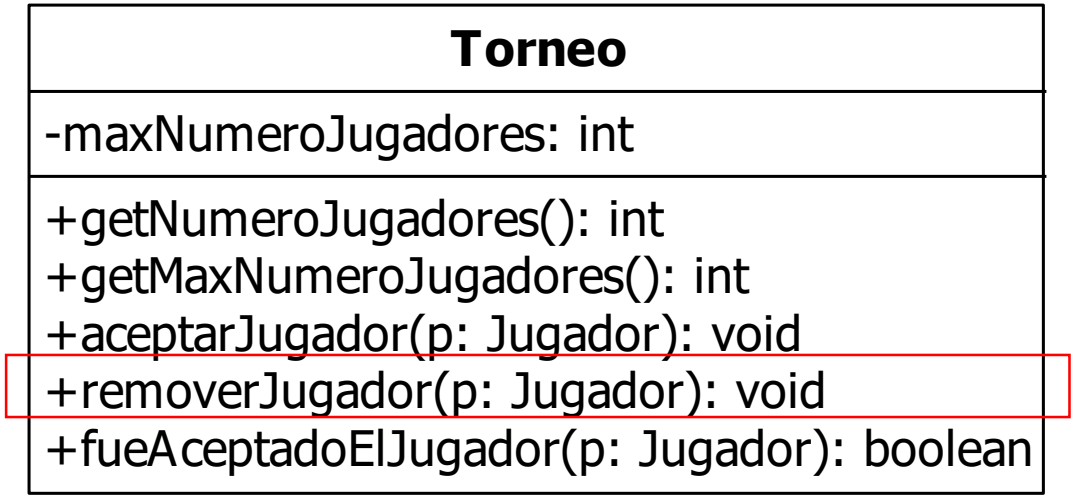
Ejemplo



context Torneo::removerJugador(p:Jugador)

Especificando pre y post condiciones

Ejemplo



context Torneo::removerJugador(p:Jugador)
pre: fueAceptadoJugador(p)

Especificando pre y post condiciones

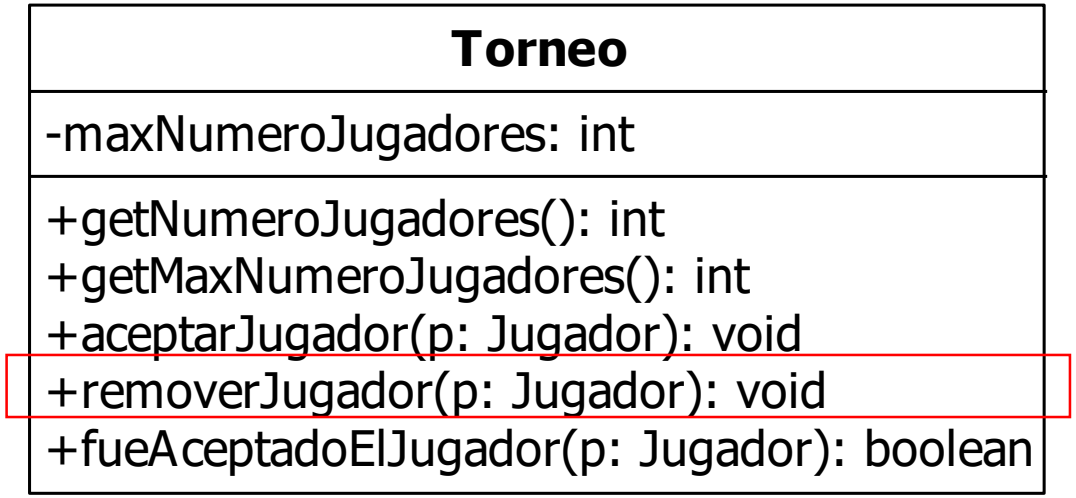
Ejemplo

Torneo
-maxNumeroJugadores: int
+getNumeroJugadores(): int +getMaxNumeroJugadores(): int +aceptarJugador(p: Jugador): void +removerJugador(p: Jugador): void +fueAceptadoElJugador(p: Jugador): boolean

context Torneo::removerJugador(p:Jugador)
pre: fueAceptadoJugador(p)
post: !fueAceptadoJugador(p

Especificando pre y post condiciones

Ejemplo



context Torneo::removerJugador(p:Jugador)
pre: fueAceptadoJugador(p)
post: !fueAceptadoJugador(p) **and** getNumeroJugadores() = **self@pre.**getNumPlayers() - 1

OCL en el código fuente

Javadoc

```
/** Un Torneo es una serie de Juegos entre Jugadores
* El torneo termina con solo un jugador como ganador.
*/
public class Torneo {
/** El máximo número de jugadores es positivo todo
el tiempo.
* @invariant maxNumeroJugadotres > 0
*/
private int maxNumeroJugadores;
/** la lista de jugadores contiene referencias a
los jugadores registrados
*/
private List jugadores;

/** El método aceptarJugador() asume que el jugador no
ha sido aceptado antes y todavia hay disponibilidad
* @pre !fueAceptadoElJugador(p)
* @pre getNumeroJugadores() < maxNumeroJugadores
* @post fuePlayerAccepted(p)
* @post getNumeroJugadores()=self@pre.getNumeroJugadores()+1
*/
public void aceptarJugador (Player p) {…}
}
```

Resumen


Invariantes
Precondiciones
Postcondiciones
Contratos
OCL
Porque mas formalismo
Restricciones