OCL (**O**bject **C**onstrain **L**enguaje)

Precisar comportamiento de clases, operaciones o asociaciones

Sintaxis:

Context OBJETO

ETIQUETA: INSTANCIA RESTRICCION

Context Objeto::METODO()

ETIQUETA: RESTRICCION

Ejemplos:

context Persona

inv: self.edad >= 18

context Cuenta::retirar(monto: Real)

pre: monto > 0 and monto <= self.saldo

post: self.saldo = self.saldo@pre – monto

context Cliente

inv: self.pedidos->forAll(p | p.total >= 0)

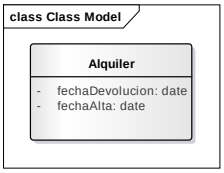
context Factura::total: Real

derive: self.items->collect(i | i.precio * i.cantidad)->sum()

Tipos de Restricciones

INVARIANTES

Condición que deben cumplir todos los objetos de una clase



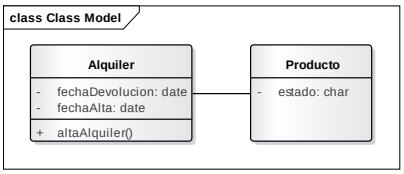
"La fecha de devolución del alquiler debe ser igual o posterior a la fecha en la que se realizó el alta del mismo."

Context Alquiler

Inv FechaDevolucionValida: Self.fechaDevolucion >= Self.fechaAlta

PRE-CONDICIONES

Condición que debe cumplirse antes de ejecutar un método



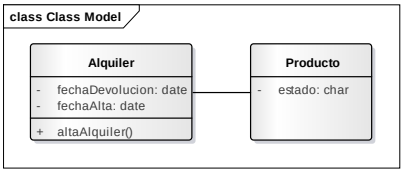
"Para dar de alta un alquiler, el producto asociado debe estar en estado disponible."

Context Alquiler::altaAlquiler()

Pre ProductoDisponible: Self.producto.estado = 'disponible'

POST-CONDICIONES

Condición que debe cumplirse después de ejecutar un metodo



"Luego de dar de alta un alquiler, el estado del producto debe quedar actualizado como alquilado."

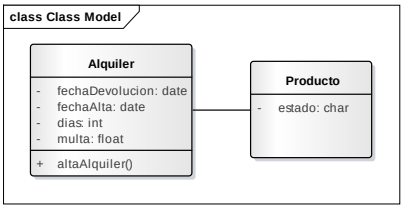
Context Alquiler::altaAlquiler()

Post EstadoProductoActualizado: Self.producto.estado = 'alquilado'

IMPLIES

Si pasa esto, entonces debe cumplirse aquello (P → Q)(

Útil cuando algo solo debe cumplirse bajo una cierta condición



"Si la fecha de devolucion supera la fecha de alta más la cantidad de dias del alquiler, entonces el valor de la multa debe ser mayor a cero."

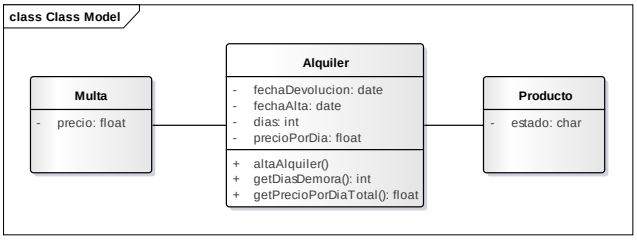
Context Alquiler

Inv multaCalculada: Self.fechaDevolucion > Self.fechaAlta + Self.dias

Implies Self.multa > 0

DERIVE

Determina como se calcula un atributo derivado



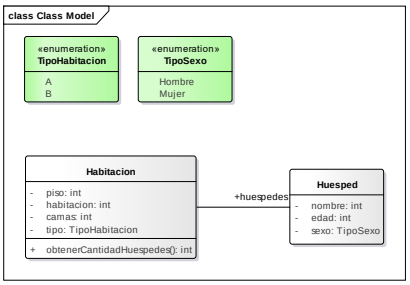
"El precio de la multa se deriva multiplicando la cantidad de dias de demora del alquiler por el 15% del precio total por dia del mismo."

Context Multa::precio:float

Derive: Self.alquiler.getDiasDemora() * 0.15 * Self.alquiler.getPrecioPorDiaTotal

BODY

Especidicar el valor devuelto por una operación o metodo definido en unca clase. ***SE ESCRIBE EL CUERPO DEL METODO***

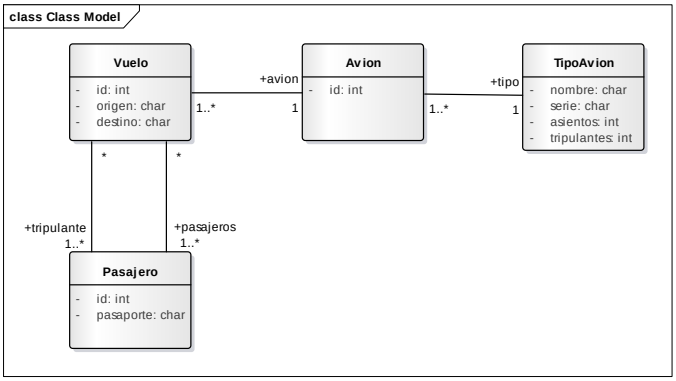


"La cantidad de huespedes en una habitacion se obtiene contando los elementos de la coleccion de huespedes asociados a dicha habitacion."

Context Habitación::ObtenerCantidadHuspedes():integer

Body: Self.huespedes->size()

Navegación Combinada

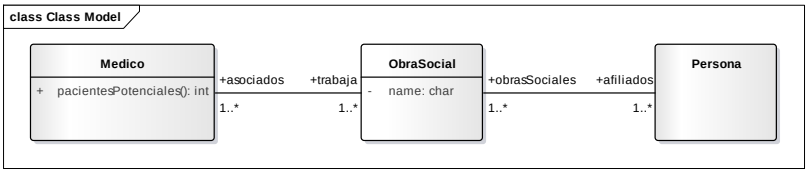


"La cantidad de pasajeros en un vuelo no debe superar la cantidad de asientos disponibles segun el tipo de avion asignado."

Context Vuelo

Inv: Self.pasajeros → size() <= Self.avion.tipo.asientos

Otro ejemplo:



"un médico quiere dimensionar los pacientes potenciales de las obras sociales 'IOMA'y 'PAMI'en las cuales recientemente se ha asociado."

Context Medico::pacientesPotenciales(): int

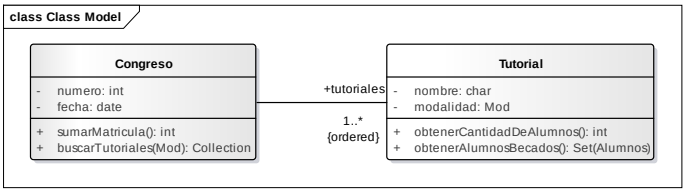
Body: Self.trabaja → select(ob | ob.name="IOMA" OR ob.name="pami").afiliados → count()

Comentado [Dp1]: Puede haber repetidos ya que es un bag

Context Medico::pacientePotenciales(): int

Body: Self.trabaja → Select(ob | ob.name="IOMA" or ob.name="PAMI").afiliados → asSet() → count()

Comentado [Dp2]: Para evitar repetidos



"podría consultar matrícula del primer tutorial, el cuarto y el ultimo y sumarlas."

Context Congreso::sumarMatricula():int

Body: Self.tutoriales → First().obtenerCantidadDeAlumnos() +

Self.tutoriales → at(4).obtenerCantidadDeAlumnos() +

Self.tutorial → last().obtemerCantidadDeAlumnos()

context Congreso::buscarTutoriales(unaModalidad: Mod) : Collection(Tutorial)

body: self.tutoriales->select(t | t.modalidad = unaModalidad)

Como la coleccion tutoriales es una **Sequence** (por ejemplo, esta ordenada), el resultado de aplicar select sobre una Sequence tambien es una **Sequence** (no cambia el tipo a Set ni a Bag).

Operadores de collection

¿Cantidad elementos? :: Devuelve entero

Self.pedidos → size() > 5

¿esta vacia? :: Devuelve true si está vacío

Self.alquileres → isEmpty()

¿tiene almenos un elemento? :: Devuelve true si tiene almenos uno

Self.alquileres → notEmpty()

Filtrar los que cumplen la condicion :: Colección del mismo tipo

Self.libros → select(l | l.año > 2010)

Filtra los que no cumplen la condición :: Colección del mismo tipo

Self.libros → reject(l | l.año > 2010)

Verifica que todos los elementos cumplan :: Devuelve boolean

Self.alumnos → forAll(a | a.edad > 17)

Verifica que al menos uno cumple :: Devuelve boolean

Self.productos → exists(p | p.precio < 100)

Mapear cada ítem a otro valor o expresión :: Colección de otro tipo

Self.libros → collect(l | l.título)

Acumular con expresión :: Devuelve un numerico

self.alumnos → iterate(a : Alumno; maxEdad : Integer = 0 |

if a.edad > maxEdad then a.edad else maxEdad endif)

"Encontrar el mayor en una coleccion"

Sumar valores numéricos :: Devuelve un numerico

Self.productos → collect(p | p.precio) → sum()

Cuenta las veces que aparece un valor especifico :: Devuelve numeric

Self.palabras → count('hola')

Una coleccion contiene un determinado elemento :: Devuelve bool

Self.prestamos → includes(p)

Una colección contiene todos los elementos de una colección

self.prestamos → includesAll(prestamosActivos)

Unir 2 colecciones :: Una colección del mismo tipo

Self.clientes → select(c | c.edad < 18) → unión(Self.clientes→ select(c | c.edad > 65))

Recuperar todos los objetos creados a partir de una clase especifica

Devuelve un Set(ClassName)

actúa sobre la clase.

“Todos los pedidos tengan al menos un ítem asociado”

Pedido.allInstances() → forAll(p | p.item → size() > 0)

Preguntar si un objeto es de un tipo especifico :: Devuelve bool

Context Pedido

Inv: Self.item → exists(i | i.producto.oclIsKindOf(Fisico))