



**UNIVERSITÀ  
DI TRENTO**

Dipartimento di Ingegneria e Scienza dell'Informazione

**Progetto:**

**Animati**



**Titolo del documento:**

**???**

**Gruppo:**

**T51**

# Indice

Diagramma delle Classi ..... 3

OCL ..... ?

## **Scopo del documento**

[...]

## Diagramma delle classi

[...]

## Codice in Object Constraint Language

In questo capitolo è descritta in modo formale la logica prevista nell'ambito di alcune operazioni di alcune classi. Tale logica viene descritta in Object Constraint Language (OCL) perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

### Cronometro

**stato** : Enum

**Invarianti:**

- stato assume i valori "reset", "run", "pause"

```
context Cronometro inv :
(stato = "reset") OR (stato = "run") OR (stato = "pause")
```

Metodo	Precondizioni	Postcondizioni
start()	<b>stato</b> deve essere in "reset"	<b>stato</b> assume valore "run"
stop()	<b>stato</b> deve essere in "run"	<ul style="list-style-type: none"> <li>• <b>stato</b> assume valore "pause"</li> <li>• <b>parziali</b> è una lista vuota</li> </ul>
riprendi()	<b>stato</b> deve essere in "pause"	<b>stato</b> assume valore "run"
ripristina()	<b>stato</b> deve essere in "pause"	<b>stato</b> assume valore "reset"
parziale()	<b>stato</b> deve essere in "run"	<ul style="list-style-type: none"> <li>• <b>stato</b> rimane al valore "run"</li> <li>• aggiunto tempo a lista <b>parziali</b></li> </ul>

```
context Cronometro::start()
pre: self.stato = "reset"
post: self.stato = "run"
```

```
context Cronometro::stop()
pre: self.stato = "run"
post: (self.stato = "pause") AND (self.parziali -> isEmpty())
```

```
context Cronometro::riprendi()
pre: self.stato = "pause"
post: self.stato = "run"
```

```
context Cronometro::ripristina()
pre: self.stato = "pause"
post: self.stato = "reset"
```

```
context Cronometro::parziale()
pre: self.stato = "run"
post: (self.stato = "run") AND (self.parziali -> includes (self.tempo))
```

## Segna-Punti

**contatori : Map<String,int>**

**Invarianti:**

- contatori contiene al più 99 elementi

```
context Segna-Punti inv :
contatori -> size() <= 99
```

Metodo	Precondizioni	Postcondizioni
aggiungiContatore(nome : String)	<ul style="list-style-type: none"> <li>• il nome non può eccedere i 99 caratteri</li> <li>• il numero di squadre può essere al massimo 98</li> </ul>	il valore del contatore col nome scelto è pari a 0
incrementa(nome : String)	il contatore può assumere valore minore di 500	il contatore col nome scelto viene incrementato di 1
decrementa(nome : String)	il contatore può assumere valore maggiore di -500	il contatore col nome scelto viene decrementato di 1

```
context Segna-Punti::aggiungiContatore(nome : String)
pre: (nome -> size() <= 99) AND (self.contatori -> size() <= 98)
post: self.contatori[nome] = 0
```

```
context Segna-Punti::incrementa(nome : String)
pre: self.contatori[nome] < 500
post: self.contatori[nome] = self.contatori[nome]@pre + 1
```

```
context Segna-Punti::decrementa(nome : String)
pre: self.contatori[nome] > -500
post: self.contatori[nome] = self.contatori[nome]@pre - 1
```

## Timer

**stato : Enum**

**Invarianti**

- stato assume i valori "reset", "run", "pause"

```
context Timer inv :
  (stato = "reset") OR (stato = "run") OR (stato = "pause")
```

Metodo	Precondizioni	Postcondizioni
start()	<b>stato</b> deve essere in "reset"	<b>stato</b> assume valore "run"
riprendi()	<b>stato</b> deve essere in "pause"	<b>stato</b> assume valore "run"
stop()	<b>stato</b> deve essere in "run"	<b>stato</b> assume valore "pause"
annulla()	<b>stato</b> deve essere in "pause"	<b>stato</b> assume valore "reset"
imposta(tempo : Time)	il tempo fornito deve essere positivo	<b>tempo</b> assume valore del tempo fornito
scegliSuono(suono : URL)		la sorgente del suono è quella scelta

```
context Timer::start()
pre: self.stato = "reset"
post: self.stato = "run"
```

```
context Timer::riprendi()
pre: self.stato = "pause"
post: self.stato = "run"
```

```
context Timer::stop()
pre: self.stato = "run"
post: (self.stato = "pause")
```

```
context Timer::annulla()
pre: self.stato = "pause"
post: self.stato = "reset"
```

```
context Timer::imposta(tempo : Time)
pre: tempo > 0
post: self.tempo = tempo
```

```
context Timer::scegliSuono(sorgente : URL)
post: self.suono.sorgente = sorgente
```

## Dado

**tipo : Enum**

**Invarianti:**

- tipo assume i valori "Numeri", "Colori", "Immagini"

Metodo	Precondizioni	Postcondizioni
estrai() : Object	<ul style="list-style-type: none"> <li>se riestrazione assume il valore "False" l'utente non può estrarre un numero maggiore degli elementi selezionati</li> <li>deve venire selezionato il tipo di oggetto estratto</li> </ul>	vengono estratti gli elementi del tipo selezionato
cambiaModalità(riestrazione : bool)		l'attributo riestrazione viene impostato al valore selezionato
scegliTipo(tipo : Enum)		l'attributo tipo viene impostato al valore selezionato
aggiungi(oggetto : Object)		viene aggiunto un oggetto alla lista di quelli selezionabili

// TODO: @teopan21

## Fischietto

Metodo	Precondizioni	Postcondizioni
<<<<<<< HEAD		
start()	il fischietto non deve essere premuto	il suono è in riproduzione
stop()	il fischietto deve essere premuto	il suono non è in riproduzione
scegliSuono(sorgente)		la sorgente del suono è quella scelta
=====		
start()	premuto deve assumere il valore "True"	il suono viene riprodotto
stop()	premuto deve assumere il valore "False"	nessun suono viene riprodotto
scegliSuono()		l'attributo suono viene impostato con il suono scelto

631652eda62208f8909c8d984c4fde2027a33600

```
context Fischietto::start()
pre: NOT self.premuto
```



```
post: self.suono.inRiproduzione = true
```

```
context Fischietto::stop()
pre: self.premuto
post: self.suono.inRiproduzione = false
```

```
context Fischietto::scegliSuono(sorgente : URL)
post: self.suono.sorgente = sorgente
```

## Creazione Squadre

**metodo : Enum**

**Invarianti:**

- metodo assume i valori "Round robin", "Random", "Fill first" e "Balanced"

```
context Creazione Squadre inv :
(stato = "Round robin") OR (stato = "Random") OR (stato = "Fill first") OR (stato = "Balanced")
numeroSquadre <= 99
numeroComponenti <= 99
numeroPartecipanti <= 9801
```

Metodo	Precondizioni	Postcondizioni
inserisciNumSquadre(numero : int)	numeroSquadre deve essere minore di 99	informazioni viene incrementato di 1
inserisciNumComponenti(numero : int)	numeroComponenti deve essere minore di 99	informazioni viene incrementato di 1
inserisciNumPartecipanti(numero : int)	numeroPartecipanti deve essere minore di 9801	informazioni viene incrementato di 1
scegliMetodo(metodo : Enum)		l'attributo metodo viene impostato con quello scelto
inserisciNome(nome : String)	la lunghezza dei nomi non deve eccedere i 99 caratteri	l'attributo nomi viene impostato con i nomi scelti
estrai()	<ul style="list-style-type: none"> <li>• informazioni deve essere uguale a 2 o 3</li> <li>• nel caso i tre valori: numeroSquadre, numeroPartecipanti e numeroComponenti non siano compatibili, verranno solamente considerati numeroPartecipanti e numeroSquadre.</li> </ul>	<ul style="list-style-type: none"> <li>• se metodo assume il valore "Random" l'ordine delle squadre assegnate sarà completamente casuale</li> <li>• se metodo assume il valore "Round robin" l'assegnamento delle squadre sarà sequenziale</li> <li>• se il metodo assume il valore "Fill first"</li> </ul>

		<p>l'assegnamento avverrà per completamento delle squadre, ovvero riempiendo i posti di ogni squadra prima di procedere con l'assegnamento per la prossima</p> <ul style="list-style-type: none"> <li>• se il metodo assume il valore "Balanced" tutte le squadre dovranno avere lo stesso numero di partecipanti prima di procedere con gli assegnamenti</li> <li>• qualsiasi sia il metodo scelto, ogni volta che un partecipante viene assegnato ad una squadra il contatore di quella squadra viene incrementato di 1</li> </ul>
--	--	--

## Suono

Metodo	Precondizioni	Postcondizioni
start()		il suono viene riprodotto
stop()		il suono non viene riprodotto

## Utente

**ruolo : Enum**

**Invarianti:**

- ruolo assume i valori "Amministratore" e "Base"

```
context Utente inv :
(ruolo = "Amministratore") OR (ruolo = "Base")
```

Metodo	Precondizioni	Postcondizioni
cambiaRuolo(utente, promotore, nuovoRuolo)	<ul style="list-style-type: none"> <li>• un amministratore può essere declassato a utente comune unicamente dall'amministratore che lo ha promosso</li> <li>• se l'attributo ruolo dell'utente ha il valore "Amministratore", il suo ruolo può essere</li> </ul>	l'attributo ruolo dell'utente assume il valore di nuovoRuolo

cambiato solo se promotore è diverso dall'attributo promossoDa dell'utente

- il promotore deve avere ruolo "Amministratore"

## Autenticazione

Metodo	Precondizioni	Postcondizioni
verificaOAuth()		
generaToken()		
ottieniUtente() : Utente		
logout()		

// TODO: @teopan21

## Catalogo

Metodo	Precondizioni	Postcondizioni
aggiornaCatalogo() : Attività[0...N]		l'attributo ultimoAggiornamento assume il valore della data corrente
filtra(cerca : String, etichette : Etichette[0...N]) : Attività[0...N]	<ul style="list-style-type: none"> <li>nella barra di ricerca del titolo non si possono inserire più di 20 caratteri</li> <li>i due valori della durata media sono compresi tra 0 e 999, sono interi e il primo è minore del secondo</li> <li>il numero di partecipanti non può superare 99</li> </ul>	il catalogo viene filtrato secondo le etichette previste
creaAttività(attività : Attività)	<ul style="list-style-type: none"> <li>la descrizione non può superare i 2000 caratteri</li> <li>i due valori della durata media sono compresi tra 0 e 999, sono interi e il primo è minore del secondo</li> <li>il numero di partecipanti non può superare 99</li> <li>il titolo non può superare i 20 caratteri di lunghezza</li> </ul>	viene aggiunta una nuova attività al catalogo

```
context Catalogo::aggiornaCatalogo()
post: self.ultimoAggiornamento = Data.now()
```

// TODO: @teopan21

## Lista di Attività

Metodo	Precondizioni	Postcondizioni
creaLista() : listaAttività	<ul style="list-style-type: none"> <li>il nome della lista di attività non può superare i 20 caratteri di lunghezza</li> <li>un utente non può creare più di 99 liste di attività</li> <li>il nome della lista di attività non può essere uguale al nome di un'altra lista già presente</li> <li>il nome della lista di attività non può essere nessuno</li> </ul>	viene creata una nuova lista di attività
aggiungiAttività(attività : Attività)	il numero di attività in una lista non può superare 9999	l'attività scelta viene aggiunta alla lista
esporta(formato : String) : File		la lista viene esportata in formato pdf o json
eliminaAttività(indice : int)		l'attività con l'indice scelto viene rimossa dalla lista

// TODO: @teopan21

## Attività

// TODO: @teopan21 Invarianti

Metodo	Precondizioni	Postcondizioni
modifica(attivitàModificata : Attività)	<ul style="list-style-type: none"> <li>la descrizione non può superare i 2000 caratteri</li> <li>i due valori della durata media sono compresi tra 0 e 999, sono interi e il primo è minore del secondo</li> <li>il numero di partecipanti non può superare 99</li> <li>il titolo non può superare i 20 caratteri di lunghezza</li> </ul>	<ul style="list-style-type: none"> <li>tutti gli attributi assumono il valore dell'attività modificata</li> <li>l'attributo ultimaModifica assume il valore della data corrente</li> </ul>

## Segnalazione

// TODO: @teopan21 invarianti

Metodo	Precondizioni	Postcondizioni
inviaSegnala(autore : Utente, attività : Attività, voto : String)	l'attributo messaggio non può superare i 500 caratteri di lunghezza	viene aggiunta una segnalazione per l'attività scelta

---

## Valutazione

// TODO: @teopan21 invarianti

Metodo	Precondizioni	Postcondizioni
inviaValutazione(autore : Utente, attività : Attività, voto : String)	il voto inserito deve essere un numero decimale compreso tra 0 e 5, con scarto di 0.5	<ul style="list-style-type: none"><li>viene aggiunta la valutazione all'attività scelta</li><li>cambia la media di voti dell'attività scelta</li></ul>

---