

- per semplicità ho riassunto getter e setter di diverse classi
- qui dovrebbero esserci i metodi principali delle classi, nello sviluppo probabilmente aggiungeremo diversi metodi (soprattutto privati) che tuttavia adesso ha poco senso indicare perchè difficili da ipotizzare e non fondamentali

## CONCORRENTE

//ricordarsi che è la gara che aggiunge il concorrente

```
+ iscriviGara(in codice: String): boolean()           // chiama g.nuovaIscrizione
+ annullaIscrizione (in codice String): boolean()
+ visualizzaProfilo():void()
+ carriera():void()
+ modificaProfilo():void() // racchiude un pò tutti i setter
+ iscrizioni():void()
+ aggiungiRecensione(in codice:String, in rec:Recensione):boolean()
```

## ARBITRO

```
+ avviaGara(in codice String):void()           //gara passa a stato in corso, solo una alla volta
+ rinuncia(in codice:String): void() //chiamerà g.trovaArbitro()
// i seguenti metodi si usano solo se la gara in corso
+ segnaAssenze(in concorrenti: List<Concorrente>):void()
+ generaGruppi(in concorrenti: List<Concorrente>):void()
+ segnaPunto(in concorrente:Concorrente):void() // aumenta di 1 punteggio(live)
+ segnaPunto(in concorrente:Concorrente, in punti:Integer):void() // class differite
+ annullaConcorrente(in concorrente:Concorrente, in motivo:String):void()
+ penalizzaConcorrente(in concorrente: Concorrente, in quantita: Integer, in motivo:String)
: void() //chiama p.penalità
+ salvaPunti(in idSett: String, in numTurno: Integer):void()
//chiamerà g.classificaTurno
```

## SOCIETA

```
+ iscrizioneGaraGruppi(in concorrenti: List<Concorrente>, in CodiceGara: String): boolean()
+ iscrizioneGaraSingolo(in concorrente: Concorrente, in CodiceGara: String): boolean()
+ getProfilo():void()
+ aggiungiSocio(in concorrente:Concorrente):void()
+ abbandonoSocio(in concorrente:Concorrente):void()
+ modificaProfilo():void() // racchiude un pò tutti i setter
+ nuovaGara(in gara:Gara):void() //in attesa conferma amministratore
+ regolamento():String()
```

## GARA

```
+ cambiaStato(in stato:StatoGara):void()           //confermata/in attesa/rinviata/annullata
+ cambiaStatoSvolgimento(in stato:StatoSvolgimento):void() //corso/intervallo/finita
+ squalifica(in concorrente:Concorrente, in motivo:String):void() //usato pure per assenze
+ preparaTurni(in concorrenti: List<Concorrente>) //metodo chiamato dal costruttore
+ preparazioneGara(in : gruppi:List<List<Concorrente>>):void //chiamerà metodo sotto
- assegnaTurno(in turno:Turno, in gruppo: List<Concorrente>):void()
+ classificaTurno(in turno:Turno, in punteggi:List<Punteggio>):void()
```

- + trovaArbitro():void()
- + nuovaIscrizione(in cf: String, out numIscritti: Integer): boolean()
- + iscrizioneMultipla(in concorrentiPresenti: List<Concorrente>): boolean()
- verificaData(): boolean()
- verificaMax(): boolean()
- verificaMin(): boolean()
- + getClassifica():List<Punteggio>()

#### CAMPIONATO

- + getClassificaTotale():List<Punteggio>()
- + classificaProva(in numProva:Integer):List<Punteggio>()

#### AMMINISTRATORE

- + nuovaGara(in gara:Gara):void() //in attesa conferma amministratore  
//visto che è lo stesso metodo di società ci sta bene un'interfaccia :)
- + confermaProposta(in numGara): void() //numGara è l'identificativo statico
- + negaProposta(in numGara, in motivo:String)
- notificaSocieta():void() // per avvisare esito
- notificaComune():void() //se proposta confermata
- + regolamento():String() //altro metodo comune con società (interfaccia)

#### TURNO

- + calcolaClassifica(in punteggi:List<Punteggio>):void()
- + getClassifica():List<Punteggio>()

#### PUNTEGGIO

- + mostraPenalita(): String()
- + setPunti(in totPunti:Integer ):void()
- + getPunti():Integer()
- + squalifica(): void()
- + penalità(in quantità:Integer, in motivo:String, out numPunti:Integer):void()

#### CONTRATTO

- + rinnova(in dataFine: Date, in premio: String): void()

#### CAMPOGARA

- + getCaratteristiche():String()
- + setCaratteristiche():void()

#### SETTORE

- + setCaratteristiche():void()
- + getCaratteristiche():String()