Contratti

**Precondizione globale:** Il *Manager* **M** esiste dall’inizio del caso d’uso e resta lo stesso per tutto l’UC.

# Iniziare partita

## iniziaCreazionePartita(porta, parolaChiave)

**Precondizioni:** *porta* è un numero, *parolaChiave* è un testo. Non esistono oggetti *Server*

**Postcondizioni**: viene creato un oggetto *Server* con attributi pari ai parametri dell’operazione.

## creaPartita(robodromo, nMaxGiocatori, nRobotPerGiocatore, condizioneFinePartita, upgradeInizialiSiONo)

**Precondizioni:** esiste un oggetto *Server* **S**, *robodromo* è un testo che identifica un tipo di robodromo, *nMaxGiocatori* e *nRobotPerGiocatore* sono numeri, *condizioneFinePartita* è uno fra **primo, primi3, tutti**, *upgradeInizialiSiONo* è **sì** oppure **no**.

**Postcondizioni**: viene creato un oggetto *Partita* **P** con attributi *nMaxGiocatori, nRobotPerGiocatore, condizioneFinePartita* e *upgradeInizialiSiONo* pari ai parametri omonimi dell’operazione, e *avviataSiONo* = **no.**

**P** viene associata al *Server* **S**.

Il *Manager* **M** *crea* **P**.

Viene creato un *Robodromo* **R** con *tipoRobodromo* pari al parametro *robodromo*, e **P** *ha* **R**.

Vengono creati *n* (vedi **regole di business**) *Robot* *disponibili in* **P** con *nome* e *colore* come da **regole di business**.

## avviaPartita( )

**Precondizioni:** Esiste una partita **P** con *avviataSiONo* = **no.** i *partecipanti* di **P** sono almeno 2 e non più di *nMaxGiocatori* di **P**. Sul *Robodromo* di **P** *gareggiano* almeno 2 *Robot.* Ciascun *Giocatore* che *partecipa* a **P** *controlla* al massimo *nRobotPerGiocatore* di **P**.

**Postcondizioni:**

L’attributo *avviataSiONo*  di **P** viene impostato a **sì**.

Se *upgradeInizialiSiONo* in **P** è pari a **sì**, l’attributo *upgrade* di ciascun *RobotInPartita* viene assegnato come da **regole di business.**

## annullaPartita( )

**Precondizioni:** Esiste una partita **P** con *avviataSiONo* = **no.**

**Postcondizioni:** Vengono eliminati: l’oggetto **P**, il suo *Robodromo* (con eventuali *Robot* che *gareggiano* su di esso), il suo *Server*, i *Robot* *disponibili* su **P**, le *Richieste di Partecipazione* *relative* a **P**, e tutte le altre associazioni di **P**.

# Gestire richieste di partecipazione

## accettaRichiesta(richiesta, robot)

**Precondizioni:** Esiste una *Partita* **P** con *partecipanti* in numero inferiore a *nMaxGiocatori*.

*richiesta* è una *Richiesta di Partecipazione relativa a* **P**, *effettuata da* un *Giocatore* **G**, con *parolaChiave* uguale alla *parolaChiave* del *Server* associato a **P**, e con *accettataSiONo* non assegnato.

*robot* è una lista di *Robot* *disponibili in* **P**.

**Postcondizioni:**

Ciascun *Robot* **Rb**nella lista *robot* *gareggia sul Robodromo* **R** associato a **P** con *dockDiPartenza* assegnato come da **regole di business** e **Rb** non è più *disponibile in* **P**.

Il *Giocatore* **G** *partecipa* a**P** e *controlla* ciascun *Robot* nella lista *robot*.

L’attributo *accettataSiONo* di *richiesta* viene impostato a **sì**.

## rifiutaRichiesta(richiesta)

**Precondizioni:** Esiste una *Partita* **P** con *partecipanti* in numero inferiore a *nMaxGiocatori*.

*richiesta* è una *Richiesta di Partecipazione relativa a* **P**, *effettuata da* un *Giocatore* **G**, con *parolaChiave* uguale alla *parolaChiave* del *Server* associato a **P**, e con *accettataSiONo* non assegnato.

**Postcondizioni:**

L’attributo *accettataSiONo* di *richiesta* viene impostato a **no**.

# Allenarsi

## iniziaAllenamento(robodromo)

**Precondizioni:**  *robodromo* è un testo che identifica un tipo di robodromo.

**Postcondizioni:** viene creato un oggetto allenamento **A**. Viene creato un *Robodromo* **R** con *tipoRobodromo* pari al parametro *robodromo*, e **A** *ha* **R**.

*avviataSiONo* = **no**.

*pausaSiONo* = **no**.

## eseguiProgramma()

**Precondizioni:**  è stato creato un allenamento **A** con *avviatoSiONo* = **no** e*pausaSiONo* = **no** e un programma **Pr**.

**Postcondizioni:** l’attributo a*vviatoSiONo* viene impostato a **sì**.

## pausa()

**Precondizioni:** è stato creato un programma **Pr** e un allenamento **A** el’attributo a*vviatoSiNO =* **sì** e*pausaSiONo* = **no**.

**Postcondizioni:** l’attributo *pausaSiONo* viene impostato a **sì**.

## undoEsecuzione(istruzione\_precedente)

**Precondizioni:** è stato creato un programma **Pr** e un allenamento **A** el’attributo a*vviatoSiNO =* **sì** e*pausaSiONo* = **sì**.

**Postcondizioni:** l’esecuzione del programma riparte dal punto in cui viene eseguita *istruzione\_precedente* e *pausaSiONo* viene impostato a**no***.*

## interrompiProgramma()

**Precondizioni:** è stato creato un programma **Pr** e un allenamento **A** el’attributo a*vviatoSiNO =* **sì** e*pausaSiONo* = **sì**.

**Postcondizioni:** Vengono eliminati: l’oggetto **Pr** e le sue associazioni.

# **Scrivere programma**

## **aggiungiIstruzione(istruzione)**

**Precondizioni:** è stato creato un allenamento **A.** i*struzione* è un testo che identifica un tipo di istruzione.

**Postcondizioni:** se non è già stato creato viene creato un oggetto programma **Pr** .

Viene creata un *istruzione* **I** con *tipoIstruzione* pari al parametro *istruzione*, e***Pr*** *ha* ***I****.*

## **rimuoviIstruzione(istruzione)**

**Precondizioni:** è stato creato un allenamento **A** e un programma **Pr**. i*struzione* è un testo che identifica una istruzione.

**Postcondizioni:** è stata rimossa da **Pr** *istruzione*.

## **modificaOrdiniIstruzioni( istruzione1, istruzione2 )**

**Precondizioni:** è stato creato un allenamento **A** e un allenamento **Pr**.i*struzione1 e istruzione2* sono un testo che identifica una istruzione.

**Postcondizioni:** l’*istruzione1* è stata scambiata con *istruzione2*.

# **Giocare**

## **entrareInPartita()**

**Precondizioni:** Esiste una *Partita* **P** con *partecipanti* in numero inferiore a *nMaxGiocatori*.

L’attributo *avviataSiONo*  di **P** è impostato a **sì**.

L’attributo *accettataSiONo* di *richiesta* è impostato a **sì**.

Se *upgradeInizialiSiONo* in **P** è pari a **sì**, l’attributo *upgrade* di ciascun *RobotInPartita* viene assegnato come da **regole di business.**

Il *Giocatore* **G** *partecipa* a**P** e *controlla* ciascun *Robot* nella lista *robot*.

**Postcondizioni:** Il *Giocatore* **G** entra in partita e l’attributo *inPartitaSiONo* viene impostato a **sì**.

## **ritiro()**

**Precondizioni:** Esiste una *Partita* **P** con *partecipanti* in numero inferiore a *nMaxGiocatori*.

L’attributo *avviataSiONo*  di **P** è impostato a **sì**.

L’attributo *accettataSiONo* di *richiesta* è impostato a **sì**.

Se *upgradeInizialiSiONo* in **P** è pari a **sì**, l’attributo *upgrade* di ciascun *RobotInPartita* viene assegnato come da **regole di business.**

Il *Giocatore* **G** *partecipa* a**P** e *controlla* ciascun *Robot* nella lista *robot*.

L’attributo *inPartitaSiONo* di **G** è impostato a **sì**.

**Postcondizioni:** Il *Giocatore* **G** esce dalla partita **P** e l’attributo di **G** *inPartitaSiONo* vieneimpostato a **no**.

# **Programmare**

## **spegniRobot()**

**Precondizioni:** Esiste una *Partita* **P** con *partecipanti* in numero inferiore a *nMaxGiocatori*.

L’attributo *avviataSiONo*  di **P** è impostato a **sì**.

L’attributo *accettataSiONo* di *richiesta* è impostato a **sì**.

Se *upgradeInizialiSiONo* in **P** è pari a **sì**, l’attributo *upgrade* di ciascun *RobotInPartita* viene assegnato come da **regole di business.**

Il *Giocatore* **G** *partecipa* a**P** e *controlla* ciascun *Robot* nella lista *robot*.

L’attributo *inPartitaSiONo* è impostato a **sì**.

L’attributo spentoSi*ONo* è impostato a **no**.

**Postcondizioni:** il *Giocatore* **G** spegne il *robot* e l’attributo *spentoSiONo* viene impostato a **sì**.

## **inserisciSchedaIstruzione(registro)**

**Precondizioni:** Esiste una *Partita* **P** con *partecipanti* in numero inferiore a *nMaxGiocatori*.

L’attributo *avviataSiONo*  di **P** è impostato a **sì**.

L’attributo *accettataSiONo* di *richiesta* è impostato a **sì**.

Se *upgradeInizialiSiONo* in **P** è pari a **sì**, l’attributo *upgrade* di ciascun *RobotInPartita* viene assegnato come da **regole di business.**

Il *Giocatore* **G** *partecipa* a**P** e *controlla* ciascun *Robot* nella lista *robot*.

L’attributo inPartitaSiONo è impostato a **sì**.

**Postcondizioni:** il *Giocatore* **G** inserisce un *Istruzione* **I** dentro un *Registro* **R** se l’attributo *bloccatoSiONo* = **no**.

## **dichiaraFine()**

**Precondizioni:** Esiste una *Partita* **P** con *partecipanti* in numero inferiore a *nMaxGiocatori*.

L’attributo *avviataSiONo*  di **P** è impostato a **sì**.

L’attributo *accettataSiONo* di *richiesta* è impostato a **sì**.

Se *upgradeInizialiSiONo* in **P** è pari a **sì**, l’attributo *upgrade* di ciascun *RobotInPartita* viene assegnato come da **regole di business.**

Il *Giocatore* **G** *partecipa* a**P** e *controlla* ciascun *Robot* nella lista *robot*.

L’attributo inPartitaSiONo è impostato a **sì**.

Il *Giocatore* **G** ha riempito tutte le schede non bloccate.

**Postcondizioni:** il *Giocatore* **G** attende che tutti i giocatori della partita dichiarino la fine della propria programmazione.

# **Eseguire programma**

## **usaUpgrade()**

**Precondizioni:** Esiste una *Partita* **P** con *partecipanti* in numero inferiore a *nMaxGiocatori*.

L’attributo *avviataSiONo*  di **P** è impostato a **sì**.

L’attributo *accettataSiONo* di *richiesta* è impostato a **sì**.

Se *upgradeInizialiSiONo* in **P** è pari a **sì**, l’attributo *upgrade* di ciascun *RobotInPartita* viene assegnato come da **regole di business.**

Il *Giocatore* **G** *partecipa* a**P** e *controlla* ciascun *Robot* nella lista *robot*.

L’attributo inPartitaSiONo di **G** è impostato a **sì**.

Il *Giocatore* **G** *è in posseso di* un *Upgrade* **U.**

**Postcondizioni:** il *Giocatore* **G** consuma l’*Upgrade* **U**.