Inizio SpecificaStatiClasse Playlist

Stato: { Attesa, Esecuzione }

Variabili di stato ausiliarie

player : Player
prossimoBrano : Brano

StatoIniziale:

statoCorrente = Attesa
player = --
prossimoBrano = --

FineSpecifica

InizioSpecificaTransizioniClasse Playlist

Transizione: Attesa ⟶ Esecuzione
play(player1)[playlist non vuota] / playSong{dest: player}(br)

Evento: play(player: Player)

Condizione: [this.contiene non vuoto]

Azione:

pre: nessuna

post: nuovoevento = playSong{mitt = this, dest = player}(br: Brano)
and
this.player = player and
this.prossimobrano = 0 and
<this, br> in contiene and
posizione(contiene(this, br)) = this.prossimobrano

...

FineSpecificaTransizioni

Realizzazione attività

① I/O signals package segnali_io

Possiamo assumere esista SegnaliIO (classe) contenente metodi pubblici statici per le operazioni di input output.

② Attività complesse

Le attività complesse hanno come esecutore Thread (implementano Runnable)

```
import package attività_complesse; import attività_atomiche; // Io

public class AttivitaPrincipale implements Runnable {
   private boolean eseguita = false;
   private TaskExecutor executor = TaskExecutor.getInstance();

   private Tipo1 val1;
   private Tipo2 val2;   } VARIABILI LOCALI

   public AttivitaPrincipale (Tipo1 val1, Tipo2 val2){
       this.val1 = val1;
       this.val2 = val2;
   }
   public synchronized void run (){
       if (eseguita == true) return;
       eseguita = true;
       executor.perform (new AttivitaAtomica1 (params)); // per att. atomiche
       Thread ab = new Thread (AttivitaComplessa1 (params));
       Thread cd = new Thread (new AttivitaComplessa2 (params));
       ab.start()
       cd.start()
       try {
           ab.join();
           cd.join();
       } catch (InterruptedException e){
           System.out.println(e); }
       // per ottenere risultati da att. complesse usare getOgto()
   }
   public synchronized boolean estEseguita (){return eseguita;}
   public synchronized Type1 getResult1(){ if ... RuntimeException("");
                                           else return ...;}
```

(3) Attività atomiche

```
package attivita_atomiche; import application; import a;
public class Task1 implements Task{
  private boolean eseguita = false;
  private Tipo1 val1;
  private Tipo2 val2;   } variabili locali
  public Task1 ( params ) {... }
  public synchronized void esegui() {
   if (eseguita == true) return;
    eseguita= true;
    ~~saldo = conto~~
```

operazioni con i parametri

```
}
public synchronized T1 getResult1() {...}
        ⋮
public synchronized ~~int~~ boolean estEseguita() { return eseguita; }
}
```