Università degli Studi di Roma "La Sapienza"
Facoltà di Ingegneria – Corso di Laurea in Ingegneria Gestionale
Corso di Progettazione del Software
Proff. Toni Mancini e Monica Scannapieco

Progetto **PI.20060908**, passo **P.1**

versione del 11 marzo 2007

Si vuole progettare e realizzare *jTunes*, un programma per la gestione e l'esecuzione di file musicali. Il programma è composto da due moduli: il *modulo di gestione* dei brani e quello di *esecuzione*. Il primo ha il compito di mantenere informazioni circa la libreria di file musicali a disposizione, di consentire la creazione di liste di brani da ascoltare (playlist), e di permettere ai diversi utenti di esprimere il loro gradimento sui singoli brani.

Il secondo modulo si occupa della esecuzione vera e propria di liste di brani. In fase di studio di fattibilità si è deciso di utilizzare per tale modulo un'applicazione software esistente e pertanto non ne è richiesta la progettazione.

Si richiede di svolgere la fase di Progetto del sistema secondo la metodologia vista nel corso. Si consideri come input lo schema concettuale prodotto nei passi A.1 e A.2.

Requisiti

jTunes deve poter mantenere informazioni sui brani musicali presenti sul computer nel quale è installato. L'insieme di tali brani è chiamato *libreria*. In particolare, per ogni brano della libreria, interessa conoscere interpreti, titolo, album (con relativo numero di traccia, ove disponibile), genere musicale e durata. Inoltre, di ogni brano il sistema deve mantenere il nome del file dove è memorizzato. Per semplicità si può assumere che ogni brano appartiene ad un album (di cui interessa il titolo e l'anno di pubblicazione), che ha uno o più interpreti (di cui interessa il nome), e che appartiene ad un unico genere musicale.

jTunes deve essere utilizzabile e personalizzabile da più utenti (ognuno dei quali è caratterizzato dal proprio nome). In particolare *il sistema deve permettere ai singoli utenti di memorizzare* il proprio gradimento per i singoli brani musicali della libreria. Il gradimento è espresso mediante un intero tra 0 e 5.

Inoltre *jTunes* deve anche offrire ai singoli utenti la possibilità di *creare* opportune liste di brani da eseguire (chiamate *playlist*), di cui interessa titolo e durata. In particolare, *jTunes* deve rappresentare e gestire due tipologie di playlist, oltre che *permetterne la creazione* da parte degli utenti: le playlist *semplici* e quelle *smart*:

- Una *playlist semplice* consiste in un elenco di brani (senza ripetizioni) esplicitamente scelti dall'utente. Eseguire una playlist semplice significa eseguire i brani che la compongono. Tuttavia, al momento della creazione di una playlist semplice, l'utente può decidere se l'ordine in cui i brani devono essere eseguiti debba essere quello indotto dal relativo elenco, oppure se questi debbano essere *mescolati* in modo pseudocasuale dal modulo di esecuzione (questa funzionalità è di solito chiamata *shuffling* di una playlist).

- Una *playlist di tipo smart*, invece, si differenzia da una semplice per il fatto che l'utente non specifica manualmente i brani che la compongono, ma indica i criteri di scelta di tali brani: è il sistema stesso che, a partire dai criteri indicati, si occupa di recuperare dalla libreria l'insieme dei brani che li soddisfano. Il vantaggio delle *playlist smart* rispetto a quelle *semplici* è che le modifiche apportate alla libreria (ad es. l'aggiunta o la cancellazione dei brani) si riflettono immediatamente nell'elenco dei brani che le compongono.

  In particolare *jTunes* deve permettere di definire una *playlist smart* a partire da una *soglia minima di gradimento* (o valutazione) g_{min} dei brani e da un *criterio di selezione*: la playlist conterrà tutti i brani della libreria per i quali l'utente o non ha espresso valutazioni, oppure ha espresso una valutazione almeno uguale a g_{min}, e che in più soddisfano il criterio di selezione indicato. I criteri di selezione dei brani di una *playlist smart* sono due: *selezione per interprete* e *selezione per genere*. In dettaglio:

  – Il criterio di selezione *per interprete* viene definito a partire da un insieme di artisti A. Questo selezionerà tutti i brani della libreria che hanno per interprete almeno un artista in A;

  – Il criterio di selezione *per genere* viene invece definito a partire da un insieme di generi G. Questo selezionerà tutti i brani della libreria che appartengono ai generi menzionati in G.

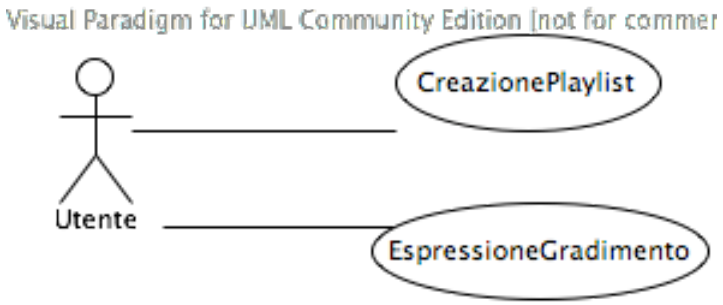
  Di una *playlist smart* il modulo di esecuzione deve poter ricavare l'elenco dei brani che ne fanno parte.

  Si osservi tuttavia che, al contrario delle playlist semplici, i brani delle playlist smart vengono sempre mescolati dal modulo di esecuzione, quindi l'ordine con il quale questi vengono selezionati dal relativo criterio di selezione non ha alcuna importanza.

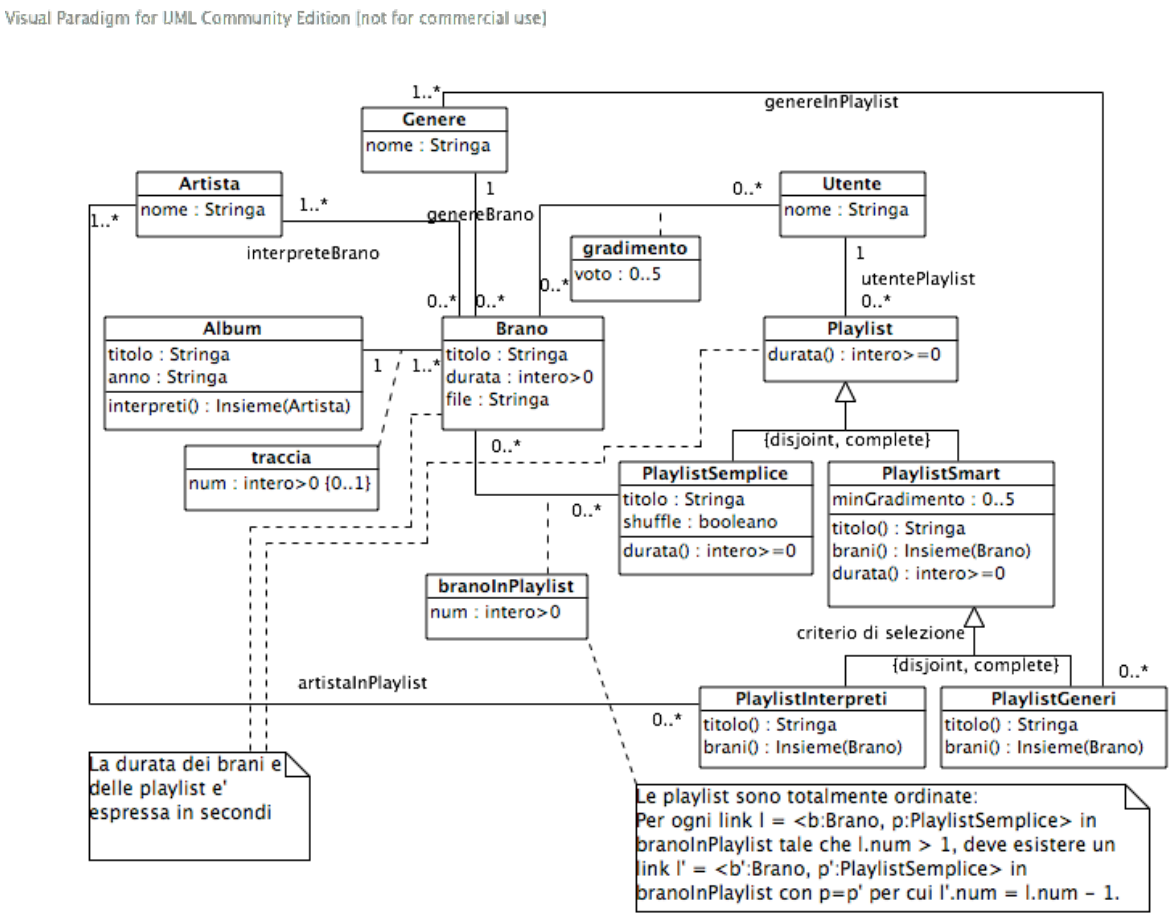
A differenza delle *playlist semplici*, il titolo di una *playlist smart* viene calcolato automaticamente come segue: se il criterio di selezione è *per interprete*, questo sarà "*Brani degli interpreti...*" seguito dall'elenco degli interpreti richiesti separato da ",". Al contrario, se il criterio è *per genere*, il titolo sarà "*Brani di genere...*" seguito dall'elenco dei generi richiesti, separato da ",".

1 Fase di Analisi

1.1 Diagramma degli Use Case



1.2 Diagramma delle classi UML



1.3 Specifica dei tipi di dato

Nessuna.

1.4 Specifica degli use case

```
SpecificaUseCase CreazionePlaylist
  creaPlaylistSemplice(u:Utente, titolo:Stringa, shuffling:booleano) : PlaylistSemplice
    pre: nessuna
    post: result e' un nuovo oggetto della classe PlaylistSemplice con:
```

```
      - result.titolo = titolo;
      - result.shuffling = shuffling.

      Inoltre, viene creato il link <result, u> di associazione utentePlaylist.

aggiungiBrano(u:Utente, p:PlaylistSemplice, b:Brano)
  pre: p.utentePlaylist.Utente = u;
  post: viene creato un nuovo link l=<b,p> di associazione branoInPlaylist.
    Il valore di l.num e' dato da 1 (uno) nel caso non esistano altri link in
    p.branoInPlaylist (ovvero la playlist era vuota), altrimenti e' pari a:
```

$$\texttt{l.num} = 1 + \max_{\texttt{l'} \in \texttt{p.branoInPlaylist}} \texttt{l'.num}.$$

```
eliminaBrano(u:Utente, p:PlaylistSemplice, b:Brano)
  pre: p.utentePlaylist.Utente = u;
  post: viene eliminato, se esiste, il link l=<b,p> di associazione branoInPlaylist.
      In questo caso, per ogni altro link l' in p.branoInPlaylist (ovvero
      ogni altro link di associazione branoInPlaylist in cui p e' coinvolto)
      per cui l'attributo num ha un valore superiore a l.num, tale valore
      viene decrementato di uno.
      Formalmente:

         per ogni l' in p.branoInPlaylist tale che pre(l'.num) > l.num,
             l'.num = pre(l'.num) - 1.

creaPlaylistSmartPerGenere(u:Utente, minGrad:0..5, generi:Insieme(Genere)) :
                                                                  PlaylistGeneri
  pre: nessuna
  post: result e' un nuovo oggetto della classe PlaylistGeneri con
     result.minGradimento = minGrad.

     Inoltre, viene creato il link <result, u> di associazione utentePlaylist.
     Infine, per ogni oggetto g nell'insieme generi, viene creato un link
     <result, g> di associazione genereInPlaylist.

creaPlaylistSmartPerInterprete(u:Utente, minGrad:0..5, artisti:Insieme(Artista)) :
                                                              PlaylistInterprete
  pre: nessuna
  post: result e' un nuovo oggetto della classe PlaylistInterprete con
```

```
       result.minGradimento = minGrad.

       Inoltre, viene creato il link <result, u> di associazione utentePlaylist.
       Infine, per ogni oggetto 'a' nell'insieme artisti, viene creato un link
       <result, a> di associazione artistaInPlaylist.

FineSpecifica


SpecificaUseCase EspressioneGradimento
  esprimiGradimento(u:Utente, b:Brano, g:0..5)
    pre: nessuna
    post: Se non esiste alcun link l=<u,b> di associazione gradimento,
       questo viene creato.
       In ogni caso, all'attributo l.voto viene assegnato il valore 'g'.
```

1.5 Specifica delle classi

La classe Playlist

```
SpecificaClasse Playlist
  durata(): intero >= 0
    pre: nessuna
    post: il valore di result dipende dalla classe piu' specifica di this.
FineSpecifica
```

La classe PlaylistSemplice

```
SpecificaClasse PlaylistSemplice
  durata(): intero >= 0
    pre: nessuna
    post: result e' pari a:
```

$$\Sigma_{l \in \texttt{this.branoInPlaylist}} \texttt{l.Brano.durata}$$

```
FineSpecifica
```

La classe PlaylistSmart

```
SpecificaClasse PlaylistSmart
  brani(): Insieme(Brano)
    pre: nessuna
    post: il valore di result dipende dalla classe piu' specifica di this.

  durata(): intero >= 0
    pre: nessuna
    post: result e' pari a:
```

$$\Sigma_{\texttt{b}\in\texttt{this.brani()}}\texttt{b.durata}$$

```
  titolo(): Stringa
    pre: nessuna
    post: il valore di result dipende dalla classe piu' specifica di this.
FineSpecifica
```

La classe PlaylistInterpreti

```
SpecificaClasse PlaylistInterpreti
  brani(): Insieme(Brano)
    pre: nessuna
    post: Detto A l'insieme degli oggetti di classe Artista collegati a this
      da link di associazione artistaInPlaylist, ovvero:
```

$$\texttt{A} = \{\texttt{a} \in \texttt{Artista} \mid \langle \texttt{a}, \texttt{this}\rangle \in \texttt{artistaInPlaylist}\}$$

```
    result e' pari all'insieme dei brani che hanno per interprete almeno un
    artista in A per cui l'utente this.utentePlaylist.Utente o non ha espresso
    valutazioni, oppure ha espresso una valutazione almeno pari a
    this.minGradimento.

    Formalmente, detto B l'insieme dei brani che soddisfano il criterio di
    selezione:
```

$$\texttt{B} = \big\{\texttt{b} \in \texttt{Brano} \mid \{\texttt{a} \in \texttt{Artista} \mid \langle \texttt{a}, \texttt{b}\rangle \in \texttt{interpreteBrano}\} \cap A \neq \emptyset \ \big\}$$

result e' pari al sottoinsieme dei brani in B che soddisfano il requisito
di gradimento:

$$\texttt{result} = \{\texttt{b} \in \texttt{B} \mid \texttt{non esiste il link l=}\langle \texttt{b}, \texttt{this.utentePlaylist.Utente}\rangle \in \texttt{gradimento oppure tale link l esiste e l.voto} > \texttt{this.minGradimento}\}$$

titolo(): Stringa
pre: nessuna
post: Detto A l'insieme degli oggetti di classe Artista collegati a this
da link di associazione artistaInPlaylist, ovvero:

$$\texttt{A} = \{\texttt{a} \in \texttt{Artista} \mid \langle \texttt{a}, \texttt{this}\rangle \in \texttt{artistaInPlaylist}\}$$

result e' pari alla stringa "Brani degli interpreti " concatenata
con l'elenco dei valori dell'attributo 'nome' dei componenti di
A separati da ",".

FineSpecifica

La classe PlaylistGeneri

SpecificaClasse PlaylistGeneri
brani(): Insieme(Brano)
pre: nessuna
post: Detto G l'insieme degli oggetti di classe Genere collegati a this
da link di associazione genereInPlaylist, ovvero:

$$\texttt{G} = \{\texttt{g} \in \texttt{Genere} \mid \langle \texttt{g}, \texttt{this}\rangle \in \texttt{genereInPlaylist}\}$$

result e' pari all'insieme dei brani che appartengono ad uno dei generi in G
per cui l'utente this.utentePlaylist.Utente o non ha espresso
valutazioni, oppure ha espresso una valutazione almeno pari a
this.minGradimento.

Formalmente, detto B l'insieme dei brani che soddisfano il criterio di
selezione:

$$\texttt{B} = \big\{\texttt{b} \in \texttt{Brano} \mid \texttt{b.genereBrano.Genere} \in G\big\}$$

result e' pari al sottoinsieme dei brani in B che soddisfano il requisito di gradimento:

$$\texttt{result} = \{\texttt{b} \in \texttt{B} \mid \texttt{non esiste il link l=}\langle \texttt{b}, \texttt{this.utentePlaylist.Utente}\rangle \in \texttt{gradimento oppure tale link l esiste e l.voto} > \texttt{this.minGradimento}\}$$

titolo(): Stringa
pre: nessuna
post: Detto G l'insieme degli oggetti di classe Genere collegati a this da link di associazione genereInPlaylist, ovvero:

$$\texttt{G} = \{\texttt{g} \in \texttt{Genere} \mid \langle \texttt{g}, \texttt{this}\rangle \in \texttt{genereInPlaylist}\}$$

result e' pari alla stringa "Brani di genere " concatenata con l'elenco dei valori dell'attributo 'nome' dei componenti di G separati da ",".

FineSpecifica

La classe Album

SpecificaClasse Album
interpreti() : Insieme(Artista)
pre: nessuna
post: Detto B l'insieme dei brani nell'album this, ovvero

$$\texttt{B} = \{\texttt{b} \in \texttt{Brano} \mid \langle \texttt{this}, \texttt{b}\rangle \in \texttt{traccia}\}$$

result e' pari a:

$$\texttt{result} = \bigcup_{\texttt{b} \in \texttt{B}} \{\texttt{a} \in \texttt{Artista} \mid \langle \texttt{a}, \texttt{b}\rangle \in \mathit{interpreteBrano}\}.$$

2 Fase di Progetto

2.1 Corrispondenza tra tipi UML e tipi Java

Tipo UML	Tipo Java	Note
Stringa	String	–
intero $>/\geq$ 0, 0..5	int	Verifica ammissibilità sul lato server
intero $>$ 0{0..1}	Integer	Verifica amm. lato server, può essere null
Insieme(...)	HashSet< ... >	Implementa l'interfaccia Set< ... >

2.2 Ristrutturazione delle gerarchie is-a

Non necessaria, le gerarchie sono tutte disjoint. Inoltre le classi *Playlist* e *PlaylistSmart* saranno dichiarate *abstract*, dato che sono radici di gerarchie complete.

2.3 Specifica realizzativa delle strutture dati

Nessuna struttura dati da progettare.

2.4 Specifica realizzativa delle classi

La classe Playlist

```
SpecificaClasse Playlist (abstract)
  +abstract durata(): int;
FineSpecifica
```

La classe PlaylistSemplice

```
SpecificaClasse PlaylistSemplice
  +durata(): int
    pre: nessuna
    algoritmo: ovvio, cf. specifica concettuale.
FineSpecifica
```

La classe PlaylistSmart

```
SpecificaClasse PlaylistSmart (abstract)
  +abstract brani(): Set;

  +durata(): int
    pre: nessuna
    algoritmo: ovvio, cf. specifica concettuale

  +abstract titolo(): String;
FineSpecifica
```

La classe PlaylistInterpreti

```
SpecificaClasse PlaylistInterpreti
  +brani(): Set
    pre: nessuna
    algoritmo:
       A = insieme vuoto di oggetti di classe Artista;
       Per ogni link l in this.artistaInPlaylist {
          A = A unione l.Artista;
       }
       result = insieme vuoto di oggetti di classe Brano;
       per ogni 'a' in A {
           per ogni link l in a.interpreteBrano {
               sia b = l.Brano;
               se b.vincoloGradimentoSoddisfatto(
                   this.utentePlaylist.Utente, this.minGradimento) = true
                      allora result = result unione b;
           }
       }
       ritorna result;
       // Attenzione: sebbene l'algorimo proposto sia il piu' semplice,
       // ne esistono di piu' efficienti! (utile esercizio)

  +titolo(): Stringa
    pre: nessuna
    algoritmo:
       A = insieme vuoto di oggetti di classe Artista;
```

```
       Per ogni link l in this.artistaInPlaylist {
          A = A unione l.Artista;
       }
       artisti = stringa ottenuta concatenando i valori dell'attributo 'nome'
       degli elementi di A separati da ",".
       /* potremmo essere piu' specifici, utilizzando un ciclo ed effettuando
          esplicitamente le concatenazione:
          artisti = "";
          per ogni a in A {
             artisti = artisti + a.nome;
             se A ha altri elementi ancora da scandire, allora artisti = artisti + ", ";
          }
        */

      return "Brani degli interpreti " + artisti;
FineSpecifica
```

La classe PlaylistGeneri

```
SpecificaClasse PlaylistGeneri
  +brani(): Set
    pre: nessuna
    algoritmo:
       G = insieme vuoto di oggetti di classe Genere;
       Per ogni link l in this.genereInPlaylist {
          G = G unione l.Genere;
       }
       result = insieme vuoto di oggetti di classe Brano;
       per ogni 'g' in G {
           per ogni link l in g.genereBrano {
               sia b = l.Brano;
               se b.vincoloGradimentoSoddisfatto(
                     this.utentePlaylist.Utente, this.minGradimento) = true
                         allora result = result unione b;
           }
       }
       ritorna result;

  +titolo(): Stringa
```

```
    pre: nessuna
    algoritmo:
       G = insieme vuoto di oggetti di classe Genere;
       Per ogni link l in this.genereInPlaylist {
          G = G unione l.Genere;
       }
       generi = stringa ottenuta concatenando i valori dell'attributo 'nome'
       degli elementi di G separati da ",".
       // potremmo essere piu' specifici, utilizzando un ciclo ed effettuando
       // esplicitamente le concatenazione...

      return "Brani di genere " + generi;
FineSpecifica


SpecificaClasse Brano
    +vincoloGradimentoSoddisfatto(u:Utente, g:int): boolean {
         // Operazione ausiliaria, utilizzata da PlaylistSmart.brani();
         per ogni link l in b.gradimento {
            se l.Utente = u {
                // l'utente u ha valutato il brano b
                se l.voto >= g allora return true;
                altrimenti return false;
            }
         }
         // l'utente u non ha espresso gradimento per il brano b
         return true;
     }
FineSpecifica
```

2.5 Specifica realizzativa degli use-case

```
SpecificaUseCase CreazionePlaylist
  +creaPlaylistSemplice(u:Utente, titolo:String, shuffling:boolean) : PlaylistSemplice
    pre: nessuna
    algoritmo: ovvio, cf. specifica concettuale

  +aggiungiBrano(u:Utente, p:PlaylistSemplice, b:Brano)
    pre: p.utentePlaylist.Utente = u;
```

```
    algoritmo: ovvio, cf. specifica concettuale

  +eliminaBrano(u:Utente, p:PlaylistSemplice, b:Brano)
    pre: p.utentePlaylist.Utente = u;
    algoritmo: ovvio, cf. specifica concettuale

  +creaPlaylistSmartPerGenere(u:Utente, minGrad:int, generi:Set<Genere>) : PlaylistGeneri
    pre: 0 <= minGrad <= 5
    algoritmo: ovvio, cf. specifica concettuale

  creaPlaylistSmartPerInterprete(u:Utente, minGrad:int, artisti:Set<Artista>) : PlaylistInte
    pre: 0 <= minGrad <= 5
    algoritmo: ovvio, cf. specifica concettuale

FineSpecifica


SpecificaUseCase EspressioneGradimento
  esprimiGradimento(u:Utente, b:Brano, g:int)
    pre: 0 <= g <= 5
    post: ovvio, cf. specifica concettuale
Finespecifica
```

2.6 Progetto dei diagrammi degli stati

Non sono stati definiti diagrammi degli stati in fase di Analisi.

2.7 Responsabilità sulle associazioni

Dai requisiti, dalla specifica delle operazioni di classi e di use case, e delle molteplicità nel diagramma delle classi emerge che:

Associazione	Classe	Ha resp?	Motivo
traccia	Brano	SI	1..1
	Album	SI	1..*
interpreteBrano	Artista	SI	PlaylistInterpreti.brani()
	Brano	SI	1..*
genereBrano	Genere	SI	PlaylistGeneri.brani()
	Brano	SI	1..1
gradimento	Brano	SI	Brano. vincoloGradimentoSoddisfatto()
	Utente	NO	–
utentePlaylist	Utente	SI	requisiti (cf. nota 1)
	Playlist	SI	vincolo 1..1
branoInPlaylist	Brano	NO	–
	PlaylistSemplice	SI	requisiti (necessario al modulo di esecuzione)
artistaInPlaylist	Artista	NO	–
	PlaylistInterpreti	SI	1..*
genereInPlaylist	Genere	NO	–
	PlaylistGeneri	SI	1..*

(1) In realtà i requisiti non la richiedono esplicitamente: tuttavia è ragionevole ritenere che il sistema debba permettere, dato un utente, di ricavare le sue playlist. Per questo motivo prevediamo responsabilità multipla.

2.8 Vincoli sull'evoluzione delle proprietà mutabili

Per esercizio.

2.9 Diagramma delle classi realizzativo

Per esercizio.