Esempio: L'EURO-CONVERTITORE (1)

Scopo

Realizzare un sistema software per la conversione di euro in lire.

In particolare, si vuole, definire un euro-covertitore e, dati due oggetti di tipo euro convertitore inizializzati a 100 euro, si vuole ad uno aggiungere 10 euro, all'altro sottrarre 10 euro e poi stampare di entrambi lo stato corrente in lire.

Esempio: L'EURO-CONVERTITORE (1)

Scopo

Definire un ADT per Euro-Convertitore

Specifica

- Componente caratterizzato in ogni istante da un accumulatore che contiene la quantità corrente di Euro da convertire
- Componente a cui si accede tramite le operazioni di:
  - **add**: Somma Euro alla quantità in accumulatore
  - **sub**: Toglie Euro alla quantità in accumulatore
  - **convert**: restituisce il corrispondente valore in Lire della quantità di Euro in accumulatore
  - **currentValue**: restituisce l'attuale valore dell'accumulatore

Esercitazione n°1

```
public class EuroConverter{
  private double i=0;      //ACCUMULATORE
  public EuroConverter(double j) {i=j;}
  public void add(double j) {i=i+j;} //Somma i di
  j Euro
  public void sub(double j){i=i-j;}  //
  Sottrazione...
  public double currentValue(){return i;}
  public double convert() {return
  Math.rint(i*1936.27);}
 }
```

La classe main

```
public class Main {
  public static void main(String[] args) {
    EuroConverter uno = new EuroConverter (100.00);
    EuroConverter due = new EuroConverter (100.00);
    uno.add(10.00);
    due.sub(10.00);
    System.out.println("Uno: Importo Lire " +
  uno.convert());
    System.out.println("Due: Importo Lire " +
  due.convert());}
}
```

Istruzioni di Selezione

```
if (condizione)          if (condizione){
   istruzione1;             // Istruzioni
else istruzione2;        } else istruzione2;
```

- *Condizione*: qualunque espressione che ritorni un valore boolean (true o false)

Se *condizione* è **true** viene eseguita *istruzione1*, se **false** *istruzione2*. In nessun caso entrambe le istruzioni (o blocchi di istruzioni).

- Parte *else* opzionale
- Possibilità di più *if* annidati

Istruzioni di Iterazione (1)

```
for (inizializzazione; condizione; iterazione) {
   // istruzioni
}
```

- *Inizializzazione*: espressione che imposta il valore della/e variabile/i di controllo del ciclo
- *Condizione*: qualunque espressione che ritorni un valore boolean (true o false)

Se *condizione* è **true** viene eseguito il corpo del ciclo, se **false** il ciclo termina

- *Iterazione*: espressione che incrementa o riduce alla fine di ogni iterazione il valore della/e variabile/i di controllo del ciclo

ES:

```
for (int a=0; a<10; a++)
  System.out.println(" Contantore: " + a);
```

Istruzioni di Iterazione (2)

```
while (condizione) {        do {
                              // corpo del ciclo }
}                           } while (condizione);
```

- *Condizione*: qualunque espressione che ritorni un valore boolean (true o false)

Viene eseguito il corpo del ciclo finchè (tante volte quante) *condizione* è **true**. Quando *condizione* diventa **false** il controllo passa alla riga di codice immediatamente sucessiva al ciclo.

- Nel ciclo ***do/while*** il corpo con le istruzioni viene eseguito almeno una volta, nel ciclo ***while*** ciò non è sempre vero.

ES:

```
int a = 0;
while (a<10){
  System.out.println(" Contantore: " + a); a++;
}
```

Ripasso …..

…..
Counter c;

c=null;

c.inc(); *NullPointerException!!!!!*

C= new Counter()

```
public class Counter{
  private int i=0;
  public  int sp;
  public Counter(int j) {i=j;}
  public Counter(int j, int x) {i=j;
                                sp=x;}

  public void inc () {i=i+sp;}
  public void dec(){i=i-sp;}
  public int currentValue(){return i;}
}
```

C'è information hiding?

Quanti costruttori?

La classe main

```
public class Main {
  public static void main(String[] args) {
    Counter uno = new Counter (5);
    Counter due = new Counter (5,2);
    uno.inc();
    uno.i=8;
    uno.sp=5;
    due.dec();
    System.out.println("Uno: " + uno.currentValue());
}
```

Un esempio completo: Torneo

FASE 1: Definizione dei Requisiti

Si progetti un sistema software per la gestione automatizzata di tornei di tiro al bersaglio. Ogni torneo prevede la partecipazione da un minimo di due ad un massimo di quattro giocatori. In particolare, il sistema software deve permettere:

• ai giocatori di registrarsi al torneo;

• ai giocatori di tirare al bersaglio.

• di gestire il punteggio (es. ad ogni tiro deve essere incrementato il punteggio relativo al giocatore che ha messo a segno il bersaglio)

•di premiare il vincitore del torneo

Per ogni torneo il sistema deve tener traccia dei giocatori iscritti, della sede del torneo e del punteggio. Per ogni giocatore il sistema deve mantenere nome, numero di tornei vinti e numero di tornei persi.

FASE 1: Definizione dei Requisiti

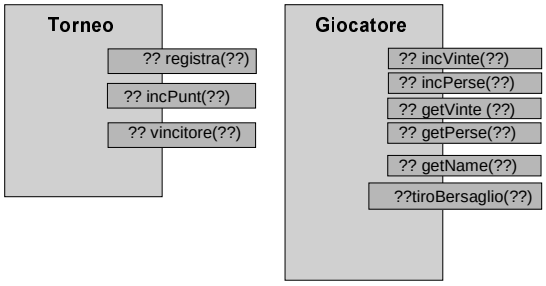
Il sistema deve poi simulare un torneo tra Bart e Liza Simpson a Roma. In particolare, Bart e Liza possono cominciare a gareggiare solo una volta terminata la registrazione al torneo. La gara prevede che Bart e Liza possano tirare al bersaglio per 21 volte. Ad ogni tiro il punteggio deve essere aggiornato e al termine deve essere dichiarato il vincitore. Il sistema deve poi incrementare il numero dei tornei vinti del vincitore, quello dei tornei persi del perdente.

Al termine del torneo a Roma Bart e Liza Simpson si iscrivono ad un altro torneo a Milano. In particolare, Bart e Liza possono cominciare a gareggiare solo una volta terminata la registrazione al torneo. La gara prevede che Bart e Liza possano tirare al bersaglio per 40 volte. Ad ogni tiro il punteggio deve essere aggiornato e al termine deve essere dichiarato il vincitore. Il sistema deve poi incrementare il numero dei tornei vinti del vincitore, quello dei tornei persi del perdente.

FASE 2: Identificazione delle Entità

Si progetti un sistema software per la gestione automatizzata di tornei di tiro al bersaglio. Ogni torneo prevede la partecipazione da un minimo di due ad un massimo di quattro giocatori.

FASE 3: Identificazione della Interfaccia

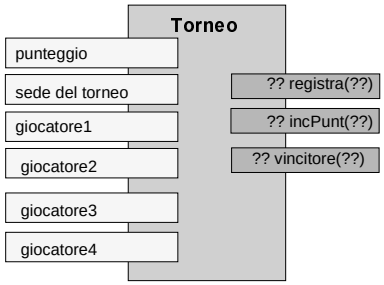


FASE 4: Identificazione dei Dati

Per ogni torneo il sistema deve tener traccia dei giocatori iscritti, della sede del torneo e del punteggio. Per ogni giocatore il sistema deve mantenere nome, numero di tornei vinti e numero di tornei persi.

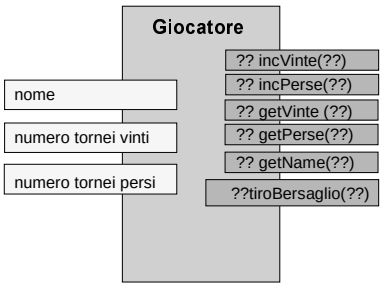
FASE 4: Identificazione dei Dati

Per ogni torneo il sistema deve tener traccia dei giocatori iscritti, della sede del torneo e del punteggio. Per ogni giocatore il sistema deve mantenere nome/cognome, numero di tornei vinti e numero di tornei persi.



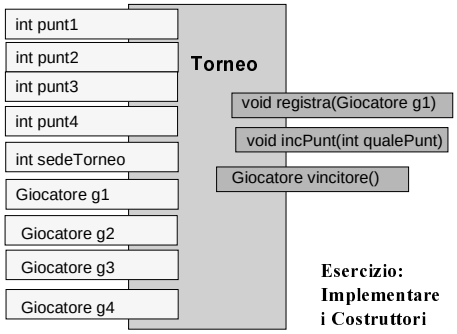
FASE 4: Identificazione dei Dati

Per ogni torneo il sistema deve tener traccia dei giocatori iscritti, della sede del torneo e del tabellone. Per ogni giocatore il sistema deve mantenere nome, numero di tornei vinti e numero di tornei persi.



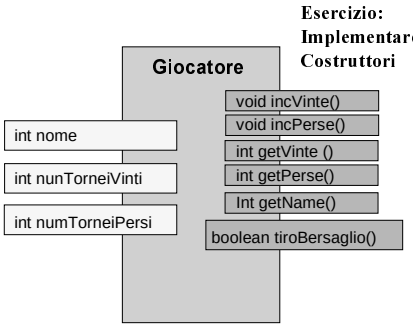
FASE 5: Implementazione…

Scelte implementative: nomi come numeri interi; tabellone come insieme di 4 interi, sede del torneo come intero



FASE 5: Implementazione…

Scelte implementative: nomi come numeri interi



FASE 3: Implementazione dell'applicazione

```
public class Main
{\\convenz.: Bart = 0; Liza = 1; Roma=3; Milano =4
  public static void main (String args[])
  {….
   Giocatore gioc1 = new Giocatore(0);
   Giocatore gioc2 = new Giocatore(1);
   int sede=3;
   Torneo match1 = new Torneo (gioc1, gioc2, sede);
   for (i=0; i<21; i++) // .…21 tiri
   { boolean t = gioc1.tiroBersaglio();
     boolean v = gioc2.tiroBersaglio();
     if (t= = true) match1.incPunt(1);
     else match1.incPunt(2);
   }
   Giocatore vinc = match1.vincitore();
   if (vinc.getName() == 0) gioc1.incVinte();
   else gioc2.incVinte()
              ……. . .
  }
}
```