Università di Ferrara
Dipartimento di Ingegneria

Esercitazioni di

FONDAMENTI DI INFORMATICA MODULO B

ESERCITAZIONE 1 JAVA

Tutor

Arnaud Nguembang Fadja: ngmrnd@unife.it
Damiano Azzolini: damiano.azzolini@student.unife.it

Esercizio su Oggetti Composti

Gli oggetti composti **consentono**:

- di ***aggregare*** componenti complessi a partire da componenti più semplici già disponibili
- di **mantenere** l'unità del componente, assicurandone la protezione e l'incapsulamento

Possiamo anche usarli per:

- ***specializzare*** *componenti* già esistenti
- ***aggiungere*** *loro nuovi dati* o *metodi*

Ad esempio, un contatore *con decremento*

Progettazione Incrementale

Spesso si incontrano problemi che richiedono **componenti simili** *ad altri già disponibili,* **ma non identici.**

Altre volte, **l'evoluzione dei requisiti** comporta una corrispondente **modifica dei componenti**:

- necessità di nuovi dati e/o nuovi comportamenti
- necessità di modificare il comportamento di metodi già presenti

Come fare per non dover rifare tutto da capo?

Progettazione Incrementale - Approcci

- Ricopiare manualmente il codice della classe esistente e cambiare quel che va cambiato

- **Creare un oggetto composto** *(e usare delega)*
  - che incapsuli il componente esistente...
  - … gli "inoltri" le operazioni già previste…
  - … e crei, sopra di esso, le nuove operazioni richieste (eventualmente definendo nuovi dati)
  - **sempre che ciò sia possibile!**

- **Specializzare** (per ereditarietà) la classe Counter (lo vedrete …)

Progettazione Incrementale per Delega

Creare un oggetto composto:

- che incapsuli il componente esistente...
- … gli "inoltri" le operazioni già previste…
- … e crei, sopra di esso, le nuove operazioni richieste (eventualmente definendo nuovi dati)
- sempre che ciò sia possibile

Progettazione Incrementale per Delega

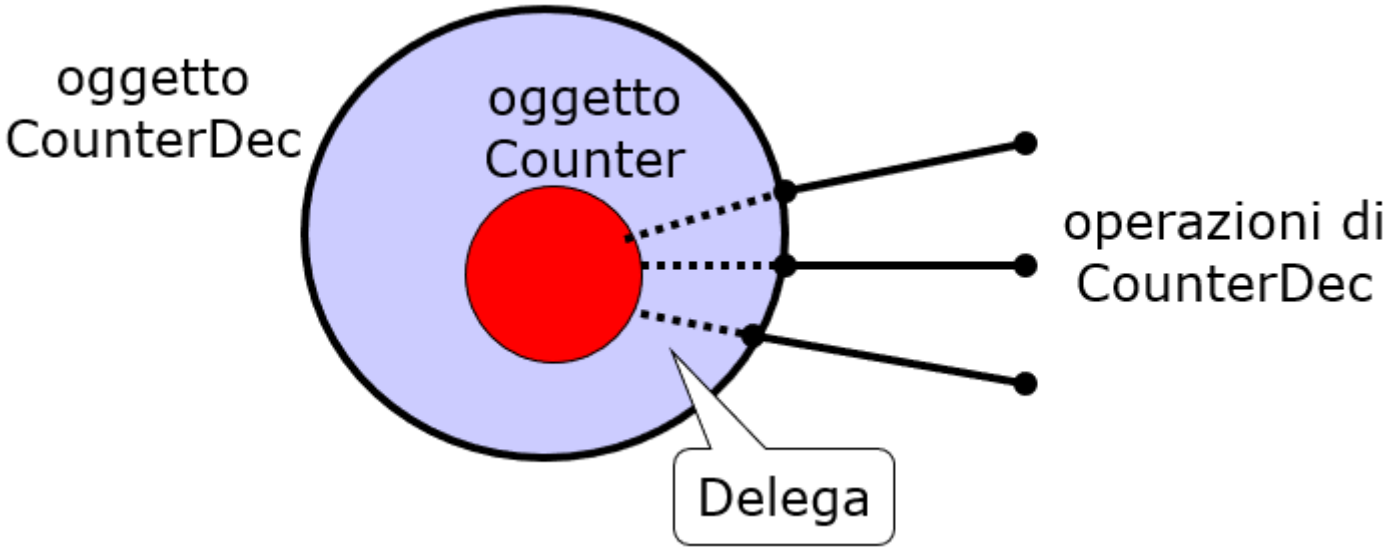
Dal contatore (solo in avanti) …

```
public class Counter {
      private int val;
      public Counter() { val = 1; }
      public Counter(int v) { val = v; }
      public void reset() { val = 0; }
      public void inc()   { val++; }
      public int getValue() { return val;}
}
```

Progettazione Incrementale per Delega

… al *contatore avanti/indietro (con decremento)*

- Concettualmente, ogni oggetto **CounterDec** **ingloba** un oggetto **Counter** al suo interno
- Ogni operazione richiesta a **CounterDec** viene ***delegata*** all'oggetto **Counter** interno



Progettazione Incrementale per Delega

- Aggiungere alla classe **Counter** anche un opportuno metodo **ToString** e un metodo **equals**
- Definire poi l'ADT **CounterDec** che inglobi un oggetto c istanza di **Counter** al suo interno
- Ogni istanza di **CounterDec** sarà un oggetto **composto**
- Nel main definire un oggetto cd istanza di **CounterDec** che ne azzeri il valore (reset), lo incrementi due volte, lo stampi a video, lo decrementi e ne ristampi il valore a video

Progettazione Incrementale per Delega

… al contatore avanti/indietro (con decremento)

```
public class CounterDec {
    private Counter c;
    public CounterDec() { c = new Counter(); }
    public CounterDec(int v){ c = new Counter(v); }
    public void reset() { c.reset(); }
    public void inc()   { c.inc(); }
    public int getValue() { return c.getValue(); }
    public void dec() { ... }
}
```

Progettazione Incrementale per Delega

Il metodo `dec()` deve:

- recuperare il valore attuale V del contatore
- resettare a zero il contatore
- riportarlo, tramite incrementi, al valore V' = V-1

```
public void dec() {
      int v = c.getValue(); c.reset();
      for (int i=0; i<v-1; i++) c.inc();
}
```

Progettazione Incrementale per Delega

- Poiché i campi privati non sono accessibili, bisogna riscrivere anche tutti i metodi che concettualmente rimangono uguali, procedendo per delega (delegation).

- Non è detto che le operazioni già disponibili consentano di ottenere qualsiasi nuova funzionalità si renda necessaria (potrebbe essere necessario accedere ai dati privati).

- Occorre poter riusare le classi esistenti in modo più flessibile.

Esercizio su Stringhe

- Si leggano N stringhe da riga di comando.
- Per ogni stringa si sostituiscano tutte le occorrenze di `old_char` con `new_char` (variabili inizializzate nel codice).
  - Utilizzare la classe `StringBuffer` (e relativi metodi)
    - Stampare l'ID dell'istanza ad ogni modifica dell'oggetto
  - Utilizzare la classe `String` (e relativi metodi)
    - Stampare l'ID dell'istanza ad ogni modifica della stringa

Esercizio su Stringhe

Domande:

- Come si stampa l'ID di un'istanza?
- Quali sono i metodi necessari per sostituire un carattere in un'istanza della classe:
  - `StringBuffer`
  - `String`