Programmazione a Oggetti
Modulo B

Lezione 7

Dott. Alessandro Roncato

24/02/2014

Riassunto

Esercizi d'esame

Esercizio Appello 7/1/14

Data la seguente classe:

```
public class BinaryTree<T>{
  T value;
  BinaryTree<T> left = null;
  BinaryTree<T> right= null;
  public BinaryTree(T t){
    value=t;
  }
  public void visit(){
    System.out.println(value.toString());
    if (left!=null)
      left.visit();
    if (rigth!=null)
      rigth.visit();
  }
}
```

Applicare il pattern Null Object.

Soluzione

```
public class NullBinaryTree extends BinaryTree{
  public NullBinaryTree(){super(null); }
  public void visit(){ }
}
public class BinaryTree<T>{
  T value;
  BinaryTree<T> left = new NullBinaryTree();
  BinaryTree<T> right= new NullBinaryTree();
  public BinaryTree(T t){
    value=t;
  }
  public void visit(){
    System.out.println(value.toString());
    left.visit();
    rigth.visit();
  }
}
```

Soluzione con Singleton

```
public class NullBinaryTree extends BinaryTree{
  private NullBinaryTree(){super(null);}
  public void visit(){}
  public static NullBinaryTree singleton(){return ref;}
  private static NullBinaryTree ref = new NullBinaryTree();
}
public class BinaryTree<T>{
  T value;
  BinaryTree<T> left = NullBinaryTree.singleton();
  BinaryTree<T> right= NullBinaryTree.singleton();
  public BinaryTree(T t){
    value=t;
  }
  public void visit(){
    System.out.println(value.toString());
    left.visit();
    rigth.visit();
  }
}
```

Soluzione con Interfaccia

```
public interface IBinaryTree  {
  void visit();
}
public class NullBinaryTree implements IBinaryTree{
  public NullBinaryTree(){}
  public void visit(){}
}
public class BinaryTree<T> implements IBinaryTree{
  T value;
  IBinaryTree<T> left = new NullBinaryTree()
  IBinaryTree<T> right= new NullBinaryTree();
  public BinaryTree(T t){
    value=t;
  }
  public void visit(){
    System.out.println(value.toString());
    left.visit();
    rigth.visit();
  }
}
```

Esercizio Appello 4/9/13 …

Siano date la seguenti classi.

```
public class SegreteriaStudenti{
   static Set<CorsoLaurea> corsi;
   static Map<Integer,Studente> studenti;
   static String indirizzo;
   static int matricola=0;
   ...
   public static int getNewMatricola(){
      return ++matricola;
   }
}
```

… Esercizio Appello 4/9/13

…continua

```
public class Studente {
   int matricola;
   String nome;
   ...
   public Studente(String nome,...){
   matricola=SegreteriaStudenti4.getNewMatricola();
   ...
   }
}
```

Applicare il pattern Singleton.

Esercizio Appello 7/1/14

Che pattern sono stati usati nel seguente codice?

```
public class Cliente {
   ...
   public Ordine creaOrdine(...){
      Ordine o = new Ordine();
      ...
      return o;
   }
}
```

Esercizio

Che pattern sono stati usati nel seguente codice?

```
public class OrdineFactory {
   ...
   public Ordine creaOrdine(...){
     Ordine o = new Ordine();
     ...
     return o;
   }
}
```

Regole Progetto

Progetto è opzionale.

Per chi sceglie di farlo, il progetto verrà valutato da 0 a 3/30 che verranno aggiunti alla media dei voti dei 2 moduli (modulo A e modulo B) per chi avrà tale media maggiore di 18. Per gli studenti ammessi all'orale, verrà valutato assieme all'orale.

Da consegnare una settimana prima della prima sessione scritta.

Allegare una documentazione con i diagrammi UML e una breve descrizione delle scelte fatte.

Progetto

Progettare e sviluppare un'applicazione per la gestione di un magazzino di una azienda assemblatrice di computer.

L'applicazione deve gestire il carico (= articoli in arrivo) e lo scarico (= articoli in uscita) dal magazzino.

Inoltre, si dovranno gestire le composizioni dei prodotto: esempio la vendita di un computer deve corrispondere allo scarico dei prodotti che compongono il computer stesso

Progetto

- Prodotti sono i PC venduti dall'azienda
- Articoli sono i pezzi che compongono i Prodotti (Assemblati)
- Bisogna modellare anche le relazioni tra articoli e prodotti
- Come procediamo?

Dall'idea al codice

- Brainstorming
- Separare oggetti da azioni
- Oggetti => Concetti
- Azioni => Casi d'uso
- (Casi d'uso => Prototipo)
- Diagramma Classi
- Diagramma di sequenza
- Implementazione

Brainstorming

- Buttare giù le idee così come vengono riguardo all'applicazione che stiamo studiando.

Separare oggetti da azioni

Oggetti

- Articoli
- Prodotto
- Magazzino
- Fattura
- Istanze
- …

Azioni

- Caricare
- Scaricare
- Cercare
- Inserire
- …

Compatibilità

Oggetti=>Classi

- Prodotto: codice, prezzo, collocazione ...
- Articolo: nome, …
- Fattura: totale, elenco prodotti,...
- Magazzino: scaffali
- Composizione:???

Azione => Casi d'uso

Caricare: nell'applicazione quali ruoli degli utenti possono fare questa operazione

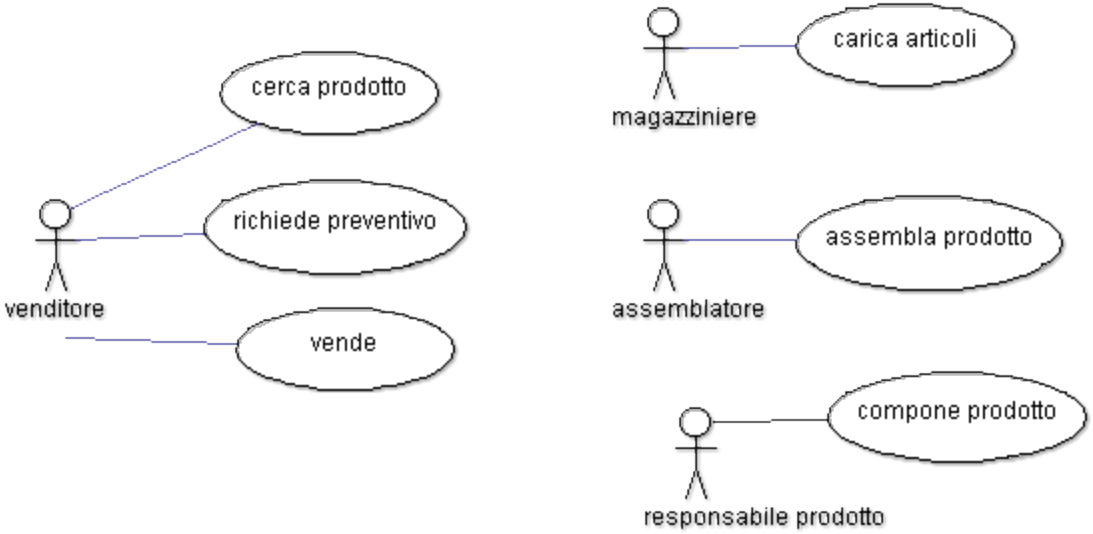
- Scaricare: è simmetrico al carico? No ...
- Cercare: chi, cosa e come può ricercare?
- Inserire: Prodotti, Articoli, Lotti etc.
- Cancellare: chi cosa come (attenzione!)

Casi d'uso

- Come l'utente interagisce con l'applicazione.
- Rappresentano la cosa più importante che l'utente “vede” dell'applicazione.
- Nel caso del nostro progetto ci dicono quali sono le funzionalità che l'utente può usare

Casi d'uso

- Di solito si disegna il diagramma dei casi d'uso.
- I casi d'uso devono essere tutte e le sole interazioni dell'utente con l'applicazione



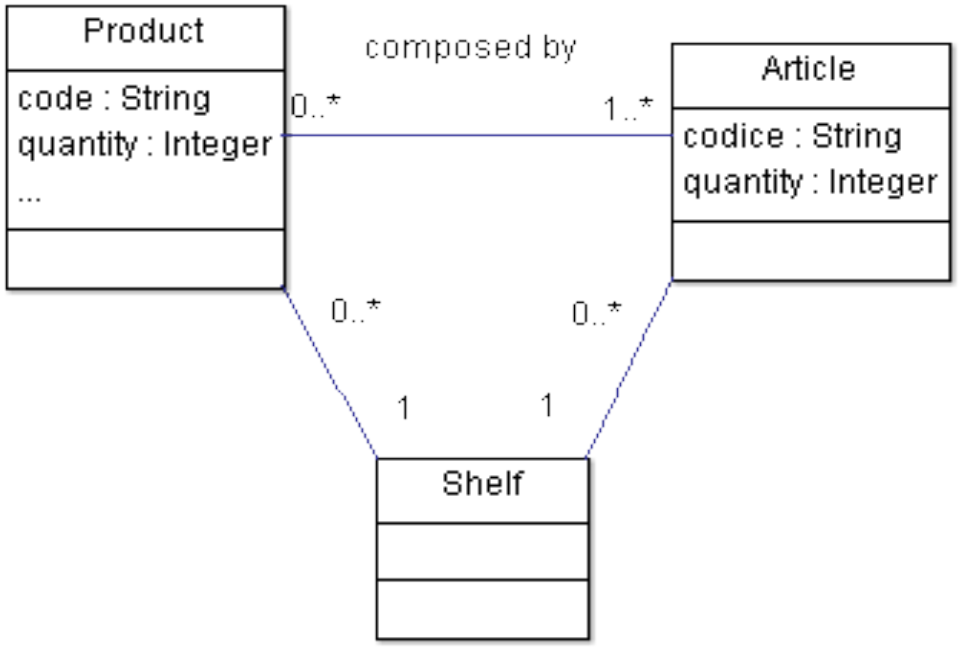
Completare

User Friendly?

Prototipo Applicazione

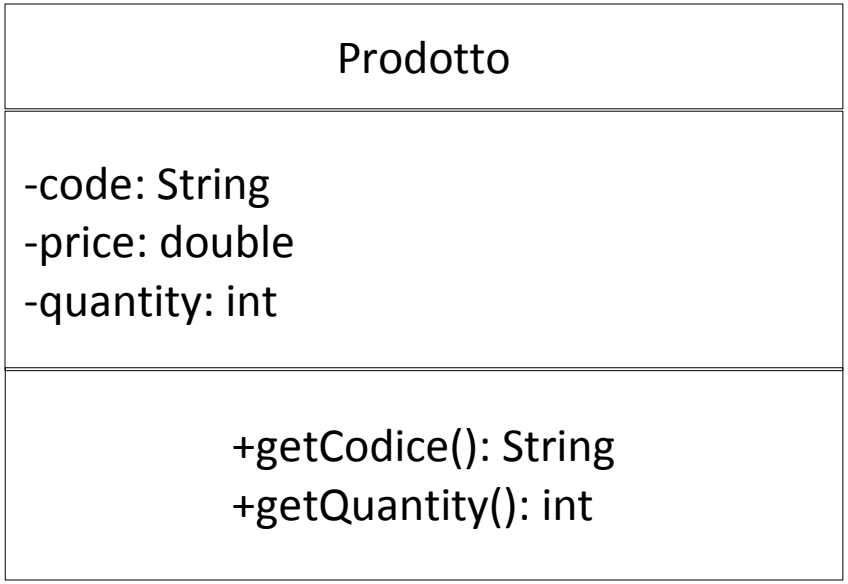
- Aiuta a capire se i casi d'uso esaudiscono tutti i desideri dell'utente.
- E' una delle cose più importanti

Diagramma classi



Parziale, completare

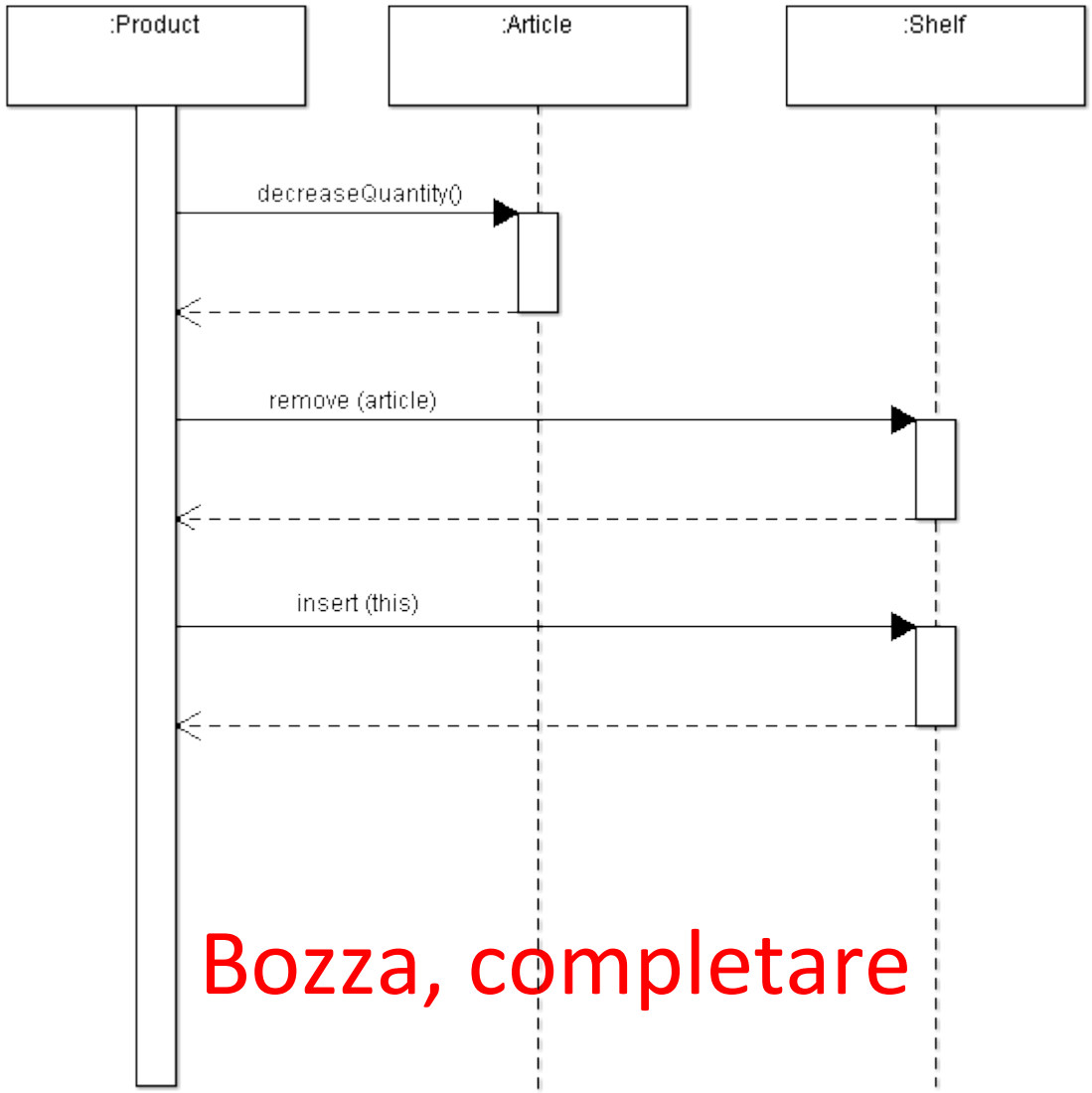
Diagramma delle classi



Diagrammi di sequenza

- Un diagramma di sequenza aiuta a capire come gli oggetti collaborano per implementare un singolo caso d'uso
- Molto spesso può accadere che un caso d'uso semplice non abbia bisogno di un diagramma di sequenza
- Può anche accadere che un caso d'uso abbia bisogno di più diagrammi di sequenza (es. se ci sono molte condizioni)

Dia. Sequenza Caso d'uso Scarica



Bozza, completare

Diagrammi di sequenza

- Fate tutti i diagrammi di sequenza
- I diagrammi di sequenza potrebbero richiedere modifiche al diagramma delle classi
- Modifiche al diagramma delle classi, potrebbero richiedere modifiche ai diagrammi di sequenza

Implementazione

- L'implementazione segue **quasi** direttamente dai diagrammi delle classi e di sequenza
- Dal diagramma delle classi completo è possibile produrre in **automatico** le “interfacce” delle classi. Cioè esistono dei Tool che dal diagramma generano codice Java con gli attributi e le firme dei metodi.

domande