Modelli ad oggetti

- Descrivono il sistema in termini di classi (OOP)
- Una classe ha attributi ed operazioni comuni ad un set di oggetti
- Vari modelli (e diagrammi) ad oggetti possono essere prodotti
  - Di ereditarietà, aggregazione, interazione
- Pro del modello ad oggetti
  - Mappa naturalmente entità del mondo reale
  - Classi che rappresentano entità del dominio sono *riusabili*
- Contro
  - Entità astratte sono più difficilmente modellabili
  - L'identificazione di classi è un processo difficile che richiede una comprensione profonda del dominio applicativo

Identificazione classi

- Dall'elenco dei requisiti
  - Analisi grammaticale del testo
    - Nomi --> classi o attributi
    - Verbi --> operazioni
  - Individuare oggetti fisici
    - Questi suggeriscono classi corrispondenti
  - Raggruppare in modo coeso operazioni tra loro e dati tra loro
    - Questi gruppi suggeriranno delle classi

Esempio: 'Gestione Ordini'

- Requisiti (frammenti)
  - … dovrà essere possibile cercare un cliente ed avere mostrati i dati anagrafici del cliente trovato
  - … la scheda cliente dovrà mostrare tutti i dati anagrafici ed un elenco di fornitori da cui il cliente ha già acquistato
  - … su richiesta dell'utente dovrà essere calcolato l'importo complessivo degli ordini fatti dal cliente nell'intervallo di tempo selezionato
  - … per ciascun ordine dovranno essere mostrati: nome fornitore, nome cliente, linea di appartenenza dei prodotti acquistati, importo complessivo
  - … il report mensile dovrà contenere per ciascun cliente: la provincia di appartenenza e il totale ordinato per ciascun fornitore

Esempio: 'Gestione Ordini'

- Requisiti
  - … dovrà essere possibile cercare un cliente ed avere mostrati i dati anagrafici del cliente trovato
  - … la scheda cliente dovrà mostrare tutti i dati anagrafici ed un elenco di fornitori da cui il cliente ha già acquistato
  - … su richiesta dell'utente dovrà essere calcolato l'importo complessivo degli ordini fatti dal cliente nell'intervallo di tempo selezionato
  - … per ciascun ordine dovranno essere mostrati: nome fornitore, nome cliente, linea di appartenenza dei prodotti acquistati, importo complessivo
  - … il report mensile dovrà contenere per ciascun cliente: la provincia di appartenenza e il totale ordinato per ciascun fornitore

Identificazione classi

- Classi (in verde)
  - Cliente, Fornitore, Ordine, Prodotto, ReportMensile, SchedaCliente
- Attributi (in marrone)
  - Dati anagrafici cliente, nome cliente, provincia cliente
  - Linea appartenenza prodotti
  - Importo ordine
  - Nome fornitore
- Metodi (in arancio)
  - Cercare un cliente
  - Mostrare dati anagrafici e fornitori per un cliente
  - Calcolare totale ordini per un cliente
  - Selezionare ordini in un intervallo temporale
  - Calcolare totale ordini per un cliente per ciascun fornitore per mese

Notazione UML per classi e interfacce

- Esistono varie notazioni per la classe e l'interfaccia (Java)
- Le notazioni indicano
  - Nome classe; nome classe e attributi; nome classe, attributi e metodi
    - Per la visibilità di attributi e metodi: + public, # protected, – private
    - I nomi delle interfacce sono in corsivo
- Uno stereotipo (es. «interface») indica una variazione di un elemento UML, che ha tutte le proprietà dell'elemento di partenza

Interfaccia

```
public interface Persona {
   public String getNome();
   public boolean isEmpty();
}
```

Notazione UML per classi e interfacce

- Forma degli attributi

```
visibilità nome: tipo                    es. – prezzo : int
visibilità nome: tipo = valore iniziale  es. – prezzo : int = 10
visib nome[molteplicità]: tipo           es. – prezzo[5] : int
```

- Forma dei metodi

```
visibilità nome(par: tipo): tipo di ritorno  es. + getCosto() : int
```

  - I metodi statici sono sottolineati

Classe

```
public class Prodotto {
  private String nome;
  private int costo;
  public int getCosto() {
    return costo;
  }
}
```

Diagramma UML delle classi

- Il diagramma delle classi mostra le classi, le loro caratteristiche e le loro relazioni (ereditarietà, implementazione, associazione, uso)
- Una associazione descrive una connessione tra istanze delle classi e specifica
  - Molteplicità: quante istanze di una classe possono essere in relazione con una istanza dell'altra classe (* indica illimitate)
    - Se indicato come attributo, es. `nome[0..1] : Persona`
  - Nome ruolo: usato per attraversare l'associazione (è il nome dell'attributo all'interno della classe di partenza)
  - Navigabilità: verso di attraversamento
  - Nome: spiega l'associazione e il verso

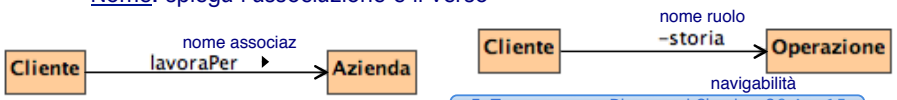


Diagramma delle classi per 'Ordini'

```
import java.util.Date;
public class Ordine {
  private int id;
  private Date data;
  private int importo;
  private boolean archiviato;

  private Cliente cli;
  private Fornitore forn;
  private Stato[] s;
  private Prodotto[] prod;

  public boolean verifica() {
    // implementazione
  }
  public int getImporto() {
    // implementazione
  }
}
```

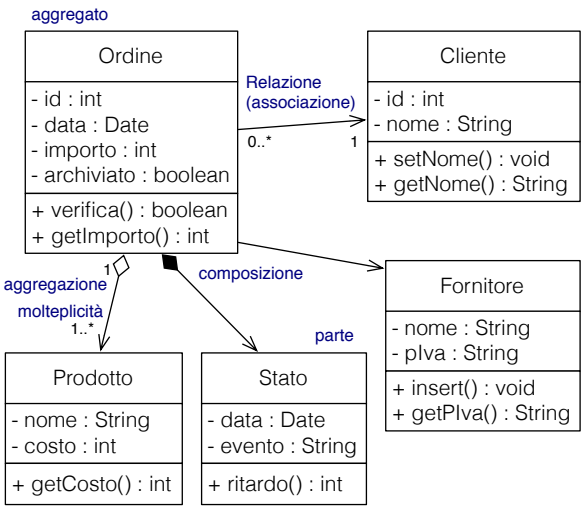
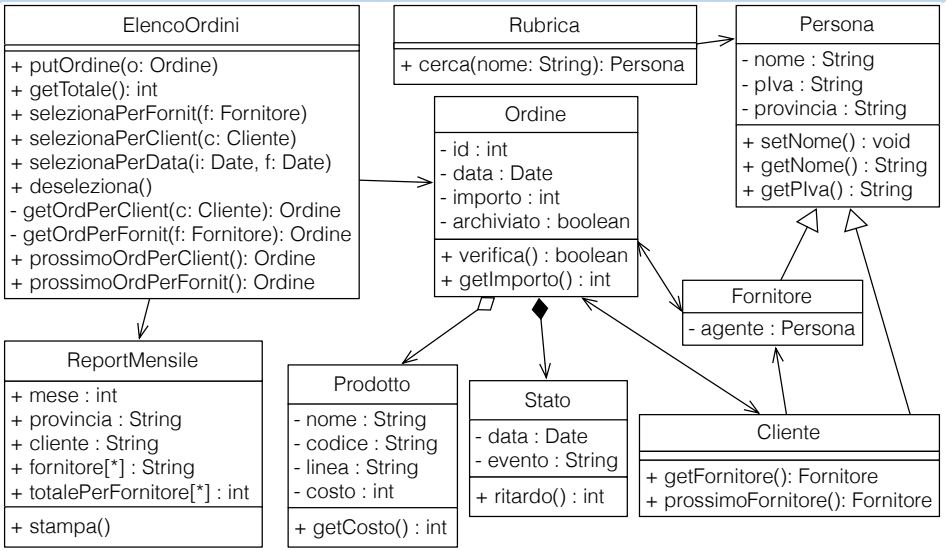


Diagramma delle classi (migliorato)



Considerazioni su requisiti e progettazione

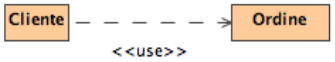
- L'analisi grammaticale sui requisiti non ha evidenziato le classi Persona, Rubrica, ElencoOrdini, e Stato
  - Il progettista deve capire ciò che occorre
- 'Cerca un cliente' è realizzato dal metodo cerca() di Rubrica
  - Rubrica può contenere istanze di Fornitore o Cliente
  - Persona può essere una interfaccia
- 'L'elenco dei fornitori di un cliente' è realizzato dai metodi getFornitore() e prossimoFornitore() di Cliente
  - La classe cliente tiene una lista di fornitori
- 'Calcolare totale ordini' è nel metodo getTotale() d'ElencoOrdini
  - L'insieme degli ordini su cui calcolare il totale è generato dai metodi selezionaPerFornit(), selezionaPerClient(), selezionaPerData()

Considerazioni

- Modularità
  - I metodi selezionaPerFornit(), selezionaPerClient(), selezionaPerData() permettono ciascuno di estrarre una parte degli ordini secondo criteri diversi, quindi si possono usare separatamente, e sono metodi brevi
- Generalizzazione
  - Si possono richiamare i suddetti metodi separatamente per ottenere selezioni di ordini (e totali) differenti rispetto al requisito
  - prossimoXyz() permette di scorrere la lista correntemente selezionata dall'esterno della classe ElencoOrdini [vedi classi Java LinkedList, StringTokenizer, etc.]
- Miglioramenti futuri [vedi lezioni successive]
  - Usare design pattern *Factory Method* per Cliente e Fornitore
  - Usare design pattern *Composite* per Ordine e Prodotto
  - Usare design pattern *Observer* per ReportMensile e ElencoOrdini

Costrutti di estensibilità

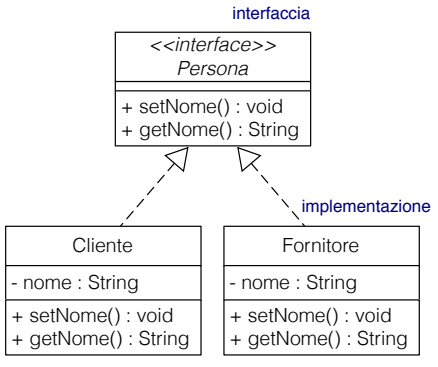
- Vincoli (Constraints)
  - Si usano per indicare condizioni o restrizioni e sono rappresentati da espressioni entro parentesi graffe
  - Es. accanto ad un attributo: {il valore è multiplo di 10}
- Stereotipi
  - Si usano per definire nuovi elementi o per specificare tipi di relazioni sono rappresentati da testo entro « »
- Dipendenze e stereotipi predefiniti
  - Associazioni (o dipendenze) tra classi: «use», «call», instantiate», «destroy»
  - «use» indica che un elemento (Ordine) è richiesto per il corretto funzionamento di un altro (Cliente)
    - Es. necessario a compile time perché è un parametro di un metodo
  - Generalizzazioni tra classi: «implements»
- Tagged Value
  - Coppia di stringhe che indica un dato



ed il suo valore entro { } Es. dentro una classe: {nome=John}

Diagramma UML di ereditarietà

- Organizza le classi in una gerarchia
  - Le classi in alto nella gerarchia (superclassi) mostrano le proprietà comuni delle classi in basso (sottoclassi)
  - Le classi ereditano gli attributi e i servizi da una o più super-classi



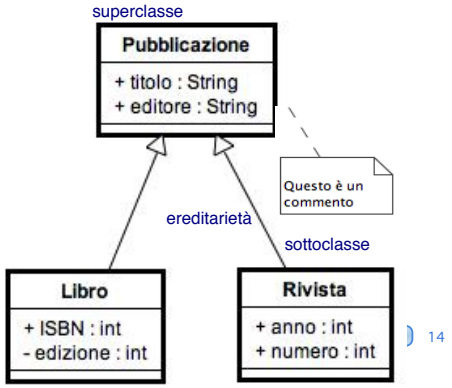


Diagramma UML di sequenza

- Mostra interazioni tra oggetti
  - L'asse temporale è inteso in verticale
  - In alto in orizzontale ci sono i vari oggetti che ne prendono parte
    - In ciascuna colonna verticale, se l'oggetto che partecipa esiste è indicato con una linea tratteggiata, se è attivo con una barra (di attivazione)
  - Un messaggio è una freccia dalla barra di attivazione di un oggetto ad un altro
    - Frecce piene indicano comunicazione sincrona, viceversa vuote asincrona

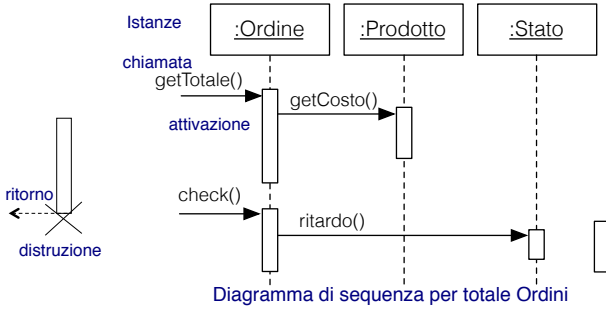


Diagramma di sequenza per totale Ordini

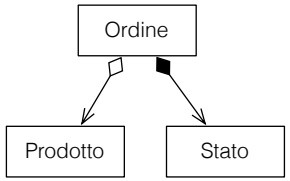


Diagramma delle classi per Ordini

Diagramma UML di collaborazione

- Mostra interazioni tra oggetti
  - Il flusso dei messaggi è indicato da frecce accanto alle associazioni fra istanze che partecipano all'interazione
  - I messaggi sono mostrati da etichette sulle frecce ed hanno
    - Un numero sequenziale che indica l'ordine temporale con cui avvengono
    - Il metodo chiamato
    - Un valore di ritorno (opzionale)

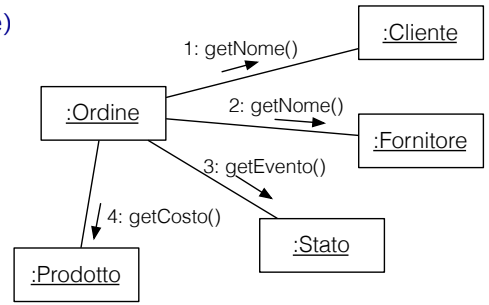


Diagramma di collaborazione per Stampa Ordini