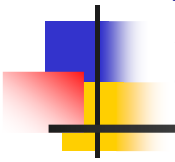
Linguaggi

Corso M-Z - Laurea in Ingegneria Informatica
A.A. 2007-2008

Alessandro Longheu
http://www.diit.unict.it/users/alongheu
alessandro.longheu@diit.unict.it



Esercitazione

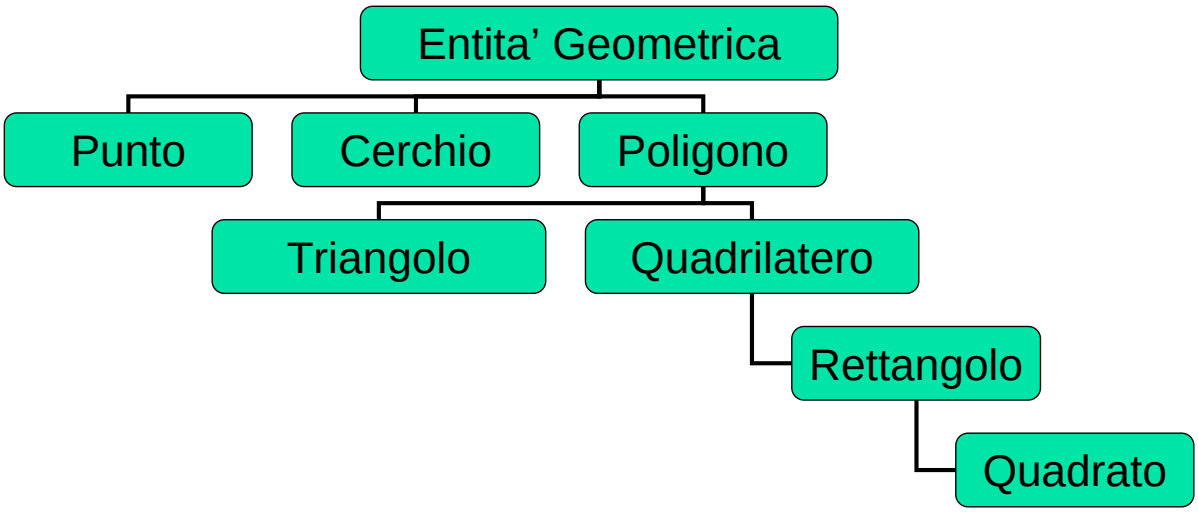
Programmazione Object Oriented in Java

Approccio OO

l'approccio per la stesura di codice OO può seguire i passi:

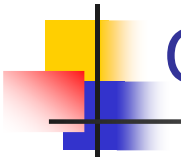
1. **esaminare la realtà**, ed individuare gli elementi essenziali (protagonisti) della stessa
2. **differenziare i ruoli**: gli elementi importanti diventeranno classi, quelli meno diventeranno attributi; le azioni che gli oggetti possono fare o subire diventano invece metodi. Inizialmente, si devono solo individuare classi, attributi e metodi
3. Se qualche classe dovrà possedere attributi e/o metodi già posseduti da altre, **sfruttare il meccanismo di ereditarietà**
4. occorre poi **stabilire il livello di protezione** degli attributi e gli eventuali metodi per la gestione degli stessi (metodi probabilmente necessari in caso di attributi privati); occorre anche stabilire il livello di protezione dei metodi
5. il passo successivo è la **stesura dei costruttori** delle classi, per decidere qual è lo stato iniziale degli oggetti
6. individuati attributi e metodi (passo 2) insieme al livello di protezione (passo 4), l'interfaccia di ogni classe è definita; è quindi possibile passare **all'implementazione dei metodi**

Esercitazione
Gerarchia delle classi di esempio



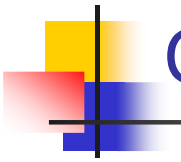
Esercitazione
Codice delle classi di esempio

```
public class EntitaGeometricaPiana extends Object{
        private String colore;
        public EntitaGeometricaPiana(){colore = "nero";}

        protected void setColor (String color){
                colore = color; }

        public void stampa() {
                System.out.println("Sono una generica figura
                        geometrica" );}

        public void disegna() {
                        System.out.println("Disegno la figura
                        di colore " +colore);
```



Esercitazione
Codice delle classi di esempio

```
public class Punto extends EntitaGeometricaPiana {
        private double x, y;
        public Punto() {x = y = 0; }
        public Punto(double newx, double newy) {
                x = newx;
                y = newy; }
        public double getx() { ... }
        public double setx(double newx) { ... }
        public double gety() { ... }
        public double sety(double newy) { ... }
```

Esercitazione
Codice delle classi di esempio

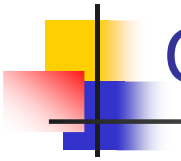
```
public boolean equals(Punto p) {
                if ((x==p.x)&&(y==p.y)) return true;
                return false; }
public void sommaVett(Punto vett) {
                x = x + vett.getx();
                y = y + vett.gety(); }
public void trasla(Punto nuovoCentro) {
x = nuovoCentro.getx();
y = nuovoCentro.gety(); }
public String toString() { .....}
}
```

Esercitazione
Codice delle classi di esempio

```
public class Poligono extends EntitaGeometricaPiana {
        private int num_lati;
        public Poligono(int lati){
                num_lati = lati;}
        protected double lato(Punto a, Punto b) {
                double temp = Math.sqrt((b.gety()-a.gety())*
                (b.gety()-a.gety())+(b.getx()-a.getx())*
                (b.getx()-a.getx()));
                return temp; }
        public void stampa() {
                System.out.println("\nSono un poligono generico"); }
        }
        public String toString () {        ...        }
        public boolean equals(Poligono p) {        ...        }
        }
```

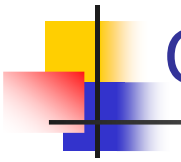

Esercitazione
Codice delle classi di esempio

```
public class Quadrilatero extends Poligono {
        protected Punto v1, v2, v3, v4;
        public Quadrilatero(Punto a, Punto b, Punto c,
                                                Punto d) {
                super(4);
                v1 = new Punto(a.getx(), a.gety());
                v2 = new Punto(b.getx(), b.gety());
                v3 = new Punto(c.getx(), c.gety());
                v4 = new Punto(d.getx(), d.gety()); }

        public Quadrilatero() {
                super(4);
                v1 = new Punto(); v2 = new Punto();
                v3 = new Punto(); v4 = new Punto();}
```



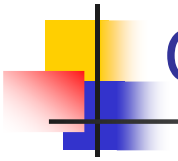
Esercitazione
Codice delle classi di esempio

```
public void trasla (Punto vettore) { ... }
public boolean equals(Quadrilatero q) {
        if (v1.equals(q.v1)&&v2.equals(q.v2)&&
        v3.equals(q.v3)&&v4.equals(q.v4)) return true;
        else return false; }

public double perimetro() {
        double lato1 = lato(v1,v2); double lato2 = lato(v2,v3);
        double lato3 = lato(v3,v4); double lato4 = lato(v4,v1);
        return (lato1 + lato2 + lato3 + lato4); }

public void stampa() { ... }

public String toString() { ... }
```

Esercitazione
Codice delle classi di esempio

```
public class MainClass {
...
  public static void main(String[] args) {
   // crea  un array di figure geometriche
   //disegna tutte le figure
   //calcola il perimetro di tutti i poligoni
   //stampa tutte le figure
   //effettua una traslazione di tutti i poligoni
   //stampa tutte le figure
   }
}
```

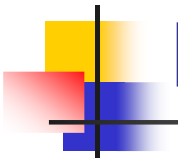


Esercitazione
Codice delle classi di esempio

Input: Utilizzare l'array di stringhe argomento del main nel seguente modo

Numero_di_figure [tipo caratteristiche]+

2 punto giallo 23.5 2.3 triangolo verde 3 1.3 2.4 4.5 3.4 2.2 1.2



Esercitazione – 2

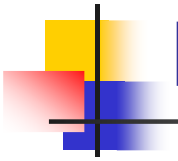
- Occorre realizzare un sistema di prenotazione per un'agenzia di viaggi
- Seguendo i punti indicati in precedenza, si devono individuare i "protagonisti" del problema: cliente, biglietto, prenotazione, agenzia, compagnia, volo
- Nel secondo passo si focalizzano i ruoli e, in accordo ai requisiti imposti dalla realtà di interesse, si decide che cliente, prenotazione, biglietto e agenzia saranno oggetti, quindi occorrono le relative classi, mentre si decide che volo e compagnia non sono rilevanti, quindi risulteranno al più attributi.
- Si decide che biglietto e prenotazione sono distinti per permettere di prenotare senza emettere biglietti

Esercitazione – 2

- in merito al cliente si individuano (sempre dalla precedente fase di raccolta dei requisiti):
  - attributi: nome, cognome, indirizzo e telefono, tutti privati
  - metodi:
    - metodi di gestione degli attributi privati,
    - Prenotazione controlla_prenotazione(String info),
    - Prenotazione prenota(String info),
    - Boolean cancella_prenotazione(String info),
    - Boolean cambia_prenotazione(String info)
  - si suppone che le info siano solo una stringa, in prima approssimazione; potrebbero anche essere un altro oggetto (equivalente ad una struct in C), ad esempio un oggetto di tipo prenotazione
  - si suppone anche che si possano effettuare le quattro operazioni classiche delle basi di dati (inserimento, modifica, cancellazione e ricerca) relativamente alle prenotazioni associate al cliente

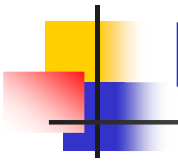
Esercitazione – 2

- in merito al biglietto si individuano un codice ed una stringa descrittiva come attributi (in prima approssimazione pubblici) e due metodi
  - void associa_a_prenotazione(Prenotazione p) per associare un generico biglietto ad una prenotazione esistente
  - void rimborsa(Prenotazione_p), per sganciare il biglietto dalla prenotazione e rimborsarlo ad un cliente
- in merito alla prenotazione, si suppone che essa abbia un codice e una stringa descrittiva come il biglietto (anche qui pubblici); le stringhe non è detto che contengano gli stessi dati;
  - riguardo i metodi, le operazioni sulla prenotazione si suppone vengano fatte dal cliente e/o dall'agenzia; in tal senso, la prenotazione, al pari del biglietto, è un'entità passiva



Esercitazione – 2

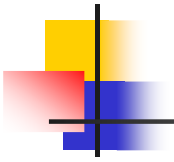
- l'agenzia viaggi è descritta da un nome (e possibilmente da altri campi qui nel seguito tralasciati), e si suppone che essa detenga l'accesso esclusivo all'archivio delle prenotazioni; la struttura di tale archivio potrebbe essere un oggetto, un array, un vector, una lista (esistono librerie già pronte in Java relative alle collezioni), un file o un database (soluzione in genere migliore, e per la quale Java consente la piena gestione tramite librerie).
- Le richieste relative alle prenotazioni vengono dal cliente ma sono di fatto rigirate all'agenzia, che deve implementare gli opportuni metodi
- Per gli attributi delle classi, i tipi di dati vanno scelti ove possibile fra quelli base, o alternativamente fra gli oggetti più adatti (consultare le librerie); se nessun oggetto è adatto si possono definire nuove classi per ottenere l'equivalente di una struct



Esercitazione – 2

Estensioni del sistema:

- supportare N clienti con una sola agenzia; in tal caso conviene pensare in termini di client/server, Java supporta facilmente questa impostazione, permettendo all'agenzia di utilizzare un oggetto di tipo ServerSocket per ascoltare le connessioni richieste da diversi client, ognuno dei quali può aprire un socket ed avere una propria connessione verso l'agenzia.
  - La lista delle prenotazioni diviene una risorsa potenzialmente condivisa, quindi una sezione critica, per la quale si dovranno utilizzare opportuni metodi sincronizzati, a meno che la problematica della risorsa condivisa non venga scaricata sull'eventuale db sottostante (transazioni), caso in cui saranno i comandi SQL lanciati dall'applicativo Java ad essere gestiti dal DBMS
  - L'impostazione client/server non necessariamente richiede la rete
- GUI per cliente e agenzia



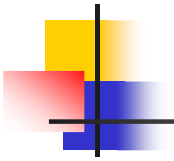
Esercitazione – 2

Estensioni del sistema:

- Gestione da parte dell'agenzia del DB dei clienti, con possibilità di iscrizione, modifica, cancellazione da parte dell'agenzia e/o del cliente
- Sempre ipotizzando che la realtà di interesse lo preveda, si potrebbero avere diverse tipologie di cliente, ad esempio uno standard ed uno business (caratterizzato da un utilizzo frequente del servizio di prenotazione, gestito tramite una tessera). E' buona norma individuare le caratteristiche comuni fra le due tipologie e vedere se si tratta di una superclasse (cliente standard) ed una sottoclasse (business) oppure se entrambe sono sottoclassi (sorelle) della stessa (che rappresenterebbe un cliente generico)
- Si potrebbe anche pensare di avere diverse categorie di biglietto (business, economic, scontati…), utilizzando anche qui il meccanismo di ereditarietà…

Esercitazione – 2

```
public Cliente{
    private String nome; private String cognome;
public Cliente(String n,String c){
if checkString(n) this.nome=n;
if checkString(c) this.cognome=c;
}
public boolean checkString(String pippo) {
for (int i=0;i<pippo.length();i++) {
    if (!Character.isLetter(pippo.charAt(i))) return false;
    }
return true;
}
public Cliente (String n, String c, String i, String t) {
this(n,c);
public String LeggiIdentita(){
    return new String("IDENTITA "+nome+" "+cognome);}}}
```



Esercitazione – 2