Componenti di un programma Java

Outline

- Classi
- Oggetti
- Attributi (dati membro)
- Metodi (funzioni membro)
- Variabili
- Costruttori
- Package
- Altri componenti fondamentali

Classi e oggetti

Una `classe` è un'astrazione indicante un insieme di oggetti che condividono le stesse caratteristiche e le stesse funzionalità.

Un `oggetto` è un'istanza (cioè una realizzazione fisica) di una classe.

Nella programmazione orientata ad oggetti una classe definisce la strutture che avranno gli oggetti che verrano istanziati a partire da essa.

Una sola classe, molti oggetti derivanti da essa.

Esempio classe : classe Punto

```
public class Punto
{
    public int x; // attributo intero x
    public int y; // attributo intero y
}
```

La classe va compilata ma non puo' essere eseguita da sola, una classe non fa nulla se non creiamo, cioe' istanziamo degli oggetti. La classe definisce un concetto, un'idea, da cui un oggetto puo' ereditarne gli attributi e istanziarsi.

```
public class TestOggettiPunto{
    public static void main(String args[]){
        Punto punto1; // oggetto non ancora istanziato
        punto1 = new Punto(); // oggetto istanziato
        punto1.x = 2;
        punto1.y = 6;
        Punto punto2 = new Punto(); // dichiarazione e istanziazione di un secondo oggetto Punto
        punto2.x = 0;
        punto2.y = 1;
        System.out.println(punto1.x);
        System.out.println(punto1.y);
        System.out.println(punto2.x);
        System.out.println(punto2.y);
    }
}
```

Analisi del codice

- Una classe per file
- Classe Prova
- Commenti come in C, // oppure /* ... */
- Dichiarazione oggetto: `Punto` p;
- Istanziazione oggetto: `new` Punto();
- Accesso agli elementi dell'oggetto: operatore `.`
- Visualizzazione a schermo: `println` oppure `print`

Attributi

Nella definizione di classe, quando si parla di caratteristiche ci si riferisce ai dati. In un'applicazione OO (object-oriented), un classe dovrebbe limitarsi a definire la struttura comune di un insieme di oggetti.

La classe Punto non possiede le variabili x e y, ma dichiarando le due variabili, definisce che gli oggetti che verranno istanziati da essa conterranno quelle due variabili.

Metodi

Nella definizione di classe, quando si parla di funzione o funzionalita' ci si riferisce ai metodi. I metodi rendono i programmi piu' leggibili e ne favoriscono la manutenzione e lo sviluppo, inoltre evitano le duplicazioni e favoriscono il riutilizzo del software.

Metodi o funzioni?

Nella programmazione procedurale, come nel caso del linguaggio C, tutti i programmi erano formati da un programma chiamante `main` e da un certo numero di funzioni. Queste avevano di fatto il compito di risolvere determinati "sotto-problemi" generati da un'analisi di tipo top-down, il problema principale veniva scomposto in piu' problemi piu' piccoli.

Nella Object Oriented Programming i sotto-problemi saranno risolti tramite l'astrazione di classi e oggetti, che a loro volta conterranno dei metodi.

```
public class Auto{
    public int numeroRuote = 4;
    public int cilindrata; // inizializzazione allo zero del tipo
    public void muoviti(){
        // codice del metodo
    }
}
```

```
public class TestOggettiAuto{
    public static void main(String args[]){
        Auto fiat600;
        fiat600 = new Auto();
        fiato600.cilindrata = 1100;
        fiat600.muoviti();
        Auto california = new Auto();
        california.cilindrata = 4300;
        california.muoviti();
    }
}
```

Come dichiarare un metodo

[modificatori] tipo del ritorno ([parametri]) {
`corpo del metodo`
}

```
public class Calcolatrice{
    public int somma(int a, int b){
        return (a + b);
    }
}
```

Invocazione di un metodo

Tramite la classe eseguibile, cioe' contente il metodo main(), che istanzia un oggetto dalla classe `Calcolatrice` e chiama il metodo `somma()`:

```
public class Uno{
    public static void main(String args[]){
        Calcolatrice casio = new Calcolatrice();
        int risultato = casio.somma(5, 6);
    }
}
```

Metodi senza parametri o senza return value

```
public class CalcolatriceRotta{
    public int somma(){
        return (5 + 6);
    }
}
```

```
    public class Saluti{
        public void stampaSaluto(){
            System.out.println("Ciao");
        }
    }
```

Metodi con numero variabile di parametri

```
    public class CalcolatricePro {
        public void somma(int... interi){
            //codice complicato. . .
        }
    }
```

```
...
    CalcolatricePro ogg;
    ogg = new CalcolatricePro();
    ogg.somma();
    ogg.somma(1,2);
    ogg.somma(1,4,40,27,48,27,45,67,9,54,66,43);
    ogg.somma(1);
```

Variabili

Le variabili sono porzioni di memoria in cui e' immagazzinato un certo tipo di dato.

[modificatori] tipo dato variabile = [inizializzazione];

```
    public class Rettangolo {
        public int altezza;
        public int larghezza;
        public final int NUMERO_LATI = 4;
    }
```

Variabili d'istanza (o attributi)

Le variabili d'istanza vanno dichiarate dentro una classe ma fuori dai metodi, il ciclo della loro vita coincide con quello dell'oggetto a cui appartengono e vengono inizializzate automaticamente

- In Java non e' possibile definire variabili fuori dalle classi

Variabili locali

Le variabili locali sono anche conosciute come variabili **di stack**, variabili **automatiche** o variabili **temporanee**.

Non sono inizializzate automaticamente ed e' di buona abitudine inizializzarle a valori di default. Il loro ciclo di vita dura dalla definizione della variabile fino alla fine del blocco di codice in cui viene definita

```
    public int somma(int x, int y){
        int z = x + y;
        return z;
    }
```

Parametri formali

I parametri "formali" sono i **parametri** o **argomenti** di un metodo

```
    public int somma(int x, int y){
        return (x + y);
    }
```

Sono inizializzati al momento della chiamata

```
```

```
    int risultato = oggetto1.somma(5, 6);
```

I parametri si possono considerare **variabili locali dei metodi**