Classi e oggetti

Fondamenti di Informatica

Classi ed oggetti

Classe: schema con cui si generano gli oggetti

Oggetti : istanze della classe.

Oggetti e riferimenti

- Ogni “cosa” è un **oggetto**
- Gli oggetti **non** sono associati ad identificatori

- Un programma opera sugli oggetti attraverso dei **riferimenti**
- Ogni riferimento è associato ad un identificatore

OGGETTO ≠ RIFERIMENTO

Riferimenti

```
MiaClasse identificatore;
```

- Crea riferimento “identificatore” a oggetto di tipo MiaClasse
- **Non crea l’oggetto**

identificatore

Riferimenti

```
MiaClasse identificatore = new MiaClasse();
```

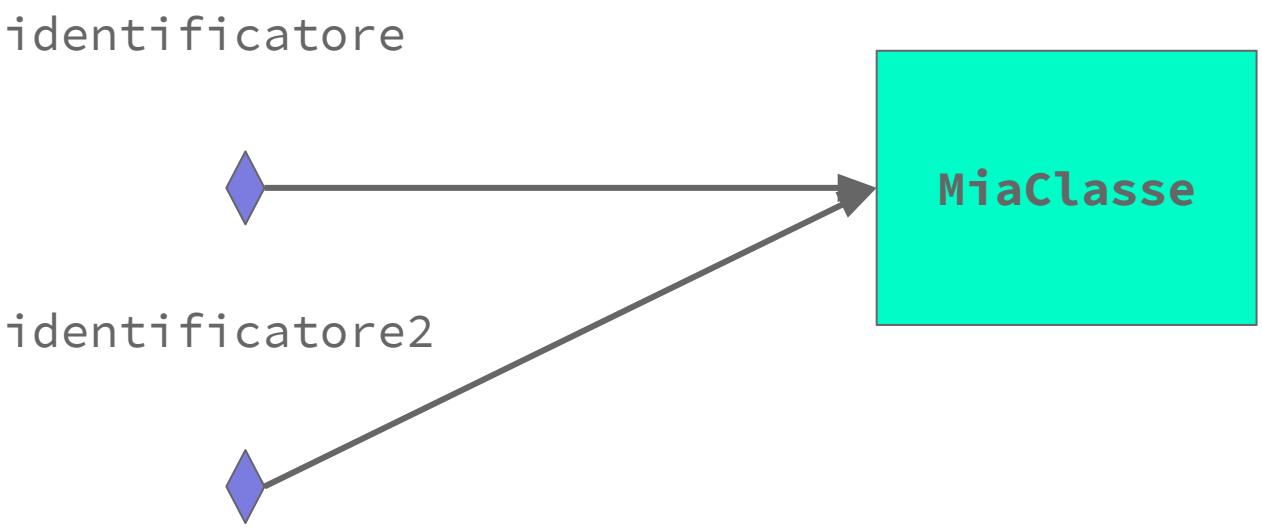
- Crea riferimento “identificatore” a oggetto di tipo MiaClasse
- **Crea oggetto** di tipo MiaClasse e lo inizializza


identificatore
MiaClasse

Riferimenti

```
MiaClasse identificatore = new MiaClasse();
MiaClasse identificatore2 = identificatore;
```

- Crea riferimento “identificatore” a oggetto di tipo MiaClasse
- **Crea oggetto** di tipo MiaClasse e lo inizializza
- Crea riferimento “identificatore2” e lo associa allo stesso oggetto di identificatore


identificatore
identificatore2
MiaClasse

Oggetti e riferimenti

- Oggetto ≈ televisore
- Riferimento ≈ telecomando

- Riferimento: dice “cambia canale”
- Oggetto: esegue l’operazione

Oggetti e riferimenti

- Riferimento: può esistere senza essere associato ad un oggetto
- Oggetto: può esistere senza essere riferito
- **Più** riferimenti possono riferire lo stesso **oggetto**

Oggetti e riferimenti

identificatore

Se manipolo un riferimento senza oggetto ho un errore in esecuzione

Classi

Le **classi** sono la descrizione di un oggetto

Una classe contiene **dati** (proprietà) e **metodi**

- dati: possono essere tipi primitivi oppure altri oggetti
- metodi: operazioni che l'oggetto può compiere

Classi - visibilità

Lo scopo delle classi è nascondere al programmatore i dettagli implementativi

Visibilità di metodi/proprietà:

- public: chiunque può accedervi/usarlo
- private: solo l'oggetto istanziato può usarlo
- protected: (vedremo più avanti)

Classi - dati

```
public class MiaClasse {
   private int proprieta1;
   private int proprieta2;
   // …
}
```

Classi - metodi

Su di un oggetto posso applicare solo le operazioni permesse dal suo tipo/classe (**metodi**)

Sintassi:

```
riferimento.nomeMetodo(listaArgomenti);
```

Metodi - dichiarazione

```
visibilità Tipo nomeMetodo (Tipo parUno,...){
   // codice del metodo
}
```

Metodi - dichiarazione

```
public int nomeMetodo (double parUno){
   // codice del metodo
}
```

Metodi - dichiarazione

```
[public | private] Tipo nomeMetodo ([Tipo par]){
   // codice del metodo
}
```

Overloading

Un metodo di una classe è identificato dalla coppia:

nomeMetodo listaArgomenti

Name overloading: una classe può avere metodi

- stesso nomeMetodo
- diversa listaArgomenti

Classi - Metodi

```
public class MiaClasse {
    public void printSomething(){
        // ...
    }
    private int doSomething(int a, int b){
        // ...
    }

}
```

Valore di ritorno

```
public int nomeMetodo (double parUno){
   // codice del metodo
   return 10;
}
```

Void

```
public void nomeMetodo (double parUno){
   // codice del metodo
   // non serve il return
}
```

Valore di ritorno

```
public int somma (int uno, int due){
   int risultato = uno + due;
   return risultato;
}
```

Valore di ritorno

```
public void stampa (char x, char y){
   System.out.println(“Stampo: ”+ x + “ ” +y);
}
```

In altre parole

Tipo dell'output

```
public int nomeMetodo (double parUno)
```

Eventuali input

Costruttore

Ogni classe ha un'operazione costruttore

- **Crea** un oggetto del tipo della classe
- **Inizializza** l'oggetto

Un oggetto di tipo T lo si ottiene **solo** attraverso il costruttore di T

Il costruttore lo si invoca con **new**:

```
MiaClasse oggetto = new MiaClasse();
```

Costruttore

Ogni classe ha almeno un costruttore senza argomenti (default)

Ogni classe può avere più costruttori, con argomenti diversi

Quando creo la classe con new, scelgo che costruttore usare

String

La classe String è una sequenza di char

Permette di scrivere testi più lunghi di un solo carattere

Non è modificabile

```
String pippo = "Hello word!";
```

Costruttore di string??

Siccome String viene usato moltissimo, il costruttore si può abbreviare:

```
String pippo = "Ciao!";
String pippo = new String("Ciao!");
```

Confronto di Stringhe

```
String testoUno = "ciao";
String testoDue = "ciao";
if( testoUno == testoDue ) {
   System.out.println("Sono uguali!");
} else {
   System.out.println("Sondo diverse!");
}
```

Confronto di Stringhe

```
String testoUno = "ciao";
String testoDue = "ciao";
if( testoUno.equals(testoDue) ) {
   System.out.println("Sono uguali!");
} else {
   System.out.println("Sondo diverse!");
}
```

Operazioni con le stringhe

Concatenazione: posso unire più stringhe con il “+”

```
String pippo = "ciao" + ", come va?";
```

Conversione maiuscole/minuscole:

```
String pippoMaiuscolo = pippo.toUpper();
String pippoMinuscolo = pippo.toLower();
```

Lunghezza di una stringa:

```
int lunghezza = pippo.length();
```

Modifica di una stringa:

```
String modificata = pippo.replace("come va?", "mondo!");
```