Oggetti e dati primitivi

Programmazione
Corso di laurea in Informatica

I membri delle classi

- Le classi contengono 2 tipi di ***membri***, definiti per l'intera classe o per le singole istanze
  - Le ***variabili*** o i ***campi***, che rappresentano lo stato della classe o degli oggetti
  - I ***metodi***, che rappresentano il comportamento: codice eseguibile sottoforma di istruzioni
- Il tipo di un oggetto è definito dalla classe di appartenenza

Esempio

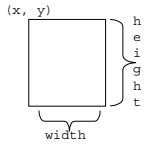
```
Class Point {
      public int x, y;
}
```

- La classe `Point` della libreria `awt` ha due campi, `x` e `y`, che rappresentano le coordinate del punto
- I campi sono dichiarati `public`, cioè chiunque acceda alla classe `Point` può modificarli

La classe predefinita Rectangle

- La classe `Rectangle` della libreria `awt` di Java e i campi d'istanza

Rectangle	
5	x
10	y
15	width
20	height



- I campi `x` e `y` rappresentano la posizione dell'angolo alto sinistro e i campi `width` e `height` rispettivamente l'ampiezza e l'attezza
- Si noti che l'astrazione operata consiste nel considerare un rettangolo come una collezione di 4 valori numerici

Creare oggetti

- Gli oggetti vengono creati mediante uno speciale *metodo di istanziazione*, detto ***costruttore***
- L'operatore `new` seguito dal nome della classe istanzia un nuovo oggetto con valori di default dei campi
  - `new Rectangle ()`
  - Costruisce un rettangolo con i 4 campi al valore 0
- o con i valori passati come parametri
  - `new Rectangle (5, 10, 15, 20)`
  - Costruisce l'oggetto raffigurato prima
- Esempio `TestRettangolo.java`

Variabili e oggetti

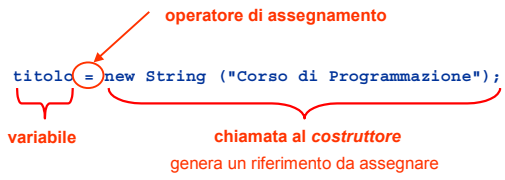
- Spesso l'invocazione al costruttore è associata ad una dichiarazione di variabile e ad un operatore di assegnamento

  sinistra `=` *destra*
- *A sinistra* del simbolo `=` è dichiarata una variabile di dato tipo e nome
- *A destra* è chiamato il costruttore della classe dichiarata come tipo della variabile che genera un riferimento ad un nuovo oggetto
- L'operatore `=` assegna il riferimento all'oggetto creato alla variabile

```
Rectangle rett = new Rectangle ();
```

Gli oggetti stringhe

- Per *istanziare* l'oggetto stringa "Corso di…"
- Un oggetto è un'istanza di una particolare classe



Creare oggetti stringhe

- Per la classe `String` non è necessario invocare il costruttore `new` per creare un oggetto stringa

```
corso = "Programmazione";
corso = new String ("Programmazione");
```

- Speciale sintassi che vale solo per questa classe

```
System.out.println ("Prima le cose importanti");
```

- Il riferimento all'oggetto di tipo `String` creato implicitamente viene passato come parametro al metodo `println`

Oggetti e i loro riferimenti

- Gli oggetti creati sono collocati in un'area di memoria detta ***heap*** e sono accessibili mediante ***riferimenti***

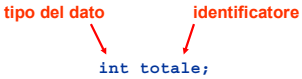
```
Point altoSinistra = new Point ();
Point bassoDestra = new Point();
```

- Le variabili che rappresentano oggetti contengono il loro riferimento tramite cui possiamo accedere ai campi degli oggetti

```
bassoDestra.x = 112;
bassoDestra.y = 40;
```


Variabili

- Una ***variabile*** è un ***dato*** identificato da un ***identificatore***, che rappresenta l'indirizzo della cella di memoria in cui il dato è archiviato
- Una variabile deve essere ***dichiarata***, specificandone l'identificatore e il tipo di informazione che deve contenere




```
int contatore, temp, risultato;
```

Più variabili possono essere specificate in un'unica dichiarazione

Variabili

- Nella dichiarazione può essere già assegnato un valore iniziale alla variabile, mediante l'operatore d'assegnamento `=`

```
int somma = 0;
int base = 32, max = 149;
```

- Quando si richiama una variabile se ne usa il valore
- Esempio `PianoKeys.java`

Assegnamento

- Una ***istruzione di assegnamento*** modifica il valore di una variabile

```
totale = 55;
```

- L'espressione alla destra del simbolo `=` è valutata e il suo valore viene assegnato alla variabile a sinistra
- L'eventuale valore precedente di `totale` è sovrascritto
- Si possono assegnare *solo* valori compatibili con il tipo dichiarato
- Esempio `Geometry.java`

Variabili riferimento a un oggetto

- Una variabile può contenere un valore di un tipo primitivo o il ***riferimento*** a un oggetto
- Il nome di una classe può essere usato per dichiarare una variabile di *riferimento a un oggetto*

```
String titolo;
```

- Nessun oggetto viene creato in questa dichiarazione
- Si dichiara che la variabile di riferimento conterrà l'indirizzo di un oggetto
- L'oggetto stesso deve essere creato separatamente

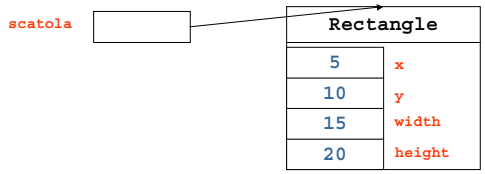
```
titolo = new String ("Il manuale di Java");
titolo = "Il manuale di Java";
```


Variabili oggetto

- La dichiarazione di una variabile oggetto

```
Rectangle scatola;
```

non causa la sua inizializzazione, che va effettuata esplicitamente mediante l'operatore `new`:



Riferimenti a oggetti

- Più variabili possono fare riferimento allo stesso oggetto

```
Rectangle scatola;
scatola = new Rectangle (5,10,15,20);
Rectangle contenitore = scatola;
```

- Il riferimento descrive la posizione dell'oggetto sullo ***heap***
- Ora `scatola` e `contenitore` si riferiscono allo stesso oggetto
- Esempio `TestRettangolo_2.java`

Definiamo una classe

- Con un unico metodo che stampi un messaggio di saluto quando viene invocato su un oggetto della classe
- Esempio `Saluti.java`
- Il metodo `diCiao ()` si conclude riportando all'ambiente chiamante un valore del tipo dichiarato
  `String`
- con l'istruzione `return`

```
return espressione;
return;
```

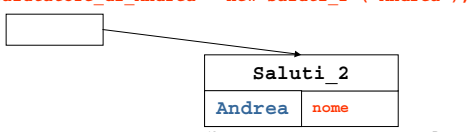

Effettuare un test della classe

- Va definito un programma che spesso si chiama ***driver*** che ha lo scopo di generare oggetti della classe di cui volete fare un collaudo e ne invoca i metodi
- Valutare i risultati ed eventualmente ridefinire la classe per migliorarla o correggerla
- Esempio `TestSaluti.java`

Variabili d'istanza

- Ciascun oggetto di una classe può possedere una propria copia di un data variabile.
- Spesso dichiarata `private` per realizzare la protezione dei dati, mediante ***incapsulamento***
- Esempio `Saluti_2.java`

```
Saluti_2 salutatore_di_Andrea;
salutatore_di_Andrea = new Saluti_2 ("Andrea");
```



Errore comune

- Dimenticare l'inizializzazione di variabili oggetto

```
   Rectangle mio_rettangolo;
*  mio_rettangolo.translate (5, 5);
   Saluti_2 salutaCarlo;
*  salutaCarlo.diCiao();
```

- Le istruzioni `*` generano un *errore*: si applica un metodo a un oggetto che non esiste ancora
- La dichiarazione serve solo per creare la variabile oggetto, ma non per inizializzarla;
- L'inizializzazione va effettuata esplicitamente mediante la chiamata all'operatore `new`

Costruttori vs metodi

- I costruttori non sono metodi
- I costruttori non possono essere invocati su oggetti esistenti
- I costruttori non vengono invocati come i metodi mediante l'operatore `dot` (`.`)
- I costruttori vengono invocati solo all'atto della generazione di un oggetto tramite l'operatore `new`
- Errore:

```
Saluti_2 salutatore_di_Andrea;
* salutatore_di_Andrea.Saluti_2 ("Andrea");
```


I campi statici di classe

- Campi associati alle istanze della classe
  - Contengono informazioni specifiche per ogni oggetto
- Campi associati alla classe
  - Rappresentano dati condivisibili da tutti gli oggetti della classe, specifici della classe e non degli oggetti
  - Le ***variabili di classe***
  - Esempio: `origine` è dichiarata come variabile di classe di una data classe, riferimento a un oggetto della classe `Point`, cioè di tipo `Point`

```
public static Point origine = new Point();
```


Prototipi e segnature

- La ***segnatura*** o ***firma*** di un metodo è costituita dal
  - Nome del metodo
  - Numero e tipo degli argomenti
- Per utilizzare un metodo occorre conoscerne anche il ***prototipo***
  - Segnatura
  - Tipo del valore di ritorno
- ***Overloading di metodi***, stesso nome diversa segnatura
  - Il compilatore sceglie il metodo in base agli argomenti usati
  - Come il metodo `+` usato per sommare e concatenare

Esempi di prototipi

```
int compareTo(String str)

boolean equals(String str)

int length()

String toUpperCase()

String substring(int offset, int indiceFin)
```


Identificatori

- Gli ***identificatori*** sono le parole usate dal programmatore
- Un identificatore è composto da lettere, cifre, il carattere underscore `_` e il segno `$`
- Non possono iniziare con una cifra
- Java è sensibile alle maiuscole, *case sensitive*
  - `Totale` e `totale` sono identificatori diversi

Identificatori - 2

- Alcuni identificatori sono definiti da noi (come `Manzoni`)
- Altri sono stati definiti da altri programmatori e noi li usiamo (come `println`)
- Alcuni identificatori speciali sono detti ***parole riservate*** e hanno un significato prestabilito
- Una parola riservata non può essere ridefinita

Parole riservate

- Sono gli identificatori predefiniti nel linguaggio:

```
abstract   default    goto         operator     synchronized
boolean    do         if           outer        this
break      double     implements   package      throw
byte       else       import       private      throws
byvalue    extends    inner        protected    transient
case       false      instanceof   public       true
cast       final      int          rest         try
catch      finally    interface    return       var
char       float      long         short        void
class      for        native       static       volatile
const      future     new          super        while
continue   generic    null         switch
```


Costanti

- Una ***costante*** è un identificatore il cui valore non può essere modificato dopo la sua dichiarazione iniziale
- Il compilatore segnala un errore se si cerca di modificare una costante
- Si dichiara con il modificatore `final`

```
final int ALT_MIN = 69;
```

  - Rendono chiara la semantica dei valori letterali altrimenti non evidenti
  - Rendono il codice modulare, facilitandone il cambiamento
  - Prevengono errori involontari