

Gli oggetti e le classi

Fondamenti di architettura e programmazione
Corso di laurea in Comunicazione digitale

Introduzione agli oggetti

- Interagiamo con oggetti di uso quotidiano, conoscendone le **funzioni**, ma **non** il **funzionamento interno**
 - Gli oggetti sono quindi **scatole nere** dotate di interfaccia che limita l'accesso ai meccanismi interni
 - Gli oggetti hanno uno **stato**
 - L'insieme delle proprietà che lo caratterizzano in un dato istante
 - e un **comportamento**
 - L'insieme delle azioni che un oggetto può compiere
- Un oggetto sw è un'**astrazione** o un **modello** della realtà che limita il numero dei **dettagli** rappresentati **all'essenziale** per il contesto considerato

AA 2008/09
© Alberti

2

Programmazione
6. Gli oggetti e le classi

Astrazione

- L'astrazione nasconde o ignora dettagli inessenziali in un dato contesto
- Effettuiamo astrazioni continuamente
 - Possiamo trattare solo poche informazioni contemporaneamente
 - Ma se raggruppiamo le informazioni (come gli oggetti) allora possiamo trattare informazioni più complicate
- Un **oggetto sw** è un'astrazione
 - Non ci preoccupiamo dei suoi dettagli interni per usarlo
 - Non conosciamo come funziona il metodo **println** quando l'invochiamo
- Quindi, possiamo anche scrivere software complesso organizzandolo attentamente in classi e oggetti

AA 2008/09
© Alberti

3

Programmazione
6. Gli oggetti e le classi

Gli oggetti software

- Lo **stato** di un oggetto sw è descritto e rappresentato dai suoi **campi**
 - I campi sono sostanzialmente variabili, ma si chiamano diversamente per distinguerli dalle variabili usate nei metodi
 - Una variabile è un **dato** individuato da un **identificatore**
- il **comportamento** è definito dai **metodi**
- Un oggetto sw è costituito dall'insieme dei suoi **membri**: i **campi** e i **metodi**

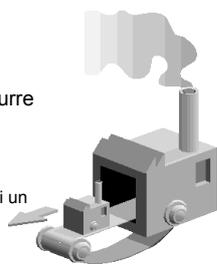
AA 2008/09
© Alberti

4

Programmazione
6. Gli oggetti e le classi

Oggetti e classi

- Gli **oggetti** sono generati da una **classe**
 - sono detti anche **istanze** della classe
- La **classe** è uno schema per produrre una categoria di oggetti identici di struttura
 - La classe definisce il **prototipo**
 - La classe descrive le caratteristiche di un oggetto
 - Una classe è una fabbrica di istanze: possiede lo schema e la tecnica di produzione

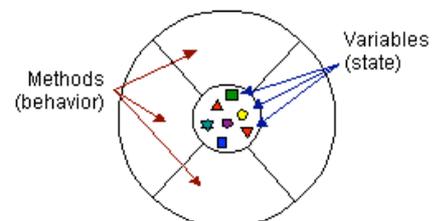


AA 2008/09
© Alberti

5

Programmazione
6. Gli oggetti e le classi

Gli oggetti come astrazione



- Un oggetto che modella una bicicletta
 - La marca, il colore, la velocità (20 Km/h), il giro dei pedali (15 g/m) e la marcia (5^e) sono **campi d'istanza**
 - proprietà rappresentate in ciascuna bicicletta modellata

AA 2008/09
© Alberti

6

Programmazione
6. Gli oggetti e le classi

Gli oggetti come astrazione – 2

- Inoltre nel modello rappresentiamo funzioni come *frenare* o *cambiare marcia*, che modificano i campi d'istanza
- Si chiamano *metodi d'istanza* perché hanno accesso ai campi d'istanza e li modificano

AA 2008/09 © Alberti 7 Programmazione 6. Gli oggetti e le classi

Le istanze

- Definita una classe, si possono creare un numero arbitrario di oggetti appartenenti alla classe

AA 2008/09 © Alberti 8 Programmazione 6. Gli oggetti e le classi

Incapsulamento dei dati

- Nascondere le informazioni fornendo un'interfaccia
 - I *campi* di un oggetto, che ne rappresentano lo stato, sono *nascosti* all'interno dell'oggetto, *accessibili* solo ai metodi
 - Idealmente i metodi proteggono i campi
- Livelli di accesso diversi a metodi e campi:
 - **public**: accessibili a chiunque
 - **private**: accessibili solo alla classe
 - **protected**: accessibili a classe, sottoclassi e pacchetto
 - **senza modificatori**: accessibili solo alle classi del pacchetto
- Consente modularità e flessibilità e protegge i dati

AA 2008/09 © Alberti 9 Programmazione 6. Gli oggetti e le classi

I messaggi

- Gli oggetti interagiscono tra loro per ottenere funzioni più complesse
 - La bicicletta appesa in garage è un oggetto e basta, ci vuole un ciclista che interagisca con lei perché diventi interessante
- Gli oggetti sw per interagire si mandano messaggi
 - Chiedendo di eseguire un certo metodo

AA 2008/09 © Alberti 10 Programmazione 6. Gli oggetti e le classi

I messaggi – 2

- Spesso i metodi necessitano di informazioni per poter essere eseguiti: *i parametri*
- Tre componenti:
 - L'oggetto a cui il messaggio è rivolto: il ricevente
 - Il metodo da eseguire per ottenere un certo effetto
 - I parametri se necessari al metodo

Il messaggio: `changeGears(1000rpm)`

Il ricevente: `YourBicycle`

AA 2008/09 © Alberti 11 Programmazione 6. Gli oggetti e le classi

I messaggi – 3

- Un *oggetto* può anche essere visto come un insieme di *servizi* che possiamo chiedere di eseguire
- I servizi sono definiti dai *metodi*
- Il comportamento degli oggetti è definito dai suoi metodi e il meccanismo di invio dei messaggi consente l'interazione tra gli oggetti
- Gli oggetti che si scambiano i *messaggi* possono anche essere *'distanti'* tra loro
 - Su macchine diverse
 - Non appartenenti allo stesso modello

AA 2008/09 © Alberti 12 Programmazione 6. Gli oggetti e le classi

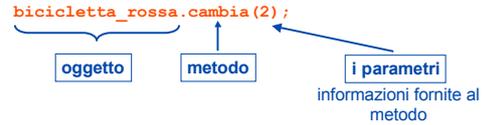
Interfaccia

- L'**interfaccia** è l'insieme dei messaggi che un oggetto è in grado di interpretare
- Un oggetto deve soddisfare la richiesta di un messaggio
- Eseguendo il **metodo** si soddisfa la risposta ad un messaggio da parte di un agente



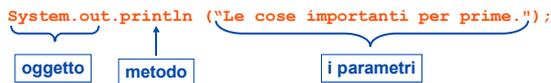
Inviare messaggi

- Un ciclista che vuole cambiare marcia invia il messaggio all'oggetto **bicicletta_rossa**
- Nel codice di un metodo dell'oggetto **ciclista** ci sarà l'istruzione seguente:



Invocazione di un metodo

- Molte istruzioni sono invocazioni di metodi su oggetti
- La sintassi della chiamata del metodo:
`oggetto.nomeMetodo (parametri)`
- Chiediamo il servizio di stampa, invocando il metodo **println** dell'oggetto **System.out**



Metodi e oggetti

- I metodi possono essere invocati su oggetti della classe che hanno quel metodo nella loro interfaccia
 - Il metodo **println** si può applicare a oggetti della classe **PrintStream**
`System.out.println()`
 - Il metodo **length** si può applicare a oggetti della classe **String**
`"salute a tutti".length()`
 - Quindi causa errore chiamare:
`"salute a tutti".println()`

I metodi println e print

- L'oggetto **System.out** della classe **PrintStream** fornisce altri servizi
 - Il metodo **print**
 - simile al metodo **println** - non fa avanzare il cursore alla riga successiva
 - Quindi quello che è stampato dopo l'istruzione **print** appare sulla stessa riga
- Esempio [Conto alla rovescia.java](#)

I membri delle classi

- Le classi contengono 2 tipi di **membri**, definiti per l'intera classe o per le singole istanze
 - 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

AA 2008/09
© Alberti

19

Programmazione
6. Gli oggetti e le classi

I campi statici di classe

- Campi associati alle istanze della classe
 - Contengono informazioni specifiche per ogni oggetto
- Campi **statici** associati alla classe
 - Rappresentano dati condivisibili da tutti gli oggetti della classe, specifici della classe e non degli oggetti
 - Si chiamano anche **campi di classe**
 - Esempio: **origine** potrebbe essere dichiarata come campo di classe in una classe **SistemaCartesiano**, e riferimento a un oggetto della classe **Point**, cioè di tipo **Point**

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

AA 2008/09
© Alberti

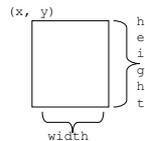
20

Programmazione
6. Gli oggetti e le classi

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'altezza
- Si noti che l'astrazione operata consiste nel considerare un rettangolo come una collezione di 4 valori numerici

AA 2008/09
© Alberti

21

Programmazione
6. Gli oggetti e le classi

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](#)

AA 2008/09
© Alberti

22

Programmazione
6. Gli oggetti e le classi

L'operatore new

- Si usa per istanziare nuovi oggetti di una classe
- È un operatore unario e viene prefisso al proprio argomento: un costruttore della classe
- **new costruttore_classe()** costituisce una espressione di Java (e non un'istruzione)
- Riporta un valore: un **riferimento** all'oggetto della classe specificata dal costruttore
- Il riferimento viene generalmente salvato in una variabile mediante assegnamento

AA 2008/09
© Alberti

23

Programmazione
6. Gli oggetti e le classi

Riferimenti

- Il **riferimento** a un oggetto contiene l'indirizzo di memoria dell'oggetto
- Descriviamo l'indirizzo come un **puntatore** all'oggetto

```
PezzoScacchi alfiere = new PezzoScacchi();
```



AA 2008/09
© Alberti

24

Programmazione
6. Gli oggetti e le classi

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;
```

AA 2008/09
© Alberti

25

Programmazione
6. Gli oggetti e le classi

In memoria

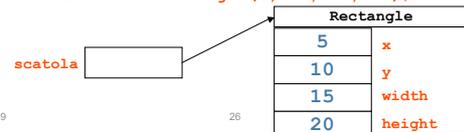
- La dichiarazione di una variabile oggetto riserva una posizione di memoria e associa l'indirizzo all'identificatore

```
Rectangle scatola; scatola
```



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

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



AA 2008/09
© Alberti

26

Programmazione
6. Gli oggetti e le classi

Variabili

- Una **variabile** rappresenta un **dato** identificato da un **identificatore**, valore simbolico dell'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

```
tipo del dato      identificatore
      |              |
      v              v
int totale;
int contatore, temp, risultato;
```

Più variabili possono essere specificate in un'unica dichiarazione

AA 2008/09
© Alberti

27

Programmazione
6. Gli oggetti e le classi

Assegnamento

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

```
totale = 55;
```



- Semantica operazionale:**
- 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](#)

AA 2008/09
© Alberti

28

Programmazione
6. Gli oggetti e le classi

Variabili e astrazione

- Il concetto di **variabile** è un'astrazione del concetto di locazione di memoria.
- L'assegnamento di un valore a una variabile è un'astrazione dell'operazione **STORE**

AA 2008/09
© Alberti

29

Programmazione
6. Gli oggetti e le classi

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**
- 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");
```

AA 2008/09
© Alberti

30

Programmazione
6. Gli oggetti e le classi

Variabili e riferimenti

- L'istruzione di assegnamento per memorizzare un riferimento in una variabile


```
<sinistra> = <destra>;
```
- <sinistra> è una variabile di un dato tipo e nome
- <destra> è un'espressione Java
 - Es: La chiamata al costruttore di una classe che genera un riferimento ad un nuovo oggetto
 - La classe deve essere coerente con quella dichiarata come tipo della variabile
- L'operatore di assegnamento = assegna il riferimento all'oggetto creato alla variabile

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

AA 2008/09
© Alberti

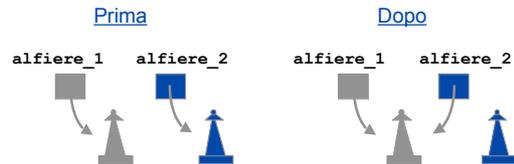
31

Programmazione
6. Gli oggetti e le classi

Assegnamento

- Per variabili riferenti a oggetti, l'istruzione di assegnamento copia l'indirizzo di memoria:

```
alfiere_2 = alfiere_1;
```



AA 2008/09
© Alberti

32

Programmazione
6. Gli oggetti e le classi

Alias

- Due o più riferimenti allo stesso oggetto si chiamano **alias**
- Un oggetto e i suoi dati possono essere visti mediante variabili diverse, che lo referenziano
- Gli alias sono utili, ma devono essere usati con cautela
- Cambiare lo stato di un oggetto (le sue variabili) tramite un riferimento, causa il cambiamento anche per tutti gli altri **alias**

AA 2008/09
© Alberti

33

Programmazione
6. Gli oggetti e le classi

Garbage Collection

- Quando un oggetto non ha più un riferimento non può più essere referenziato da alcun programma
- L'oggetto diventa inaccessibile e quindi inutile
 - si chiama **garbage** (spazzatura)
- Java effettua una raccolta automatica e periodica degli oggetti inutili (**garbage collection**)
 - per rendere nuovamente utilizzabile per usi futuri lo spazio di memoria che l'oggetto occupava
 - Per ridurre lo spazio occupato dallo **heap**
- In altri linguaggi è responsabilità del programmatore effettuare il rilascio della memoria occupata da oggetti non referenziati e quindi inaccessibili

AA 2008/09
© Alberti

34

Programmazione
6. Gli oggetti e le classi

Riferimenti a oggetti

- Il riferimento descrive la posizione dell'oggetto sullo **heap**
- Più variabili possono fare riferimento allo stesso oggetto


```
Rectangle scatola;  
scatola = new Rectangle (5,10,15,20);  
Rectangle contenitore = scatola;
```
- Ora **scatola** e **contenitore** si riferiscono allo stesso oggetto
- Esempio [TestRettangolo_2.java](#)

AA 2008/09
© Alberti

35

Programmazione
6. Gli oggetti e le classi

Espressioni

- Le espressioni sono sequenze di **operatori** e di **operandi** secondo le regole sintattiche del linguaggio.
- Hanno un **tipo complessivo** determinato dagli operatori e dal tipo degli operandi.
- Danno luogo, in fase di esecuzione, a un **valore**.
- In particolare le **espressioni di creazione di oggetti** hanno come tipo la classe dell'oggetto costruito e in fase di esecuzione producono come valore un **riferimento a un oggetto** della classe specificata dal costruttore.

AA 2008/09
© Alberti

36

Programmazione
6. Gli oggetti e le classi

Tipo di un dato

- Il tipo di una variabile specifica:
 - L'insieme dei valori che questa può assumere e
 - l'insieme delle operazioni che possono essere effettuate su di essa.
- Ad esempio una variabile `x` di tipo intero può assumere come valori solo numeri interi
- su di essa possono essere effettuate soltanto le operazioni consentite per i numeri interi.

AA 2008/09
© Alberti

37

Programmazione
6. Gli oggetti e le classi

Tipi e astrazioni

- La nozione di tipo fornisce un'astrazione rispetto alla rappresentazione effettiva dei dati.
- Tutte le variabili sono rappresentate in memoria come sequenze di bit, che possono essere interpretate diversamente in base ai tipi.
- Il programmatore può utilizzare variabili di tipi differenti, senza necessità di conoscerne l'effettiva rappresentazione.

AA 2008/09
© Alberti

38

Programmazione
6. Gli oggetti e le classi

Dichiarazione e inizializzazione

- 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](#)

AA 2008/09
© Alberti

39

Programmazione
6. Gli oggetti e le classi

La classe ConsoleOutputManager

- Una classe per la gestione dell'output
- Ogni oggetto della classe realizza un canale di comunicazione con il dispositivo di output standard, il video
- La classe mette a disposizione metodi per visualizzare a video vari tipi di dati
- Per poter inviare un messaggio a un oggetto della classe dobbiamo prima creare l'oggetto

```
new ConsoleOutputManager ();
```

AA 2008/09
© Alberti

40

Programmazione
6. Gli oggetti e le classi

Esempio

```
import prog.io.ConsoleOutputManager;  
Class PrimoProgramma {  
    public static void main(String[] a) {  
        ConsoleOutputManager video  
            = new ConsoleOutputManager();  
        video.println("primo esempio");  
    }  
}
```

AA 2008/09
© Alberti

41

Programmazione
6. Gli oggetti e le classi

Compilazione ed esecuzione

- `import ...` costituisce una direttiva per il compilatore e la Java Virtual Machine
- L'argomento della direttiva di importazione `import` indica che la classe da cercare fa parte di un package o libreria
- I separatori `.` indicano le directory in cui cercare
 - `prog.io.ConsoleOutputManager`
 - In ambiente UNIX:
`prog/io/ConsoleOutputManager.class`
 - In ambiente DOS/Windows:
`prog\io\ConsoleOutputManager.class`

AA 2008/09
© Alberti

42

Programmazione
6. Gli oggetti e le classi

La classe ConsoleInput Manager

- Le sue istanze realizzano canali di comunicazione con il dispositivo di input standard, cioè la tastiera.
- Alcuni messaggi che possiamo inviare a un oggetto di tipo **ConsoleInputManager**:
 - **readLine()**
 - permette di leggere una riga di testo
 - **readInt()**
 - permette di leggere un numero intero

AA 2008/09
© Alberti

43

Programmazione
6. Gli oggetti e le classi

Esempio

```
import prog.io.ConsoleOutputManager;
import prog.io.ConsoleInputManager;
class Pappagallo {
    public static void main(String[] args) {
        // i canali di comunicazione
        ConsoleInputManager tastiera =
            new ConsoleInputManager();
        ConsoleOutputManager video =
            new ConsoleOutputManager();
        // lettura e comunicazione
        String messaggio = tastiera.readLine();
        video.println(messaggio);
    }
}
```

AA 2008/09
© Alberti

44

Programmazione
6. Gli oggetti e le classi

Commenti

- Il metodo **readLine** restituisce un valore; un riferimento a un oggetto di tipo **String**
- L'espressione **tastiera.readLine()** è di tipo **String**

AA 2008/09
© Alberti

45

Programmazione
6. Gli oggetti e le classi

... come in ...

```
import prog.io.*;
class Saluto {
    public static void main( String[] args) {
        // i canali di comunicazione
        ConsoleInputManager tastiera =
            new ConsoleInputManager();
        ConsoleOutputManager video =
            new ConsoleOutputManager();
        // lettura nome da tastiera
        String nome =
            tastiera.readLine("Come ti chiami?");
        // comunicazione
        video.print("Buongiorno ");
        video.print(nome);
        video.println("!!");
    }
}
```

AA 2008/09
© Alberti

46

Programmazione
6. Gli oggetti e le classi

Commenti

- Si noti che abbiamo usato il metodo **readLine** con un parametro di tipo **String**
- Si tratta di esempio di **overloading** o **sovraccaricamento** di metodi

AA 2008/09
© Alberti

47

Programmazione
6. Gli oggetti e le classi