Concetti introduttivi

La programmazione

- Programma: sequenza di operazioni semplici (istruzioni e decisioni) eseguite in successione.
  - Un programma indica al computer i passaggi da compiere per svolgere un compito preciso.
  - I programmi danno flessibilità di impiego ai computer
- L'attività di progettazione e implementazione dei programmi è detta *programmazione*.
- I programmi sono scritti utilizzando linguaggi di programmazione

Linguaggi di Programmazione

- I linguaggi di programmazione sono in genere classificati in
- Linguaggi macchina
  - Istruzioni macchina codificate con sequenze numeriche
  - Dipendenti dalla macchina

```
30 40 16 100
156
```

- Linguaggi assembly
  - Istruzioni macchina codificate con codici mnemonici
  - Dipendenti dalla macchina

```
LOAD REG, loc_b
ADD REG, loc_a
MOV loc_b, REG
```

- Linguaggi di alto livello (C, Pascal, Java,ecc.)
  - Istruzioni ad un livello concettuale più elevato
  - Indipendenti dalla macchina

```
b = a+b;
```

Linguaggi di alto livello

- I linguaggi di alto livello consentono un maggiore livello di astrazione
  - Permettono di descrivere l'idea che sta dietro l'operazione da compiere

    Esempio: `if x>0 then print("x è positivo")`
    `else print("x è negativo")`

- Sono più vicini ai linguaggi naturali
- Seguono delle convenzioni rigide per facilitarne la traduzione in codice macchina (compilazione)

Compilazione

- Le istruzioni scritte in un linguaggio ad alto livello devono essere tradotte in istruzioni macchina per poter essere “comprese” dalla CPU
  - Il *compilatore* è il programma che si occupa di tradurre il codice
- L’insieme di istruzioni macchina (linguaggio macchina) dipende dalla CPU
  - Il “back-end” di un compilatore dipende dalla CPU

Linguaggi di alto livello

- I linguaggi di alto livello possono essere classificati in vari modi.
- Di interesse per il corso:
  - Linguaggi procedurali o imperativi
    - C, Pascal, …
  - Linguaggi orientati agli oggetti
    - C++, Java, …
- Altre classi di linguaggi:
  - Linguaggi funzionali
    - Lisp, SML, …
  - Linguaggi logici o dichiarativi
    - Prolog, LDL, …

Paradigma procedurale

- Enfasi sulla soluzione dei problemi mediante modifica progressiva dei dati
  - Esecuzione sequenziale di istruzioni
  - Stato della memoria
  - Cambiamento di stato tramite esecuzione di istruzioni
- Programmi aderenti al modello della macchina di von Neumann
- Molto efficienti
- Ha mostrato limiti nello sviluppo e mantenimento di software complessi
- **Pascal, C**

Influenza del modello di macchina

- Concetto di istruzione
- Concetto di sequenzialità e iterazione
  - Il programma assolve il compito eseguendo le istruzioni in sequenza
- Concetto di variabile e di assegnamento
  - Le celle di memoria hanno un indirizzo e contengono i dati da manipolare
  - Le variabili hanno un nome e un valore
  - L'assegnamento di un valore a una variabile equivale al trasferimento di un dato in una cella

Paradigma funzionale

- Primo tentativo di non rifarsi al modello di macchina di von Neumann
  - Il programmatore può IGNORARE la struttura fisica della macchina e scrivere i propri programmi in maniera assolutamente naturale basata sulla logica e la matematica.
- La computazione avviene tramite funzioni che applicate ai dati riportano nuovi valori
  - Le funzioni possono essere applicate a funzioni in catena e possono essere ricorsive
- **Lisp, ...**

Limite dei linguaggi procedurali

- Costringe a pensare soluzioni che riflettono il modo di operare del computer piuttosto che la struttura stessa del problema.
  - Per problemi non numerici questo spesso è difficile
  - Il riutilizzo delle soluzioni è più complicato e improbabile
  - La produzione e la manutenzione del software sono costose

Linguaggi Orientati agli Oggetti

- I *linguaggi ad oggetti* permettono al programmatore di rappresentare e manipolare non solo dati numerici o stringhe ma anche dati più complessi e aderenti alla realtà (conti bancari, schede personali,…)
  - Progettazione e sviluppo più semplice e veloce
  - Alta modularità
    - Estensibilità e manutenzione più semplici
- Tutto questo si traduce in costi più bassi

Concetti base della OOP

- Incapsulamento dei dati
  - Il processo di nascondere i dettagli di definizione di oggetti, solo le interfacce con l'esterno sono visibili
- Ereditarietà
  - Gli oggetti sono definiti in una gerarchia ed ereditano dall'immediato padre caratteristiche comuni, che possono essere specializzate
- Astrazione
  - Il meccanismo con cui si specificano le caratteristiche peculiari di un oggetto che lo differenzia da altri
- Polimorfismo
  - Possibilità di eseguire funzioni con lo stesso nome che pure sono state specializzate per una particolare classe

Esistono controindicazioni?

- Il paradigma di programmazione orientata agli oggetti paga la sua semplicità e versatilità in termini di efficienza
- Va molto bene per lo sviluppo di applicazioni, ma non è adatto per lo sviluppo di software di base
  - Sistemi operativi
  - Driver
  - Compilatori

Esempio

- Scrivere un programma per la gestione di un conto corrente bancario
- Dove sono finiti i concetti di conto corrente, prelievo, versamento, saldo corrente ?

Dominio del problema e dominio della soluzione

- I linguaggi procedurali definiscono un "dominio della soluzione" che "astrae" la macchina sottostante
  - Astrazione procedurale
- Il programmatore deve creare un mapping fra "dominio del problema" e "dominio della soluzione"
  - Tale mapping è spesso innaturale e di difficile comprensione

Linguaggi orientati agli oggetti

- Forniscono astrazioni che consentono di rappresentare direttamente nel dominio della soluzione gli elementi del dominio del problema
  - Oggetti
  - Classi
  - Messaggi

Osservazioni dal mondo reale

- il mondo reale è costituito da **oggetti:** *persone, animali, piante, automobili, ecc*
- suddivisione: oggetti animati e inanimati
- tutti gli oggetti hanno in comune:
  - **attributi:** *dimensione, forma, peso, età, colore, credito residuo, numero esami superati, etc*
  - **comportamento:** *la palla rimbalza, l'auto accelera, un telefonino spedisce SMS, lo studente supera gli esami, etc*
- oggetti distinti possono avere stessi attributi e comportamento → raggruppabili in **classi**
- oggetti possono essere definiti componendo altri oggetti:

  un'automobile è composta dal motore, dalle ruote, dallo sterzo, dai freni, ecc
- oggetti di classi diverse devono conoscersi per poter comunicare tra loro:

  un telefonino deve conoscere il provider per poter funzionare

OOP – l'approccio

- La progettazione orientata agli oggetti modella il software in termini simili a quelli che le persone usano per descrivere oggetti del mondo reale
- Si progettano classi per poter definire oggetti con certi attributi e comportamento
  - attributi → strutture dati
  - comportamento → procedure
- Le classi hanno relazioni con altre classi
  - per la composizione di oggetti e loro reciproca conoscenza
- Oggetti comunicano tramite messaggi:
  - esempio: *un oggetto conto corrente riceve il messaggio di sottrarre dal totale l'importo prelevato dal cliente*

I vantaggi della OOP

- Meccanismo base:
  - dati e operazioni incapsulati in un oggetto
  - la classe specifica oggetti simili
- Con la OOP si costruisce software combinando classi:
  - le classi sono parti intercambiabili
  - programmi di più semplice manutenzione
    - la modifica è localizzata in una o più specifiche classi
- Progettare grandi sistemi software in modo efficiente
- Riuso: le classi sono riusabili in diversi progetti software

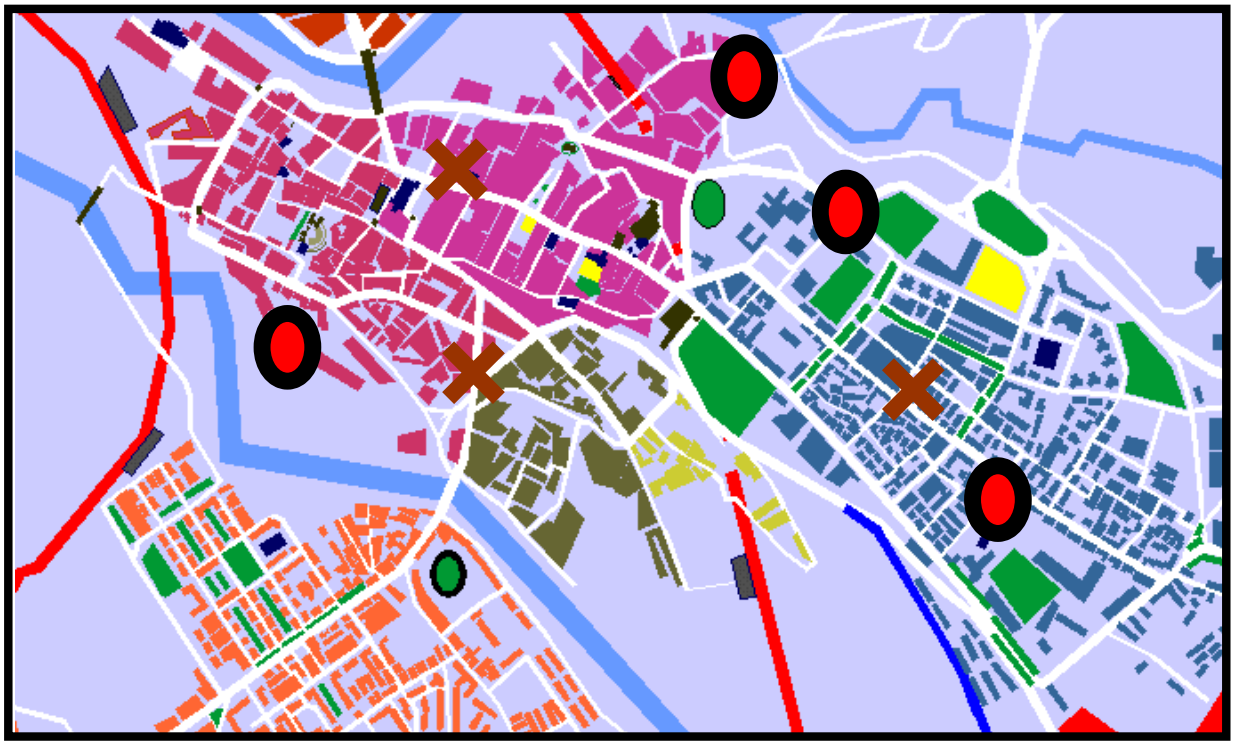
Astrazione

- *Astrazione: Una vista di un oggetto che si focalizza sulle informazioni rilevanti ad un particolare scopo e che ignora le informazioni rimanenti*

- *Information Hiding: Una tecnica per lo sviluppo del software in cui le interfacce dei moduli mostrano il meno possibile del loro funzionamento interno e gli altri moduli sono prevenuti dall'usare informazioni del modulo che non sono definite nell'interfaccia*

Modelli

- I modelli sono nati prima dei calcolatori e NON devono necessariamente essere realizzati mediante calcolatori



Chiamate

Operatori

Elementi del modello

- Ogni modello è formato da **elementi** che rappresentano entità
- Gli elementi del modello presentano un comportamento consistente
- A seconda dei loro comportamenti comuni, gli elementi possono essere raggruppati in categorie diverse
- Il comportamento di un elemento può essere provocato da azioni esterne

Modelli in Java

- Elementi del modello: **Oggetti**
- Le categorie di oggetti vengono chiamate **Classi**
- Una classe
  - Determina il comportamento degli oggetti appartenenti
  - E' definita da una sezione di codice
- Un oggetto
  - Costituisce una **istanza** di tale classe

Esempio

- Classe **operatore**:
  - Definisce il comportamento degli operatori (ad esempio, cambiamento di locazione, registrano il tempo di un intervento, ecc.)
  - Ogni operatore in servizio è una istanza di tale classe

- Classe **chiamata**:
  - Definisce il comportamento delle chiamate (ad esempio, priorità, orario di arrivo, cliente chiamante ecc.)
  - Per ogni chiamata che arriva si crea una istanza

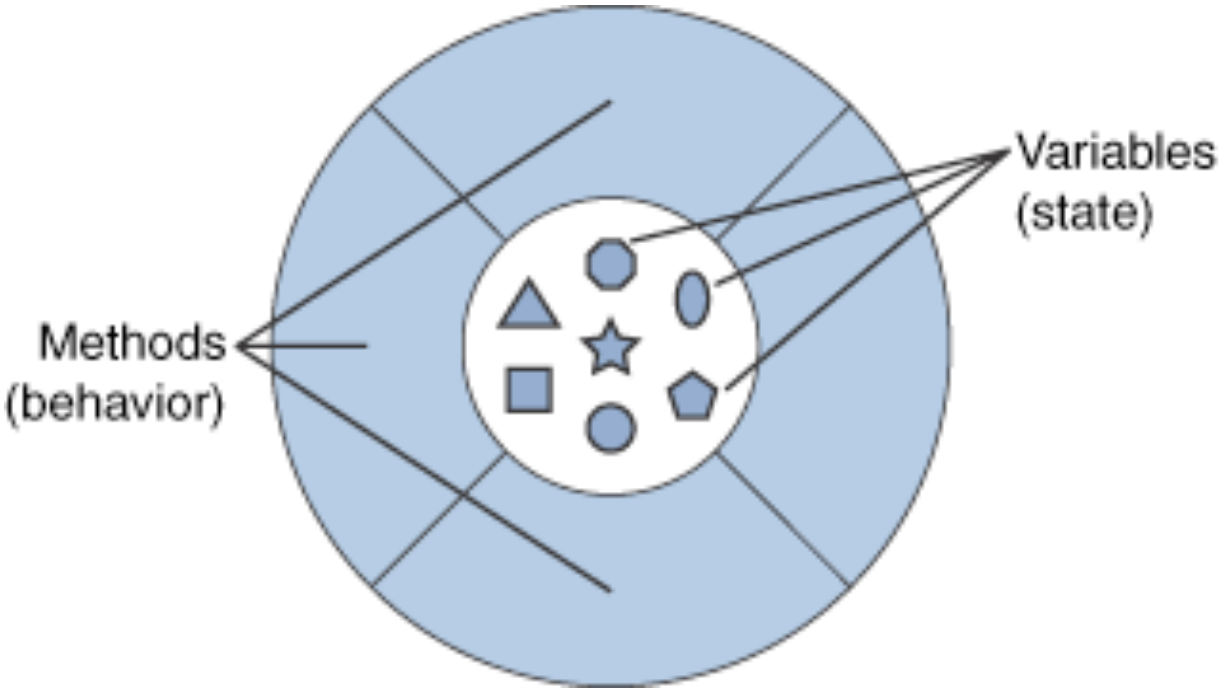
Programmazione OO

- Focus: gli oggetti
  - e le classi che ne definiscono il comportamento
- Filosofia: In un programma in esecuzione sono gli oggetti che eseguono le operazioni desiderate

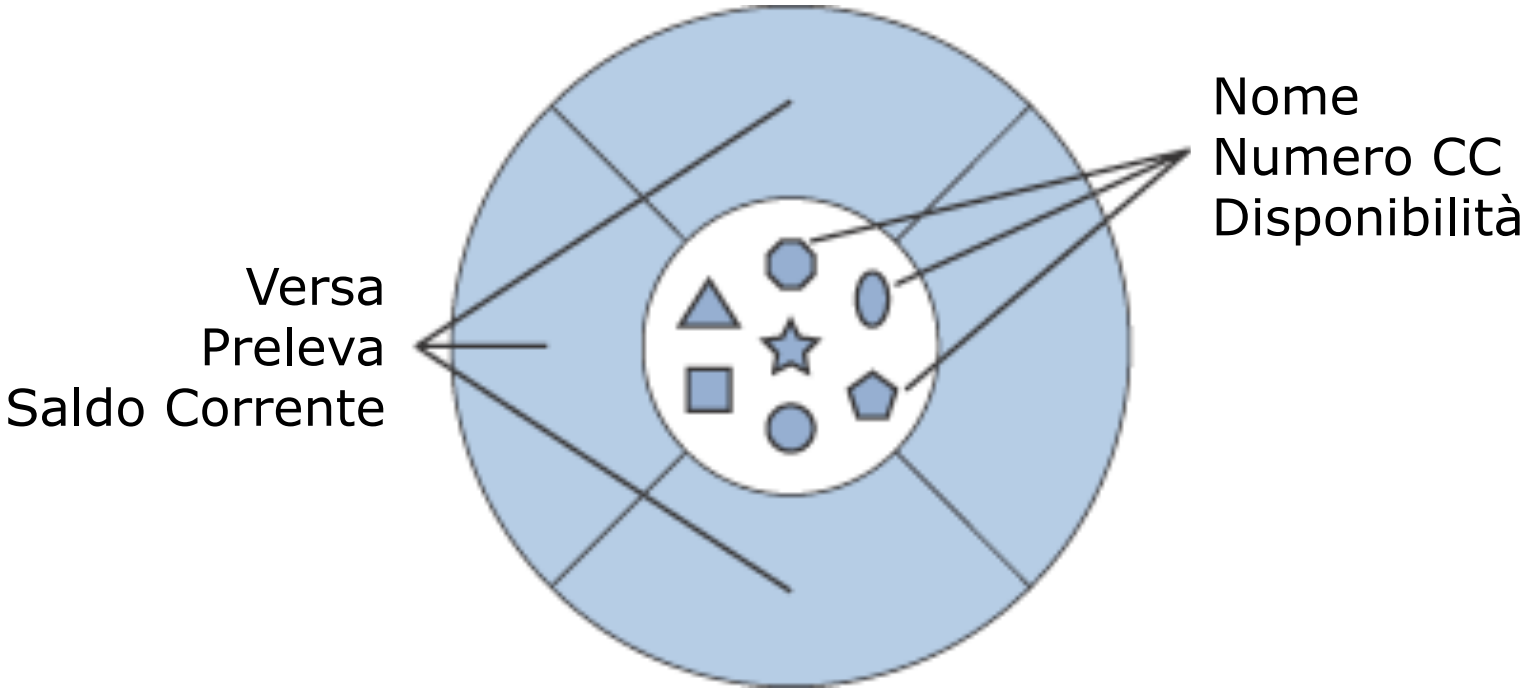
- Programmare in Java
  - Scrivere le definizioni delle classi che modellano il problema
  - Usare tali classi per creare oggetti

- Java è dotato di classi ed oggetti predefiniti
  - Non si deve continuamente reinventare la ruota

Oggetti, astrazione ed information hiding



Oggetti, astrazione ed information hiding



Oggetti e Classi

- Oggetto: entità che manipolerete nei vostri programmi (attraverso l'invocazione di metodi)
- Ogni oggetto appartiene a una classe
- Classe: Insieme di oggetti con lo stesso comportamento
- La classe determina i metodi legali

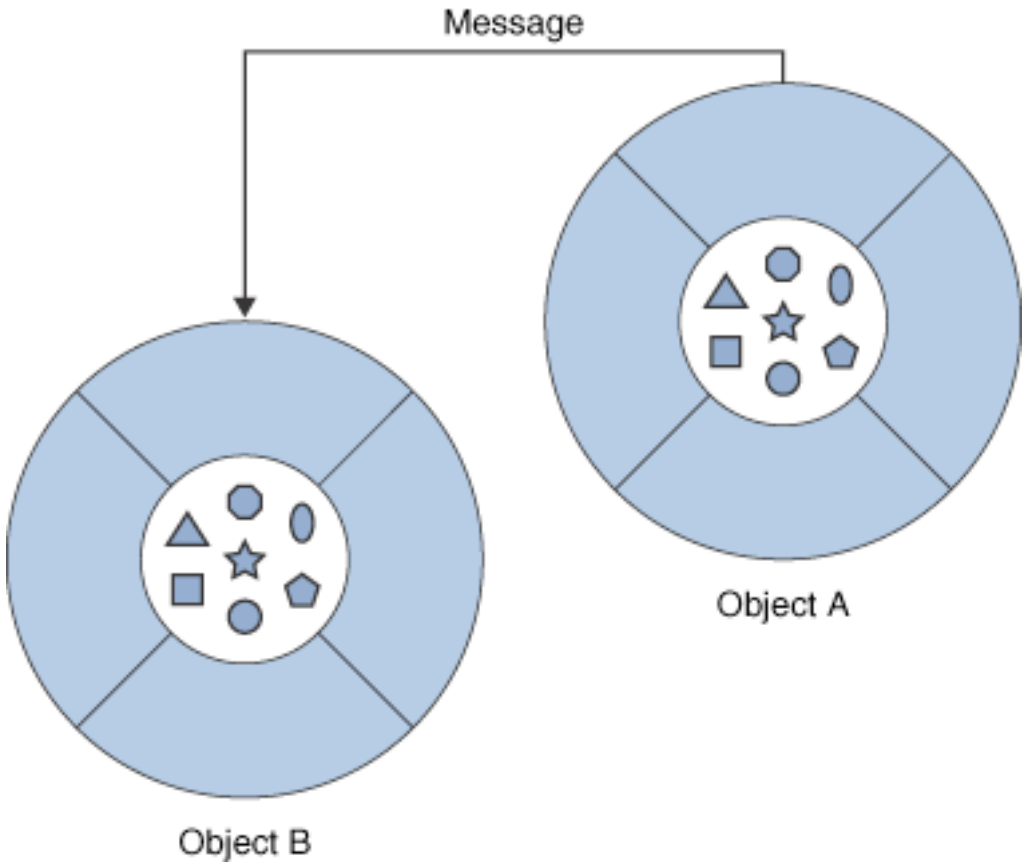
```
"Hello".println() // Error
"Hello".length() // OK
```

Messaggi

- Gli oggetti sono gli elementi attivi di un programma. Come fanno gli oggetti a compiere le azioni desiderate ?

- Gli oggetti sono attivati dalla ricezione di un messaggio
- Una classe determina i messaggi a cui un oggetto può rispondere
- I messaggi sono inviati da altri oggetti

Messaggi



Invio di un messaggio

Invio di un messaggio

Aumenta la luminosità del monitor fisso

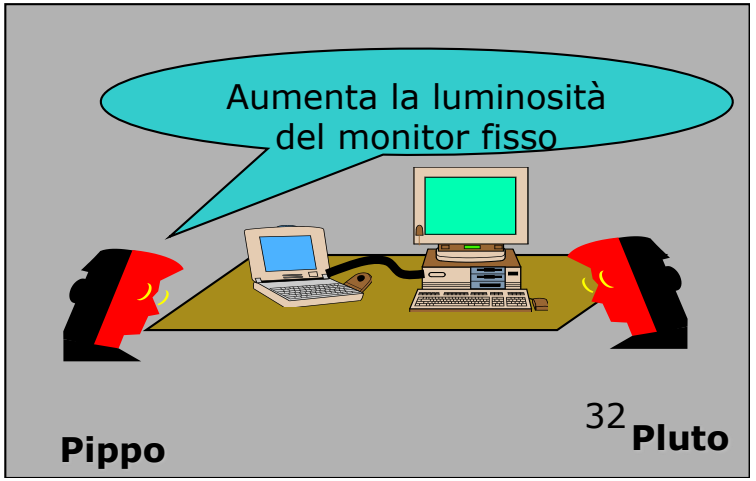
Messaggio

Destinatario
(**Receiver**)

Referenza ad un oggetto

Comportamento:

- **Modifica**
- La proprietà **luminosità**
- In **aumento**



Messaggi

- Per l'invio di un messaggio è necessario specificare:
  - Ricevente
  - Messaggio
  - Eventuali informazioni aggiuntive
- Non tutti i messaggi sono comprensibili da un determinato oggetto:
  - Un messaggio deve invocare un comportamento dell'oggetto

Incapsulamento

Le classi hanno

- un'*interfaccia pubblica*
  - Specifica cosa si può fare con i suoi oggetti
- Un'*implentazione privata*
  - chi usa gli oggetti non si interessa di come sono implementati i metodi che invoca su di essi o di come sono definiti i dati interni.

Nomi e referenze

- Le classi hanno un nome
  - Ogni classe Java deve avere un nome
  - Ogni classe ha un solo nome
  - Es: Impiegato, Molecola, ContoCorrente
  - Convenzione: comincia con una lettera maiuscola
- Regole Java per i nomi (identificatori)
  - Lettere, cifre e caratteri speciali (es: “_”)
  - Devono cominciare con una lettera
  - Il linguaggio è case sensitive

Nomi e referenze

- Gli oggetti NON hanno nome
  - In Java gli oggetti sono identificati da riferimenti
  - Un riferimento (**reference**) è una frase che si riferisce ad un oggetto
    - I riferimenti sono espressioni
  - E' possibile avere più riferimenti ad uno stesso oggetto

Classi ed oggetti predefiniti

Modellano componenti e comportamenti del sistema

Modellano l'interfaccia grafica di interazione con l'utente

Modellano "oggetti" di uso comune, ad esempio Data e Calendario

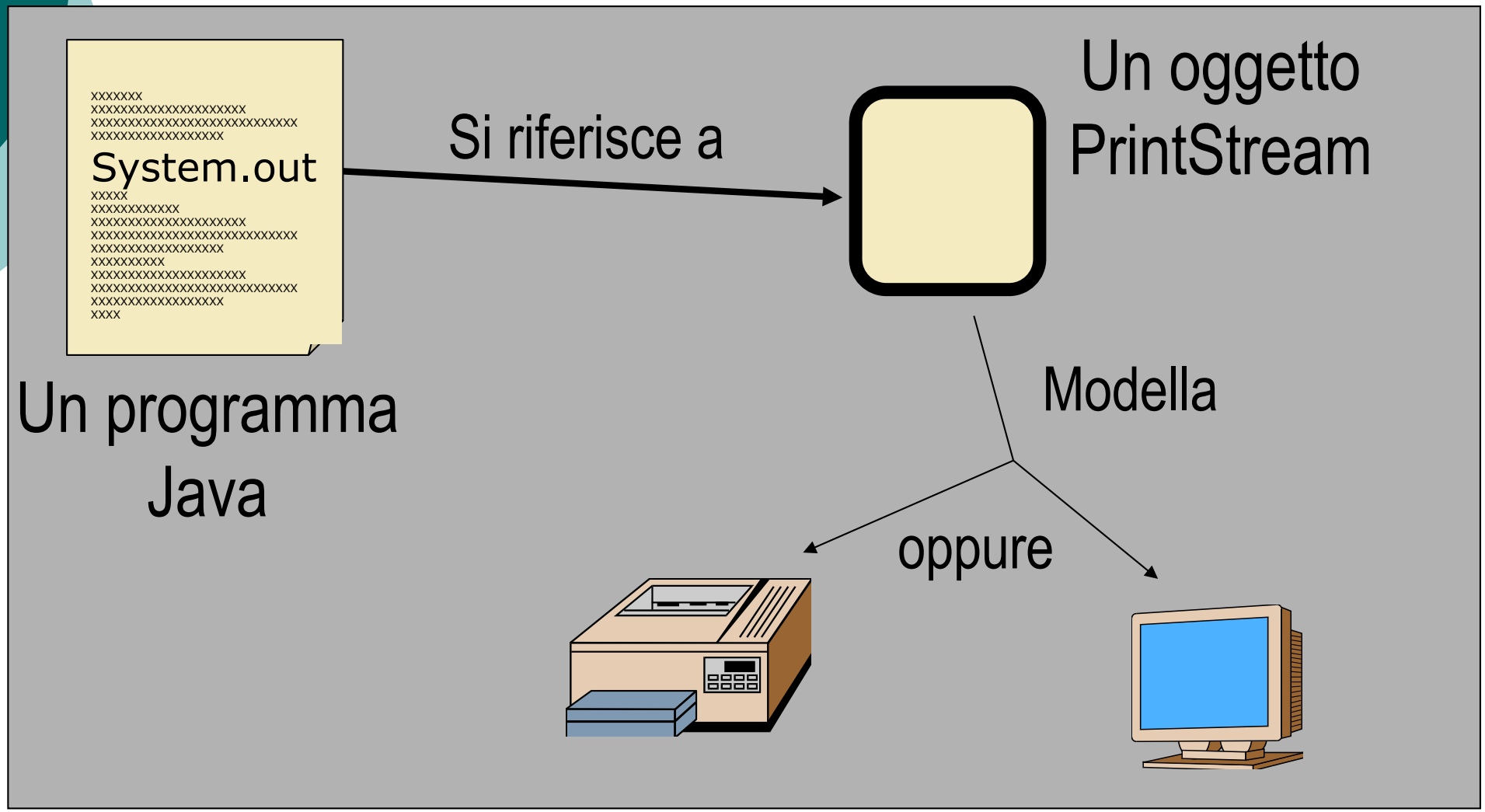
- Un esempio: il monitor
  - Ci si riferisce a lui mediante il riferimento: **System.out**

```
public class Program1 {
      public static void main (String[] arg) {
            System.out.println("Benvenuti al corso");
      }
}
```

PrintStream e System.out

- La classe **PrintStream**
  - Modella monitor e stampanti
  - Comportamento: visualizzare sequenze di caratteri
- **System.out**
  - Un riferimento ad un oggetto predefinito
  - Istanza della classe PrintStream

PrintStream e System.out



Messaggi in Java

- Forma generale

Comportamento-desiderato (altre-informazioni)

- Esempio:
- **println(“Benvenuti al corso”)**
  - Comportamento: **println** (stampa una linea)
  - Informazione: “Benvenuti al corso” (contenuto della linea)

Invio di un messaggio

- Forma generale:

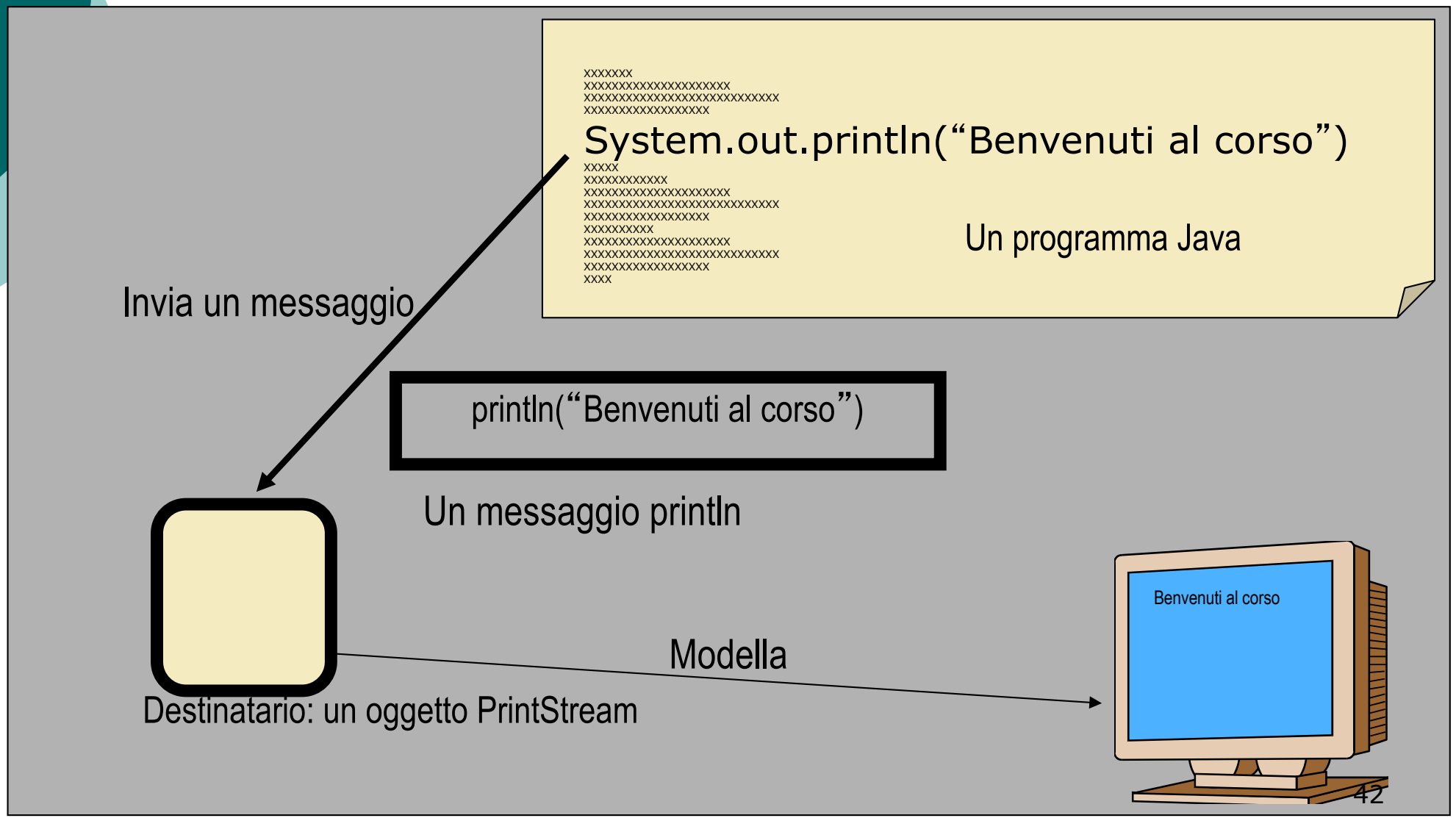
  Riferimento-al-destinatario.messaggio

- Esempio:
- **System.out.println (“Benvenuti al corso”)**

  Riferimento — **Messaggio**

  - L’oggetto a cui si riferisce il riferimento System.out è il destinatario del messaggio println(“Benvenuti al corso”)

Invio di un messaggio



Istruzioni

- Le istruzioni Java
  - Provocano un'azione (es: inviare un messaggio)
  - Devono essere chiuse da punto e virgola "**;**"

- Esempio
- **System.out.println("Benvenuti al corso");**
  - L'invio di un messaggio deve essere sempre espresso da una istruzione

Forma di un programma

- Almeno per le prime lezioni:

```
public class ProgramName {
      public static void main (String[] arg){
            statement;
            statement;
            ...
            statement;
      }
}
```