ASTRAZIONE

"*Nel processo di comprensione di fenomeni complessi, lo strumento più potente disponibile alla mente umana e' l'astrazione.*

L'astrazione nasce dall'individuazione di proprietà simili tra certi oggetti,situazioni o processi del modo reale e dalla decisione di concentrarsi su queste proprietà simili e di ignorare, temporaneamente, le loro differenze."

C.A.R Hoare

"Notes on Data Structuring"

MECCANISMI DI ASTRAZIONE

Utilizzati come base per la definizione di **componenti software**:

- ASTRAZIONE FUNZIONALE
- DATI ASTRATTI
- PROCESSI
- OGGETTI

DATI ASTRATTI

• In molte applicazioni la complessità delle strutture dati **(oggetti dati)** contribuisce in modo determinante alla complessità del problema.

• L'uso della astrazione ha come scopo di consentire **l'utilizzo degli oggetti** senza conoscere i dettagli dell'implementazione (**rappresentazione in memoria**).

• Il comportamento degli oggetti viene descritto tramite un **insieme di operazioni** che sono **associate** agli oggetti e che costituiscono l'unico strumento per operare sull'oggetto.

• Il programmatore può **ignorare** il tipo di implementazione adottato per l'oggetto.

• Le **operazioni** utilizzate dal programmatore costituiscono a loro volta delle **astrazioni funzionali**.

DATI ASTRATTI

Per realizzare il concetto di **dato astratto** occorre:

• **Definire** tutte le operazioni applicabili agli oggetti (**istanze**) del tipo di dati astratto

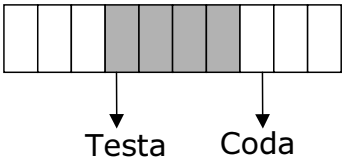
• **Nascondere** la rappresentazione degli oggetti all'utente

• **Garantire** che l'utente possa manipolare gli oggetti solo tramite le operazioni del tipo di dato astratto cui gli oggetti appartengono

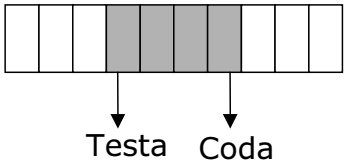
PROPRIETÀ DEI DATI ASTRATTI

• Garanzia di **consistenza** di una struttura dati

Esempio: coda

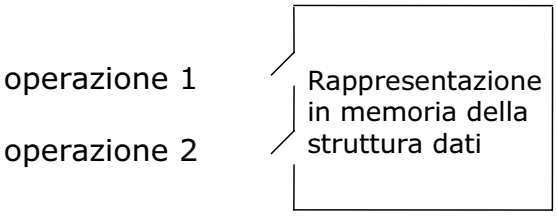


consistente



inconsistente

• **Protezione**



"Information Hiding"

• **Modificabilità dei programmi**
Se viene modificata la rappresentazione in memoria, occorre modificare **solo le operazioni.**

TIPI DI DATI ASTRATTI

type <nome del tipo>= **class**

<dichiarazione di variabili locali**>**

Struttura dati che rappresenta l'implementazione dell'astrazione

procedure entry <nome>(…);
begin… **end**;
.

procedure entry <nome>(…)
begin …**end**;

Nomi di procedure e/o funzioni esportate. Rappresentano le sole operazioni su cui operare su istanze del tipo.

procedure <nome> (…);
begin… **end**;

Procedure e/o funzioni locali al tipo. Possono essere richiamate solo dalle procedure entry.

begin <statement iniziale>**end**

TIPI DI DATI ASTRATTI

```
type T - coda =  class
  var buffer : array[0..N-1] of T; testa, coda : 0..N-1; cont: 0..N;
procedure entry inserzione (x:T);
  begin
    if cont = N then "overflow"
    else
      begin
         buffer[coda]:=x;
         coda:=(coda+1)modN;
         cont:=cont+1;
      end
  end
procedure entry estrazione (x:T);
  begin
    if cont = N then "underflow"
    else
      begin
        x:=buffer[testa];
        testa:=(testa+1)modN;
        cont:=cont+1;
      end
  end
```


TIPI DI DATI ASTRATTI

Dichiarazione:
Var <*nome dell'istanza*>: <*nome del tipo astratto*>;

Operazione:
<*nome dell'istanza*>.<*nome procedure entry*>(parametri attuali);

Esempio:

```
var a,b,c :T;
     coda1,coda2:T-coda;
…
begin
Coda1.inserzione(a);
…..
Coda2.estrazione(b);
….
end
```

SCOPE RULES

• All'esterno della definizione di tipo di dato astratto sono visibili **solo i nomi** delle procedure entry.

• Le procedure,entry o locali,di un tipo di dato astratto possono operare soltanto sui propri parametri e sulle variabili locali al tipo di dato astratto.

• I dati locali al tipo di dato astratto **sopravvivono** indipendentemente dalle chiamate alle procedure del tipo

TIPO DI DATO

Identifica:

- **l'insieme di valori** che un dato di quel tipo può assumere**.**
- **l'insieme delle operazioni** che possono agire su dati di quel tipo (tutte e solo le operazioni significative per dati di quel tipo).

OGGETTI

• **Oggetto :** concetto centrale della progettazione **object-oriented.**
E' una unità software distinta che contiene una collezione di dati correlati e di procedure non visibili direttamente al di fuori dell'oggetto.

• I dati e le procedure contenute in un oggetto sono denominati **variabili** e **metodi.**

• Qualunque cosa un oggetto "conosce" è espresso nelle sue variabili. Ogni cosa che **può fare** è espresso nei suoi metodi.

OGGETTI

• Un oggetto è un ente che possiede **stato**, **funzionamento**, **identità** e **tempo di vita**.

• Gli oggetti interagiscono mediante **messaggi** . Un messaggio contiene:

- il nome dell'oggetto che lo ha spedito
- il nome dell'oggetto che lo riceve
- il nome di un metodo dell'oggetto ricevente
- qualsiasi parametro sia necessario per qualificare l'esecuzione
- del metodo dell'oggetto ricevente.

• Un messaggio può essere utilizzato solo per chiamare un metodo dentro un oggetto.

ORIGINI

• **Linguaggio Simula** (1967), O.J.Dahl e Kristen Nygaard (Simulation Lamguage).

• Obiettivo: realizzare **modelli** accurati di alcuni sistemi fisici complessi.

• Oggetto come modulo basato su oggetti fisici che devono essere modellati nella simulazione.

• Gli oggetti fisici nella simulazione offrono un modo molto naturale di suddividere un problema da risolvere:" ogni oggetto ha un comportamento che deve essere modellato e mantenere informazioni circa il suo stato".

• *"Why looking for some other way to package procedures and data when the problem has already organized for you?"*

STRUTTURA DEGLI OGGETTI

• Le **variabili** in un oggetto sono scalari o strutturate. Ogni variabile ha un tipo (il termine variabile si usa anche per una costante).

• **I metodi** in un oggetto sono procedure che possono essere chiamate dall'esterno per effettuare alcune funzioni. Il metodo può cambiare lo stato dell'oggetto.

• **Incapsulazione** (**information hiding**). I metodi e le variabili di un oggetto sono incapsulate e disponibili attraverso una comunicazione basata sui messaggi.

STRUTTURA DEGLI OGGETTI

Vantaggi:

- Protegge le variabili di un oggetto dalla corruzione da parte di un altro oggetto.

- Nasconde la natura interna dell'oggetto: interazione tra oggetti semplice e standardizzata.

- Se la struttura interna o le procedure sono modificate senza cambiare la funzionalità esterna dell'oggetto, gli altri oggetti non sono influenzati.

CLASSE

- Concetto di base: Raggruppare comportamenti comuni di alcuni componenti in un unico contenitore (**classe**).

- La classe descrive un insieme di oggetti che hanno la stessa interfaccia **(metodi**) e gli stessi attributi (**variabili**).

- La descrizione dei metodi e delle variabili comuni a tutti gli oggetti (**istanze**) è contenuta solo nella classe. Gli oggetti della classe contengono solo i particolari valori delle variabili.

- Ogni oggetto di una classe ha un **proprio nome**. Quando riceve un messaggio per eseguire un'azione, fa riferimento al metodo corrispondente definito nella classe.

ESEMPIO

- Classificazione delle specie animali.
- **Vertebrati** divisi in varie classi:
  - Pesci
  - Anfibi
  - Rettili
  - Uccelli
  - Mammiferi
- Ciascuna classe ha le caratteristiche dei vertebrati, in più ha un insieme di caratteristiche diverse che la distinguono dalle altre.
- Procedimento di **specializzazione.**

EREDITARIETÀ

- L'**ereditarietà** consente ad una nuova classe di oggetti di essere definita in termini di una classe esistente. La nuova classe (**sottoclasse**), contiene automaticamente la definizione di metodi e variabili della classe originaria (**superclasse**).
- Una sottoclasse può differire dalla propria superclasse in vari modi:
  - Può contenere metodi e variabili aggiuntive che non si trovano nella superclasse;
  - Può sovrascrivere la definizione di alcuni metodi o variabili della propria superclasse, usando, nella nuova definizione, lo stesso nome;
  - Può restringere, in qualche maniera, una variabile o un metodo ereditati dalla superclasse.

EREDITARIETÀ

• **Gerarchia di ereditarietà.**
Il meccanismo di ereditarietà è **ricorsivo**; una sottoclasse può divenire superclasse delle proprie sottoclassi.

• **Tecnica di ricerca di metodi e variabili**
Quando un oggetto riceve un messaggio per eseguire un metodo che non è definito nella propria classe, lo ricerca lungo la gerarchia finché non lo trova. Analogamente per le variabili.

EREDITARIETÀ

• Le **superclassi** definiscono un insieme possibile di oggetti, la classe restringe la definizione per indicare un sottoinsieme dell'insieme dato.

• La superclasse in una **gerarchia di specializzazioni**, è frequentemente, ma non sempre, **una classe parziale**; comprende le caratteristiche comuni di diverse classi, ma può non essere in grado di descrivere l'implementazione di alcune operazioni (non descrivono un **oggetto utile**).

• Le sottoclassi possono **specializzare** il comportamento aggiungendo le particolari implementazioni.

EREDITARIETÀ

Modalità operative:

• Definire dapprima classi che modellino i concetti più generali e quindi trattare **casi specifici** tramite sottoclassi (**specializzazione**).

• Creare classi che modellano alcuni concetti, verificare che siano tra loro collegate, estrarre le **caratteristiche comuni** in una o più **superclassi.**

POLIMORFISMO

• Due oggetti che sono **polimorfi** uno con l'altro utilizzano gli stessi nomi di metodi e forniscono la stessa interfaccia agli altri oggetti.

• Es: Oggetti diversi per rappresentare la stampa di documenti diversi (stampaTesto, stampaDisegno..); ciascun oggetto utilizza un metodo chiamato **stampa** implementato in modo diverso.

• Gli altri oggetti invieranno un messaggio con l'indicazione del metodo stampa ad un particolare oggetto. Cambiando il nome dell'oggetto lo stesso messaggio può essere inviato ad un altro oggetto.

ORGANIZZAZIONE DEGLI OGGETTI

• **Composizione:** un oggetto del mondo reale è in genere composto da oggetti più semplici. Le proprietà di un oggetto composto non sono direttamente collegate a quelle degli oggetti componenti.

• **Classificazione:** si basa sul fatto che oggetti diversi possono far parte di classi più vaste di livello superiore. L'oggetto di una classe condivide le proprietà della classe superiore, ossia ne eredita le proprietà.

• Due **orientamenti** nel mondo degli oggetti: il primo privilegia la composizione, l'altro la classificazione.

LINGUAGGI DI PROGRAMMAZIONE

Due categorie:

• **Object-Based Programming Languages (OBPL)**
Es: ADA (package)

• **Object- Oriented Programming Languages (OOPL)**
Es: Smalltalk, C++, Java

• La prima categoria insiste in particolare sul concetto di oggetto astratto, protezione, modularità,**composizione** degli oggetti.

• La seconda, oltre alle proprietà precedenti, insiste sul concetto di classificazione, **ereditarietà** tra le classi.

PAROLE CHIAVE

Oggetto (object)

Metodi (methods)

Incapsulamento (information hiding)

Astrazione (abstraction)

Messaggi (messages)

Classe (class)

Istanza (instance)

Sottoclasse (subclass)

Ereditarietà (inheritance)

Polimorfismo (polymorfism)

BENEFICI DELLA PROGETTAZIONE ORIENTATA AGLI OGGETTI

• Organizzazione della complessità tramite **l'ereditarietà.**

• Possibilità di costruire strutture dati arbitrarie tramite il **contenimento.**

• Riduzione dello sforzo di sviluppo tramite il **riutilizzo dei componenti (classi di oggetti).**

• Sistemi più estensibili e mantenibili tramite la riduzione delle potenziali **interazioni** tra parti differenti del software e la possibilità di **modificare l'implementazione** di una classe senza impatto sulle altre.

• Utilizzo nella progettazione di **sistemi distribuiti** tramite la proprietà di comunicazione tra oggetti attraverso **messaggi.**

JAVA

- Introdotto da **SUN Microsystems** nel 1995.
- Ambiente di programmazione che comprende:
  - Linguaggio di programmazione
  - API (Application Program Interface)
  - Macchina virtuale (Java Virtual Machine, JVM).

Linguaggio di programmazione

- Object oriented, indipendente dall'architettura, distribuito e multithreaded
- Costrutto **class** per specificare gli oggetti; un programma Java consiste di una o più classi.
- Per ogni classe il compilatore fornisce un **bytecode output code** che può essere eseguito da JVM.
- Oggetti distribuiti :**remote method invocation** (RMI)
- Gestione della memoria: **garbage collection.** Viene recuperata la memoria non più utilizzata dagli oggetti.

API

• **Base API.**
Fornisce il supporto al linguaggio per funzioni base come I/O, grafica, utilities e networking (java.lang, java.awt, java.io, java.net).

• **Standard Extension**
Fornisce il supporto per la sicurezza , il commercio elettronico etc..

Macchina virtuale

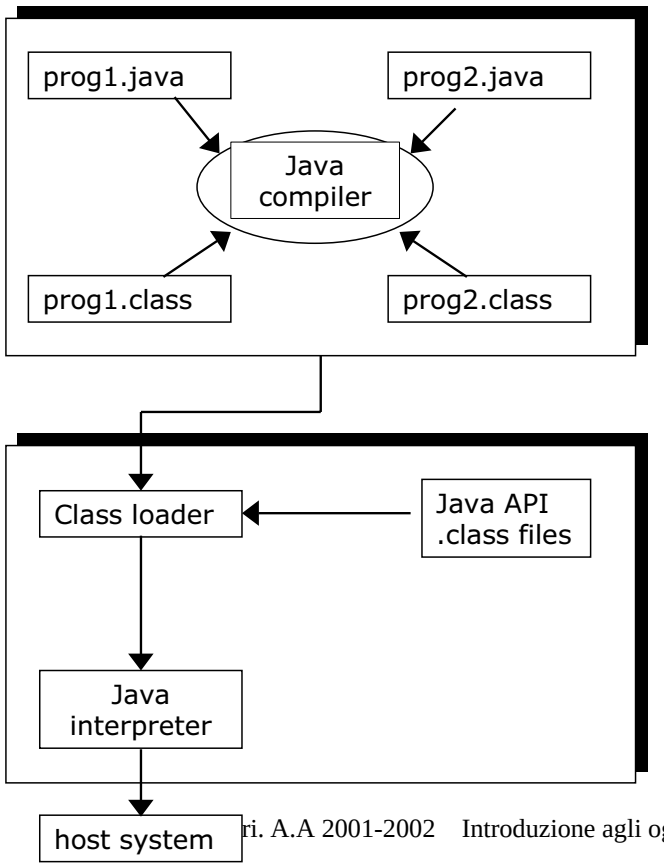
• **Class loader** Carica i file-class (programma+API)

• **Java Interpreter**.

- *Interpreter* che esegue i bytecode uno alla volta
- *Just-in-time compiler* che traduce i bytecode nel linguaggio nativo della macchina del computer host.

Piattaforma Java
Consiste di JVM e JAVA API. Può essere implementata:
- sopra un sistema operativo (Unix, Linux, Windows)
- come parte di un browser (applet)
- in hardware

Ambiente di sviluppo JAVA

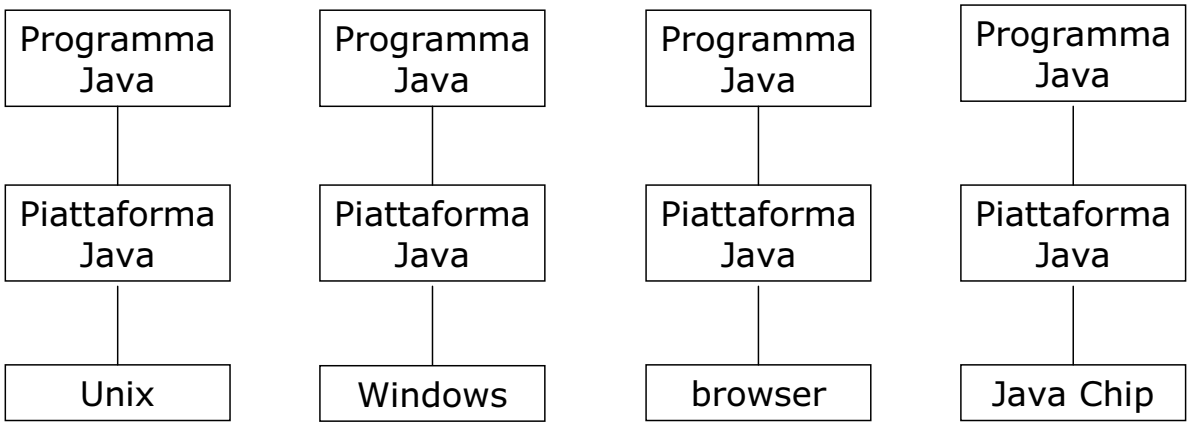


compile-time environment

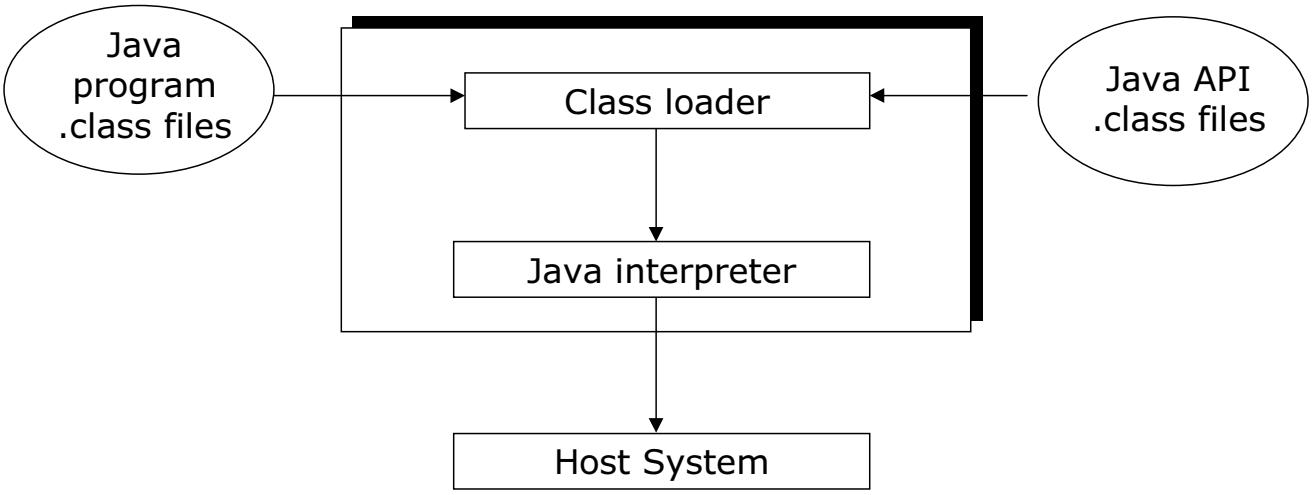
bytecodes move trough local file system or network

run-time environment (Java platform)

Piattaforma Java



Java Virtual Machine



Classi Java su piattaforme diverse