PROGRAMMAZIONE, ASTRAZIONI E LINGUAGGI

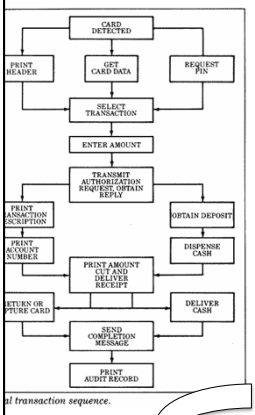
1

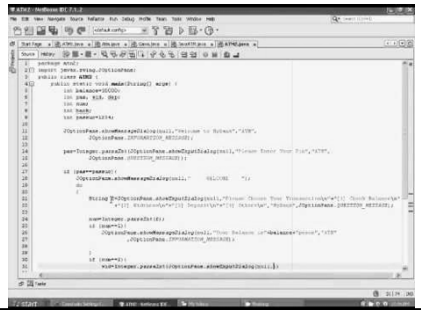
Cosa vuole dire programmare

- Comprendere i requistiti del problema da risolvere
- Progettare l'architettura del software
- Progettare le strutture dati e gli algoritmi
- Testing e debugging
- Profilare e ottimizzare il software
- Adeguarsi agli standard
- :

2

Programmare e' costruire un modello computazionale





Paradigmi di programmazione

- Consideriamo il problema di determinare l'area di una figura geometrica (ad esempio un triangolo).
- La costruzione del modello computazionale eseguibile, ovveri il programma, puo' essere fatta mediante differenti paradigmi di programmazione (e con i relativi linguaggi)

Paradigma procedurale

```
#include<stdio.h>
#include<math.h>

main()
{
  double base, height, area;
  printf("Enter the sides of rectangle\n");
  scanf("%lf%lf%lf",&base,&height);
  area = base * height;
  printf("Area of rectangle=\n", area);
  return 0;
}
```

Un esempio nel linguaggio C

5

Paradigma funzionale

```
let area b h =  b *. h;;
  val area : float -> float -> float = <fun>
```

6

Il problema dell'astrazione

- Epigramma attribuito a Euclide:
- *Un asino e un mulo viaggiavano insieme, portando un carico di sacchi di grano (o otri di vino). L'asino si lamentava per il carico eccessivo. Il mulo gli disse: "Di che cosa ti lamenti? Se tu mi dessi uno soltanto dei tuoi sacchi, io ne avrei il doppio di te. Ma se io ti dessi uno dei miei sacchi, ne avremmo tanti uguali.". Dimmi, o sapiente lettore, quanti sacchi portava l'asino e quanti il mulo?*"

7

Passo di astrazione

- *Un asino (X) e un mulo (Y) viaggiavano insieme, portando un carico di sacchi di grano (o otri di vino). L'asino (X) si lamentava per il carico eccessivo. Il mulo (Y) gli disse: "Di che cosa ti lamenti? Se tu mi dessi uno soltanto dei tuoi sacchi, io ne avrei il doppio di te (Y +1 = 2(X-1)). Ma se io (Y) ti dessi uno dei miei sacchi, ne avremmo tanti uguali.(Y-1 = X+1)".*
- *Dimmi, o sapiente lettore, quanti sacchi portava l'asino e quanti il mulo?*"

8

- *X e Y viaggiavano insieme. X si lamentava per il carico eccessivo. Y gli disse: "Di che cosa ti lamenti? Se tu (X) mi dessi uno soltanto dei tuoi sacchi, io ne avrei il doppio di te (Y +1 = 2(X-1)). Ma se io (Y) ti dessi uno dei miei sacchi, ne avremmo tanti uguali.(Y-1 = X+1)".*
- Y+1 = 2(X-1)
  Y-1 = X+1
- Sistema di equazioni lineari la cui soluzione: sette sacchi per il mulo e cinque per l'asino

9

- Alan J. Perlis, 1922 – 1990:
- A good programming language is a conceptual universe for thinking about programming
- A language that doesn't affect the way you think about programming, is not worth knowing.
- There will always be things we wish to say in our programs that in **all known languages can only be said poorly.**

10

Passo di Astrazione

- Generalizziamo il problema:
  - Determinare il valore dell'area di figure geometriche


```
double size() {
  double total = 0;
  for (Shape shape : shapes) {
     switch (shape.kind()) {
     case SQUARE:
       Square square = (Square) shape;
       total += square.width * square.width;
       break;
    case RECTANGLE:
       Rectangle rectangle = (Rectangle) shape;
       total += rectangle.width * rectangle.height;
       break;
    case CIRCLE:
      :
     }
  }
  return total;
}
```

Astrazioni sui dati

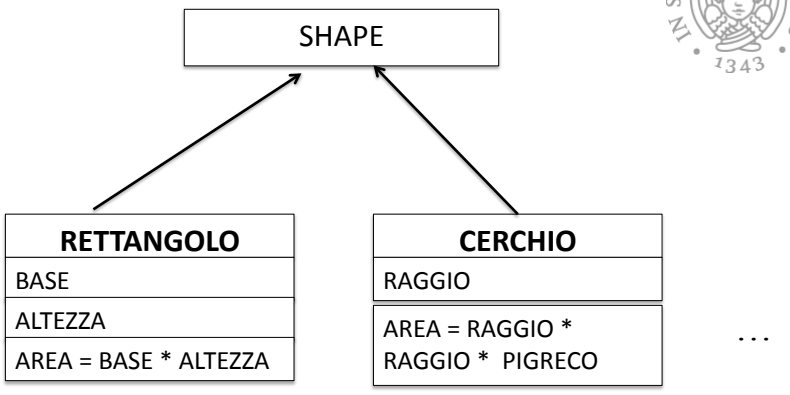
Object-oriented

```
double size() {
  double total = 0;
  for (Shape shape shapes) {
     total += shape.size();
  }
  return total;
}
```

Ogni forma geometrica e' responsabile del calcolo dell'area

Distribuiamo la computazione tra le strutture (oggetti)

```
public class Square extends Shape { ...
 public double size() {
  return width*width;
 }
}
```

Oggetto Shape: gestisce del contenuto informativo (dati) e fornisce un servizio (il calcolo dell'area)

Object Oriented programming

- Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. http://en.wikipedia.org/wiki/Object-oriented_programming)

15

OOP

- Due aspetti importanti
  - Informazione (strutture dati) contenuta nell'oggetto
  - Operazioni significative per operare sui dati dell'oggetto
- Nel mondo della Ingegneria del software (lo vedrete il prossimo anno) questi due aspetti si riflettono nella nozione di *Information Hiding*

16

Information Hiding

- information hiding is the principle of segregation of the design decisions in a computer program that are most likely to change, thus protecting other parts of the program from extensive modification if the design decision is changed. The protection involves *providing a stable interface which protects the remainder of the program from the implementation* (the details that are most likely to change).

17

Perche' e' importante?

- Esporre solamente l'informazione necessaria per operare
- Dichiarare "cosa si fa" non "come si fa" (separation of concerns)
- Decomposizione di un sistema in parti
- Protezione dei clienti dalla modifiche di implementazione
- Definizione di contratti di uso
- Facilita' nella manutenzione e evoluzione del software

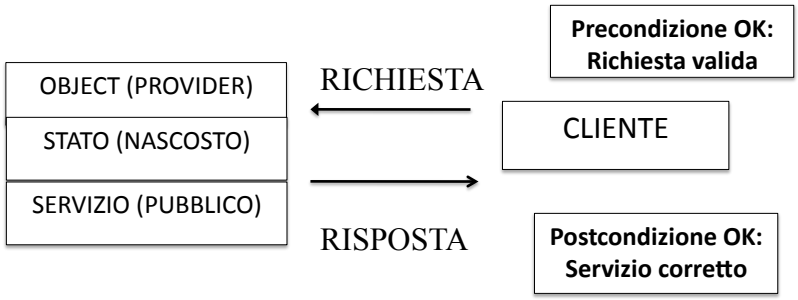
18

Cosa c'entra con programmazione II

- La programmazione "by contract" nello spirito OOP usando il linguaggio di programmazione Java come esempi

19

Programming by contract

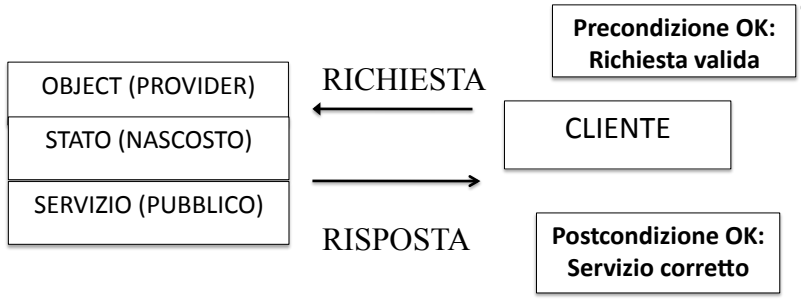


Un contratto definisce il vincolo del servizio:
La visione del: cliente richieste valide,
La visione del provider: fornire correttamente il servizio.

Cosa succede quando viene violato il contratto? **Exception**!!

20

Programming by contract



Contratto: Invariante di rappresentazione dello stato
Descrive quali sono gli stati validi dell'oggetto (alla costruzione, prima e dopo l'invocazione dei metodi pubblici

Cosa succede quando viene violato il contratto? Exception!!!

21

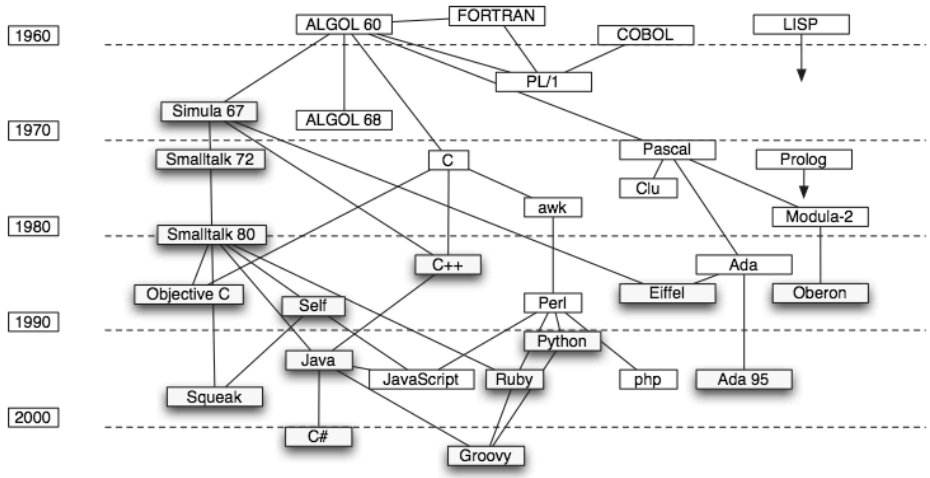
Java. Perche'?

- Modello a oggetti semplice e maneggevole
- Ogni entita' e' un oggetto
- Ereditarieta' singola
- Garbage collection
- Caricamento dinamico delle classi
- Librerie (viviamo in un mondo di API)
- Controllo di tipi statico e dinamico
- Meccanismi per la sicurezza
- Morale: poche novità, ma ragionevolmente pulito, semplice e fruibile

22

LINGUAGGI DI PROGRAMMAZIONE OO (VISIONE SEMPLIFICATA)

23



24

ASTRAZIONI

25

Meccanismi di astrazione

- Astrazione sui dati e' un esempio significativo di astrazione.
- Quali sono i meccanismi di astrazione legati alla programmazione:
  - lo strumento fondamentale è l'utilizzazione di **linguaggi ad alto livello**
  - enorme semplificazione per il programmatore
  - usando direttamente i costrutti del linguaggio ad alto livello invece che una delle numerosissime sequenze di istruzioni in linguaggio macchina "equivalenti"

26

I linguaggi non bastano

```
// ricerca all'insù
  found = false;
  for (int i = 0; i < a.length; i++)
     if (a[i] == e) {
          z = i; found = true;}
// ricerca all'ingiù
  found = false;
  for (int i = a.length - 1; i >= 0; i--)
     if (a[i] == e) {
          z = i; found = true;}
```

- sono diversi e possono dare risultati diversi
- potrebbero essere stati scritti con l'idea di risolvere lo stesso problema
  - verificare se l'elemento è presente nell'array e restituire una posizione in cui è contenuto

27

Migliori astrazioni nel linguaggio?

- il linguaggio potrebbe avere delle potenti operazioni sull'array del tipo **isIn** e **indexOf**

```
// ricerca indipendente dall'ordine
  found = a.isIn(e);
  if found z = a.indexOf(e);
```

- l'astrazione è scelta dal progettista del linguaggio
  - quali e quante quanto complicato diventa il linguaggio?
- meglio progettare linguaggi dotati di meccanismi che permettano di definire le astrazioni che servono

28

Il più comune tipo di astrazione

- ***l'astrazione procedurale*** presente in tutti i linguaggi di programmazione
  - definizione di procedure, chiamata di procedure
- la separazione tra "definizione" e "chiamata" rende disponibili nel linguaggio i due meccanismi fondamentali di astrazione
- **l'astrazione attraverso parametrizzazion** si astrae dall'identità di alcuni dati, rimpiazzandoli con parametri
  - si generalizza un parametro per poterlo usare in situazioni diverse (Shape definito in precedenza)
- **l'astrazione attraverso specifica** si astrae dai dettagli dell'implementazione della procedura, per limitarsi a considerare il comportamento che interessa a chi la procedura (ciò che fa, non come lo fa)
  - si rende ogni procedura indipendente dalle implementazioni dei moduli che la usano

29

Astrazione via parametrizzazione

- l'introduzione dei parametri permette di descrivere un insieme (anche infinito) di computazioni diverse con un singolo programma che le astrae tutte
- Il programma seguente descrive una particolare computazione

  x * x + y * y

- La definizione

  fun(x,y:int) = (x * x + y * y)

  descrive tutte le computazioni che si possono ottenere chiamando la funzione, cioè applicando la funzione ad una opportuna coppia di valori

30

Astrazione via specifica

- la nozione di procedura si presta a meccanismi di astrazione più potenti della parametrizzazione
- possiamo astrarre dalla specifica computazione descritta nel corpo della procedura, associando ad ogni procedura una specifica
  - semantica intesa della procedura
- derivando la semantica della chiamata dalla specifica invece che dal corpo della procedura
  - non è di solito supportata dal linguaggio di programmazione
  - se non in parte tramite le specifiche di tipo, come avviene nei linguaggi funzionali della famiglia di ML

- Si realizza con opportune annotazioni
  - Esempio Aspect Oriented Programming (AOP)
  - Esempio: JML

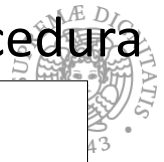
31

```
float sqrt (float coef) {
   // REQUIRES: coef > 0
   // EFFECTS: ritorna una approssimazione
   // della radice quadrata di coef
   float ans = coef / 2.0; int i = 1;
   while (i < 7) {
      ans = ans-((ans*ans-coef)/(2.0*ans));
      i = i+1;   }
   return ans;  }
```

- *precondizione* (asserzione requires)
  - deve essere verificata quando si chiama la procedura
- *postcondizione* (asserzione effects)
  - tutto ciò che possiamo assumere valere quando la chiamata di procedura termina, se al momento della chiamata era verificata la precondizione

32

Il punto di vista di chi usa la procedura



```
float sqrt (float coef) {
   // REQUIRES: coef > 0
   // EFFECTS: ritorna una approssimazione
   // della radice quadrata di coef
   …  }
```

- gli utenti della procedura non si devono preoccupare di capire cosa la procedura fa, astraendo le computazioni descritte dal corpo
  - cosa che può essere molto complessa
- gli utenti della procedura non possono osservare le computazioni descritte dal corpo e dedurre da questo proprietà diverse da quelle specificate dalle asserzioni
  - astraendo dal corpo (implementazione), si "dimentica" informazione evidentemente considerata non rilevante

33

Tipi di astrazione

- parametrizzazione e specifica permettono di definire vari tipi di astrazione
  - **astrazione procedurale**
    - ✓si aggiungono nuove operazioni
  - **astrazione di dati**
    - ✓si aggiungono nuovi tipi di dato
  - **iterazione astratta**
    - ✓permette di iterare su elementi di una collezione, senza sapere come questi vengono ottenuti
  - **gerarchie di tipo**
    - ✓permette di astrarre da specifici tipi di dato a famiglie di tipi correlati

34

Astrazione procedurale

- fornita da tutti i linguaggi ad alto livello
- aggiunge nuove operazioni a quelle fornite come primitive dal linguaggio di programmazione
  - Esempi che avete visto in C, e OCAML
- la specifica descrive le proprietà della nuova operazione

35

Astrazione sui dati

- fornita da tutti i linguaggi ad alto livello moderni
- aggiunge nuovi tipi di dato e relative operazioni
  - l'utente non deve interessarsi dell'implementazione, ma fare solo riferimento alle proprietà presenti nella specifica
  - le operazioni sono astrazioni definite da asserzioni logiche (ricordate le specifiche di LPP?)
- la specifica descrive le relazioni fra le varie operazioni
  - per questo, è cosa diversa da un insieme di astrazioni procedurali

36

Iterazione astratta

- permette di iterare su elementi di una collezione, senza sapere come questi vengono ottenuti
  - per esempio, potremmo iterare su tutti gli elementi di un MultiInsieme senza imporre nessun vincolo sull'ordine con cui vengono elaborati
- astrae (nasconde) il flusso di controllo nei cicli

37

Gerarchie di tipo

- fornite dai linguaggi ad alto livello moderni
  - per esempio, Ocaml, F#, Java, C#
- permettono di astrarre gruppi di astrazioni di dati (tipi) a famiglie di tipi
- i tipi di una famiglia condividono alcune operazioni
  - definite nel supertype, di cui tutti i tipi della famiglia sono subtypes
- una famiglia di tipi astrae i dettagli che rendono diversi tra loro i vari tipi della famiglia

38

Astrazione e programmazione orientata ad oggetti

- il tipo di astrazione più importante per guidare la decomposizione è l'astrazione sui dati
  - gli iteratori astratti e le gerarchie di tipo sono comunque basati su tipi di dati astratti
- l'astrazione sui dati è il meccanismo fondamentale della programmazione orientata ad oggetti
  - anche se esistono altre tecniche per realizzare tipi di dato astratti
    - ✓per esempio, all'interno del paradigma di programmazione funzionale

39