1

Mo Tu We Th Fr Sa Su

No. SISTEMI INF.

Date 29 · 11 · 19

CLASS DIAGRAM

Altro strumento di specifica

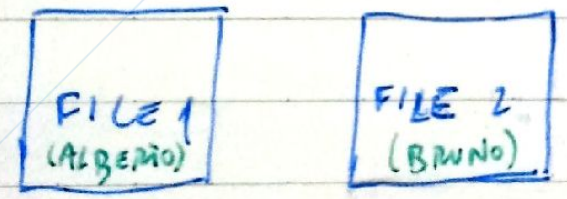
↓

dobbiamo prima introdurre

↓

PROGRAMMAZIONE AD OGGETTI

CODICE C → NON È OBJECT ORIENTED



ALBERTO

```
typedef struct {
    int giorno;
    int mese;   → char mese[20];
    int anno;
} data;
```

2

No. SISTEMI INF.

Date 29 · 11 · 18

```
void stampadata (DATA D){
  printf("%d.%d", D.GIORNO)
  if ( D.MESE=11)
    printf("GIORNO D");
    ...
```

C ci permette di fare questo

BRUNO

```
DATA X
X.GIORNO=29
X.MESE=11;
X.ANNO=2019;
  ...
  IF(X.GIORNO= ...)
  ....
```

Supponiamo ora che

il mese di Alberto nella struct diventi un array di caratteri:

↓

X.MESE=11 di BRUNO non funziona più

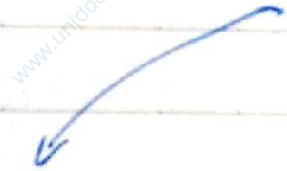
3

No. SISTEMI INF.

Date 29 11 19

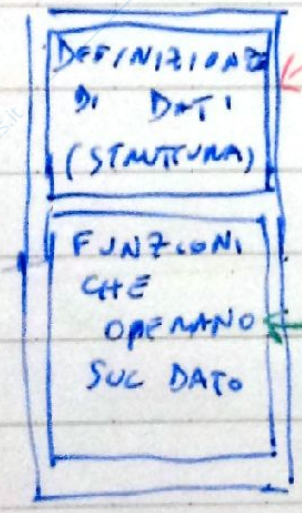
Dobbiamo considerare che se abbiamo una corrispondenza di file una modifica a uno di questo causa una modifica sugli altri file

PROBLEMA

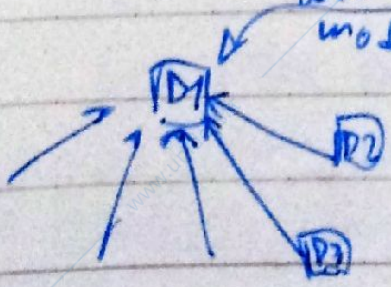


Si introduce quindi il concetto di dato astratto

DATO ASTRATTO = (anche chiamato classe) entità che contiene una struttura (definizione) di dati



non dobbiamo andare a modificare i file ad ogni cambiamento di un dato, ma solo andare a modificare le funzioni



4

Mo Tu We Th Fr Sa Su

No. SISTEMI INF.

Date 29 11 19

OGGETTO = ISTANZA DI UNA CLASSE

una variabile di un certo tipo o classe

Come possiamo modificare il codice e renderlo più object oriented?

ALBERATO

```
typedef struct {
        int GIORNO;
        int MESE;
        int ANNO;
    }

Void SET_GIORNO_INT(data D, int GIORNO){
    D.GIORNO = G;
}

Void SET_MESE_INT (data D, int M) {
    D.MESE = M;
}
```

Mo Tu We Th Fr Sa Su

No. SISTEMI INF.

Date 29 · 11 · 19

```
void SET_MESE_STRING (data D, char S, mese)|
    IF (MESE == 'GENNAIO')
      D.MESE = 1;
    IF . . .
}
void SET_ANNO
. . . . .
```

— funzioni per ogni variabile

```
int GET_MESE_INT(data D)|
    return D.MESE;
{
. . . .
```

Anche se stiamo C in questo modo C non è object oriented

✓ infatti

✗ supponiamo

BRUNO

```
DATA X;
SET_GIORNO_INT (X, 29)
SET_MESE_INT(X, 11);
SET_ANNO_INT (X, 2019);
IF (X.GIORNO==
```

sostituisco con GET_GIORNO_INT(X) ==

① DENTRO BRUNO

X.GIORNO

② NON È UN VERO E PROPRIO DATO ASTRATTO (data)

6

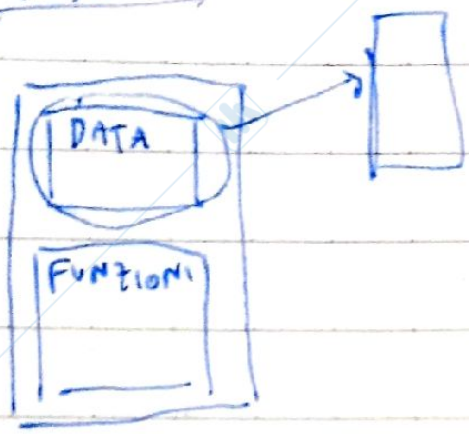
Mo Tu We Th Fr Sa Su

No. SISTEMI INF.

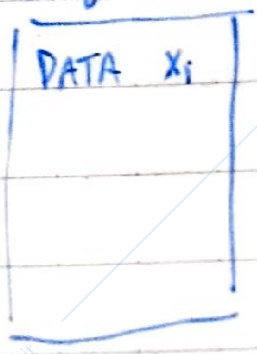
Date 29 . 11 . 19

Alloco solo DATA

ALBERO



BRUNO



BRUNO

DATA X_1, X_2

X_1 . SET_GIORNO_INT(29);	SET_GIORNO_INT(X_1, 29);
X_2 . SET_GIORNO_INT(30);	SET_GIORNO_INT(X_2, 30);

in C

prendi l'oggetto e usalo

dentro 29

funzione definita

all'interno dell'oggetto

Mo Tu We Th Fr Sa Su 7

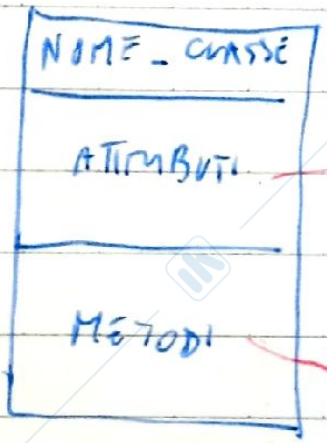
No. SISTEMI INF.

Date 29 · 11 · 19

CLASS DIAGRAM

Un grafico di specifica non banale può andare a suddividere in classi il progetto software

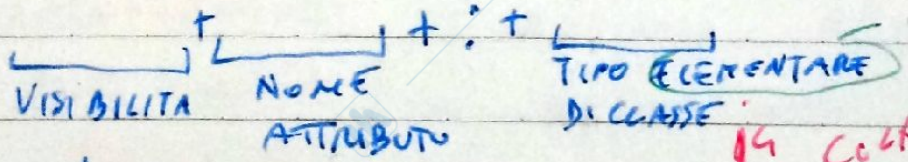
Si rappresenta dal punto di vista grafico una classe



→ sono gli elementi che costituiscono la struttura dati

→ funzioni che operano sulla struttura dati

Ogni singolo attributo va specificato come:



è un simbolo tra [+, -, #]

- Public
- Private
- protected → via di mezzo

Visibile dall'esterno

Non visib. dall'esterno

In certe situazioni l'attributo si comporta come public altre come private

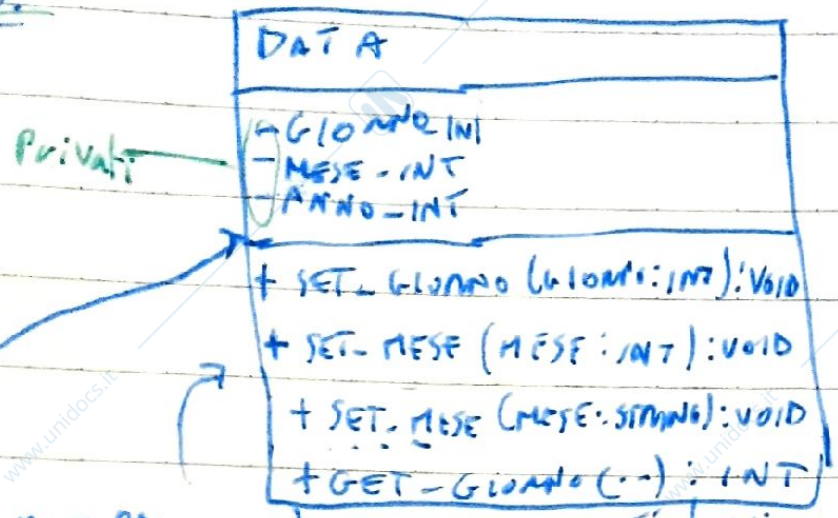
Mo Tu We Th Fr Sa Su

No. SISTEMI INF.

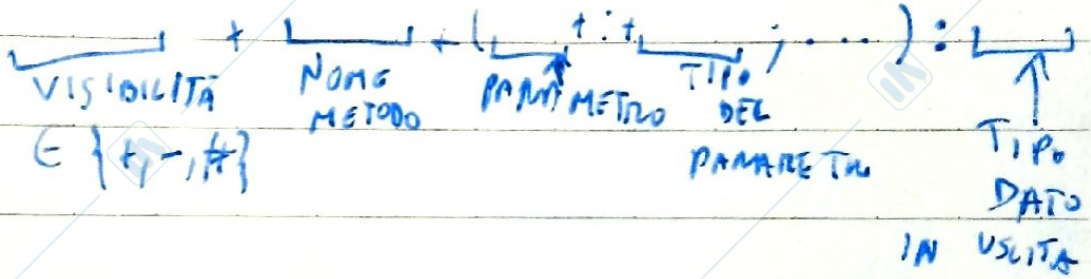
Date 29.11.19

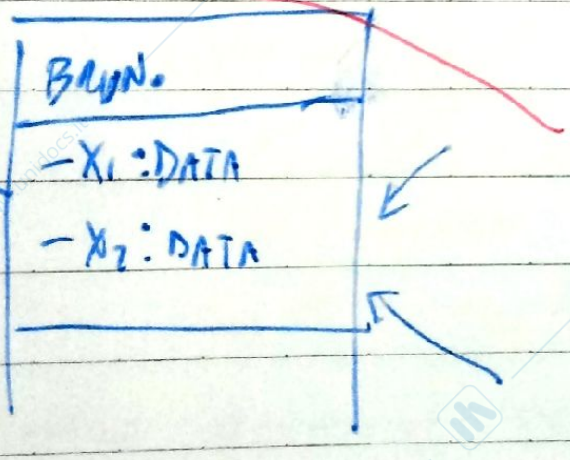
8

C).



I METODI hanno una sintassi simile agli ATTRIBUTI:





CLASS DIAGRAM con una freccia indica la dipendenza di una classe da un'altra classe

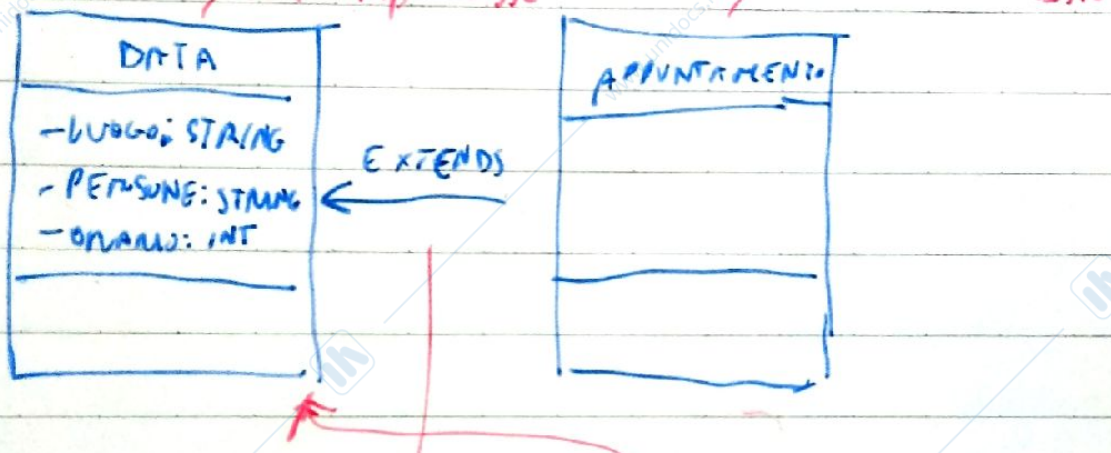
9

No. SISTEMI INF.

Date 29.11.19

- GENERAZIONE CODICE → tool che di partenza di alcune strutture del codice generata in automatico
- EXTENDS

Vuol dire che



prende questa classe e ci costruisce attorno come se l'avessi scritta

Se faccio extends e abbiamo attrib. o metodi # da appuntamento questi altri e metodi in data diventano public per APPUNTAMENTO

invece se una classe X usando DATA allora # attr. e metodi sono private per X