

Tema 1 Introducció a les estructures de dades

Dra. Maria Salamó Llorente

Estructura de Dades

Grau d'Enginyeria Informàtica

Facultat de Matemàtiques i Informàtica,

Universitat de Barcelona

Contingut

- 1.1 Introducció
- 1.2 Què és la ciència de la computació?
- 1.3 Què és la programació?
- 1.4 Característiques de la OO
- 1.5 Tipus Abstractes de Dades

1.1 Introducció

Perquè necessitem estructures de dades?

- Imaginem que ens contracten per fer un projecte en una Agència de Viatges i ens demanen realitzar la gestió informàtica de totes les dades amb les que treballa l'agència
- La nostra tasca seria:
 - **Emmagatzemar i representar totes les dades** amb les que treballa l'agència
 - **Gestionar les dades eficientment**
 - Buscar
 - Recuperar
 - Insertar
 - Eliminar, etc.
- Anirieu a una agència que treballa amb 100.000 hotels i per buscar un hotel, els mira tots un a un i triga un segon en consultar cada hotel?

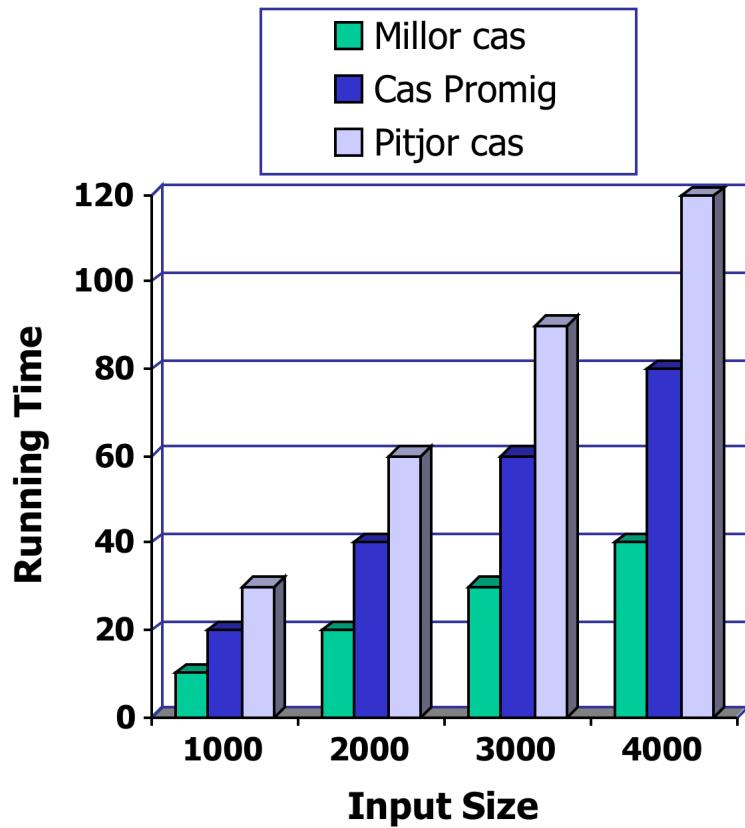
Quantes dades generem?

- Google: 100 milions de cerques al dia
- Wal*Mart: 20 milions de transaccions en un dia
- 1.000 milions de transaccions de targetes de crèdit per mes
- 3.000 milions de trucades diàries en USA
- 30.000 milions d' e-mails diaris, 1.000 milions de SMS
- Trànsit de xarxes IP: 1.000 milions de paquets per hora per router

**TOTES AQUESTES DADES S'HAN DE GUARDAR I
CONSULTAR !**

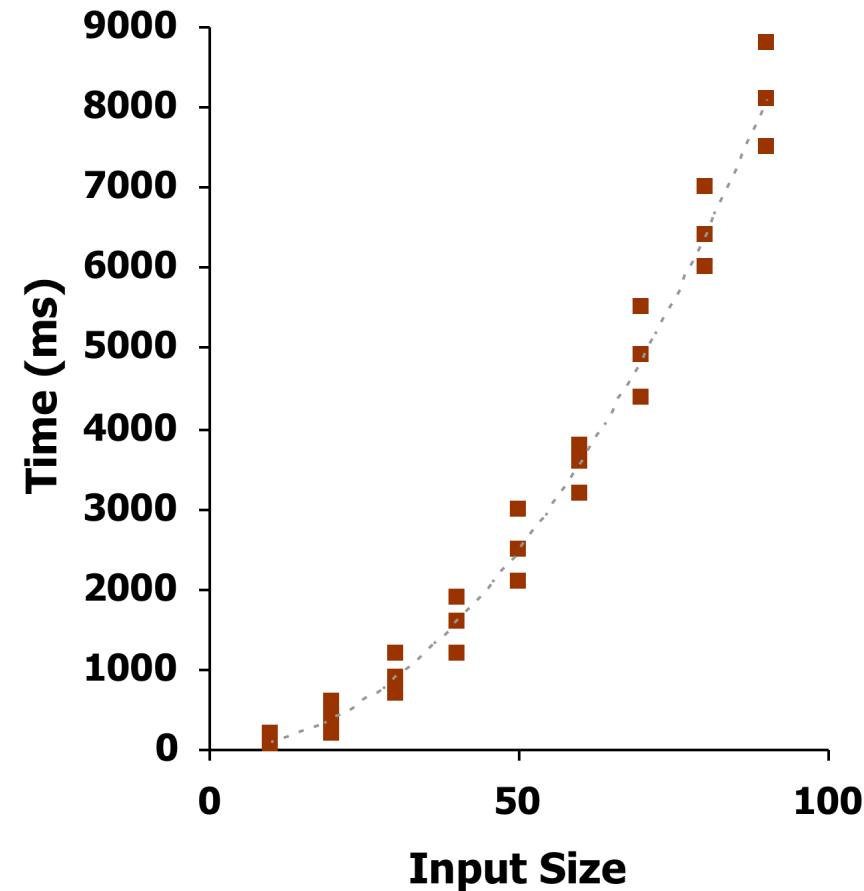
Quant de temps necessitem per processar les dades?

- La majoria d'algorismes transformen objectes d'entrada en objectes de sortida
- El temps de procés d'un algorisme **creix segons la mida** de les dades d'entrada
- El temps del cas promig és difícil de determinar
- Normalment s'enfoca des del **pitjor cas**
 - Més fàcil d'analitzar
 - Crucial per aplicacions com jocs, finances i robots



Experiments

- Escriu un programa implementant l'algorisme
- Executa el programa amb una entrada que canvii de mida i tingui diferent composició
- Usa un mètode com `clock()` per obtenir una mesura acurada del temps actual d'execució
- Mostra els resultats en un gràfic

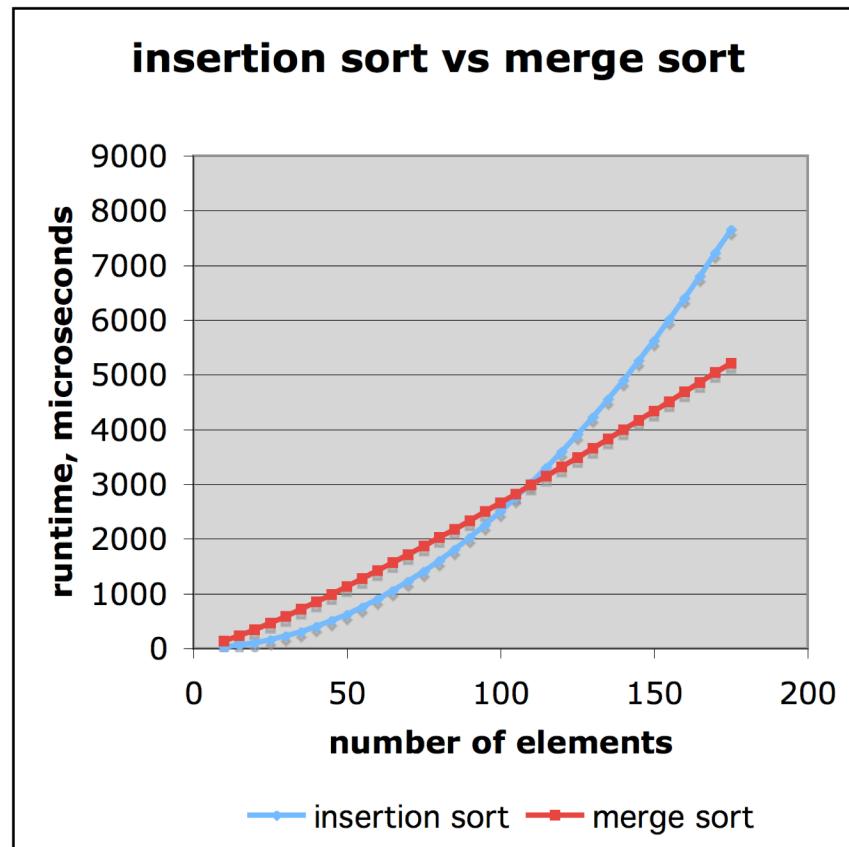


Limitacions dels experiments

- És necessari implementar l'algorisme, i pot ser difícil
- Els resultats no indicaran el temps per altres entrades no incloses en l'experiment
- Per poder comparar dos algorismes, s'ha d'usar el mateix entorn hardware i software



Comparativa de dos algorismes



insertion sort: $n^2 / 4$

merge sort: $2 n \log n$

Si ordenem un milió d'ítems?

insertion sort triga
aproximadament **70 hores**
mentre que
merge sort triga
aproximadament **40 segons**

En una màquina lenta, però podria
ser 100 més ràpida i llavors
trigaria 40 minuts
versus menys de 0.5 segons

Anàlisis teòric



- Usa una descripció a més alt nivell de l'algorisme enllot de la implementació
- Caracteritza el temps d'execució com una funció dependent de la mida de les dades, n .
- Considera totes les possibles entrades
- Permet avaluar la velocitat d'un algorisme independentment de l'entorn hardware o software

Pseudocodi

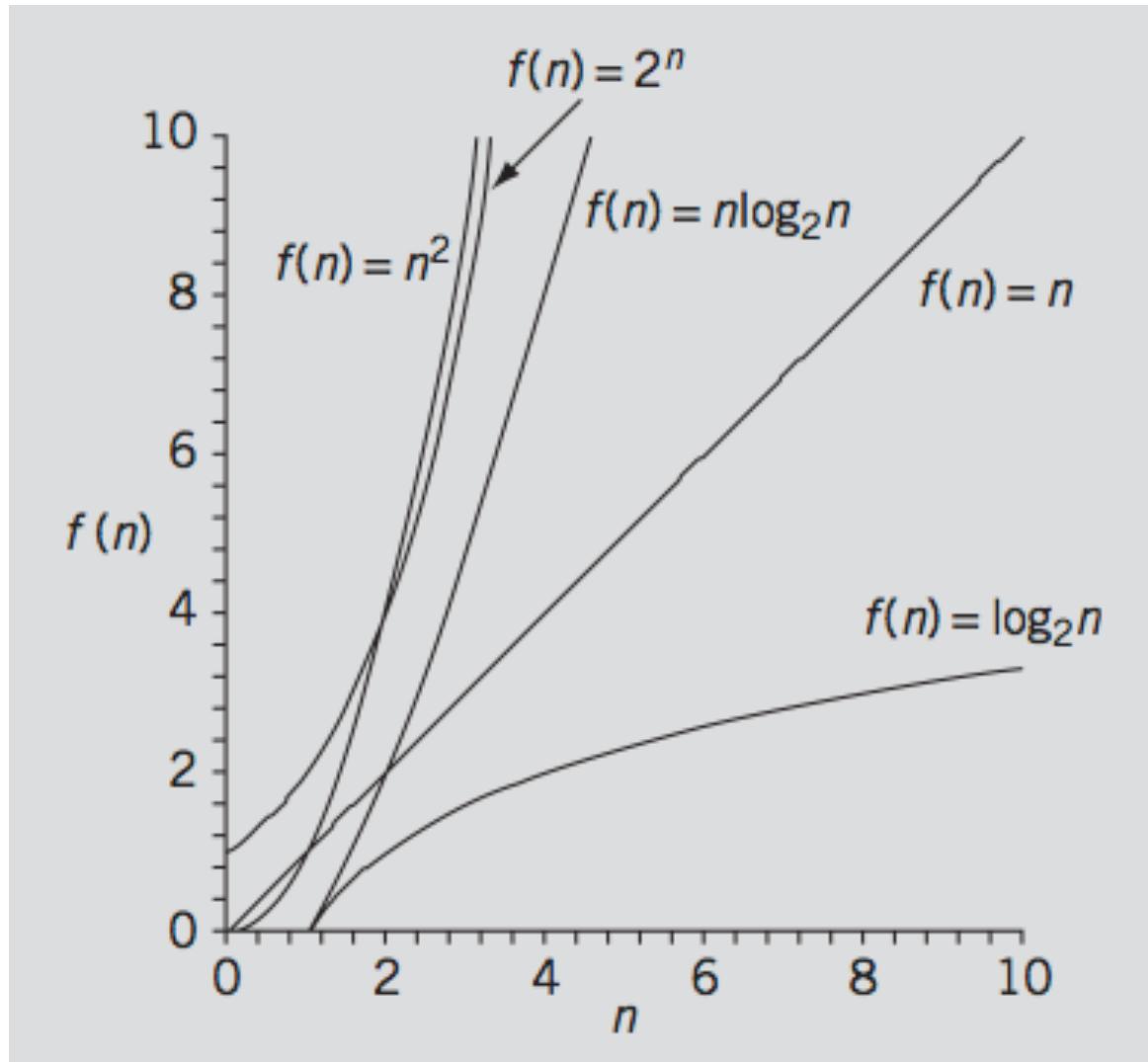
- Descripció a alt nivell d'un algorisme
- Més estructurat que la prosa en llenguatge natural
- Menys detallat que un programa
- Notació preferida per descriure algorismes
- Oculta els detalls en el disseny i anàlisi dels programes

Exemple: trobar l'element max d'un array

```
Algorithm arrayMax(A, n)
Input array A of n integers
Output maximum element of A

currentMax  $\leftarrow A[0]$ 
for i  $\leftarrow 1$  to n – 1 do
    if A[i] > currentMax then
        currentMax  $\leftarrow A[i]$ 
return currentMax
```

7 Functions Importants



- Set funcions que apareixen sovint en l'anàlisi d'algorismes:
 - Constant ≈ 1
 - Logarítmica $\approx \log n$
 - Lineal $\approx n$
 - N-Log-N $\approx n \log n$
 - Quadràtica $\approx n^2$
 - Cúbica $\approx n^3$
 - Exponencial $\approx 2^n$

Comparativa de les funcions

n	$f(n) = n$	$f(n) = \log_2 n$	$f(n) = n \log_2 n$	$f(n) = n^2$	$f(n) = 2^n$
10	0.01μs	0.003μs	0.033μs	0.1μs	1μs
20	0.02μs	0.004μs	0.086μs	0.4μs	1ms
30	0.03μs	0.005μs	0.147μs	0.9μs	1s
40	0.04μs	0.005μs	0.213μs	1.6μs	18.3min
50	0.05μs	0.006μs	0.282μs	2.5μs	13 days
100	0.10μs	0.007μs	0.664μs	10μs	4×10^{13} years
1000	1.00μs	0.010μs	9.966μs	1ms	
10,000	10μs	0.013μs	130μs	100ms	
100,000	0.10ms	0.017μs	1.67ms	10s	
1,000,000	1 ms	0.020μs	19.93ms	16.7m	
10,000,000	0.01s	0.023μs	0.23s	1.16 days	
100,000,000	0.10s	0.027μs	2.66s	115.7 days	

1.2 Què és la ciència de la computació?

Ciència de la Computació

• El seu objectiu **NO** és: estudiar els ordinadors!

"Els ordinadors són per a la informàtica el que són els telescopis per l'astronomia" – Edsger Dijkstra (1930 - 2002)

Dijkstra fou un científic informàtic holandés que va rebre el premi Turing per les seves contribucions fonamentals en el desenvolupament de llenguatges de programació al 1972



- La ciència de la computació estudia:
 - Com resoldre problemes (independentment de les màquines)
 - Com estudiar problemes sense solucions
 - Com fer els problemes computacionals (desenvolupar els algorismes)
 - Com usar els ordinador com a mitjà per resoldre problemes

Àrees en la ciència de la computació

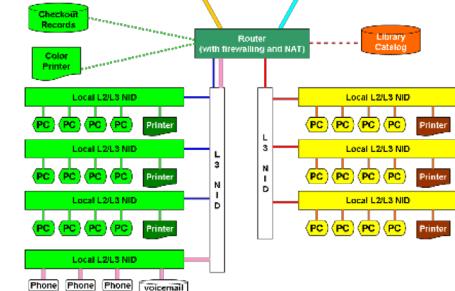
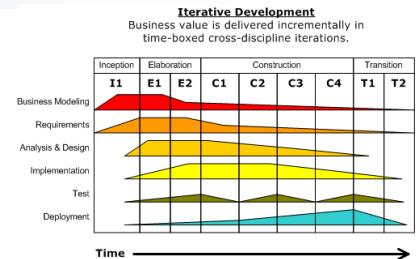
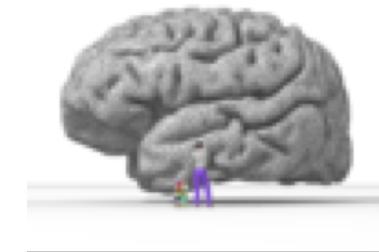
La Ciència de la Computació se centra en respondre a les preguntes fonamentals sobre el que es pot calcular de forma computacional i quina quantitat de recursos són necessaris per a fer aquests càlculs

- **4 grans àrees:**

- La teoria de la computació
- Algorismes i estructures de dades
- Metodologia de programació i llenguatges
- Elements informàtics i arquitectura

Altres àrees

- Enginyeria del Software
- Intel·ligència Artificial
- Processament del Llenguatge Natural
- Aprendentatge automàtic
- Visió per computador
- Gràfics per ordinador
- Factors humans
- Xarxes



1.3 Què és la programació?

Què és la programació?

- La programació és el procés de codificar un algorisme en un llenguatge de programació per a que sigui executable
- La ciència de la computació \neq programació
- Els algorismes expressen la solució a través de:
 - Els passos necessaris per obtenir el resultat
 - La representació de les dades (tipus de dades)
- Els tipus de dades permeten interpretar les cadenes de bits representades en l'ordinador
- La complexitat dels problemes porta a necessitar tipus de dades complexes i sofisticats

- Orientació a Objectes (OO) consisteix en organitzar el software com una col·lecció discreta d'objectes que incorporen tant les dades com el comportament.
- La OO estructura el software en dades i el conjunt d'operacions associades a aquestes dades, en unes entitats anomenades classes.
- Un programa és un conjunt d'objectes (que pertanyen a diferents classes) que interactuen entre si per tal de resoldre el problema.

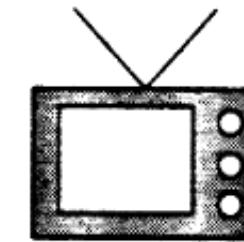
- La OO és una filosofia, no està lligada a cap llenguatge de programació
- Java i C++ són llenguatges de programació orientat a objectes (POO)

variable name	address
aCredit	10000007
aDebit	13537163
anAccount	56826358
aSavingsAccount	45205128

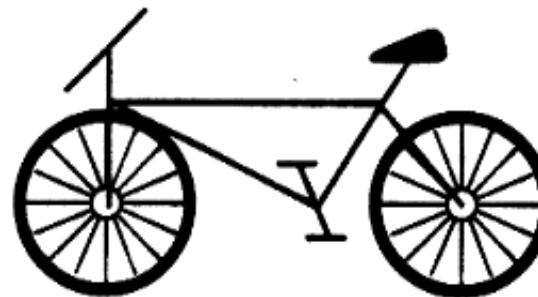
a symbol table



a binary tree



the gray television



Mike's bicycle



Brian's bicycle



a white rook

Objectes, classes i mètodes

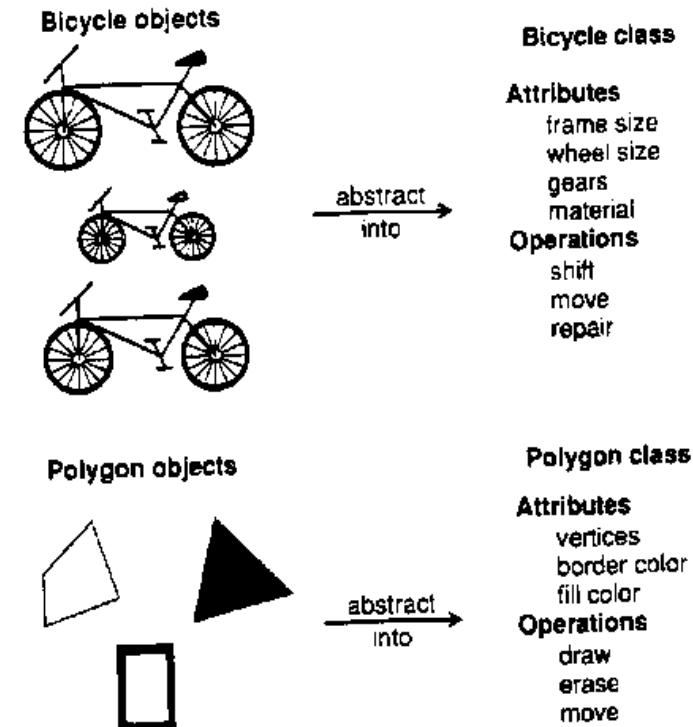
Objectes

- Els objectes es poden usar per representar entitats del món real
- Un objecte està format per:
 - Un conjunt de dades (**atributs o estat**)
 - Un conjunt d'operacions (**mètodes o comportament**)
- El comportament d'un objecte pot modificar el seu estat
- Cada objecte té un identificador únic pel qual pot ser referenciat.

Objectes, classes i mètodes

Classes

- Una **classe** és una abstracció d'un cert concepte.
- Un **objecte** es defineix mitjançant una classe.
- Es diu que un objecte és una **instància** d'una classe.
- La classe representa un concepte i l'objecte representa la materialització d'aquest concepte.



Objectes, classes i mètodes

Classes

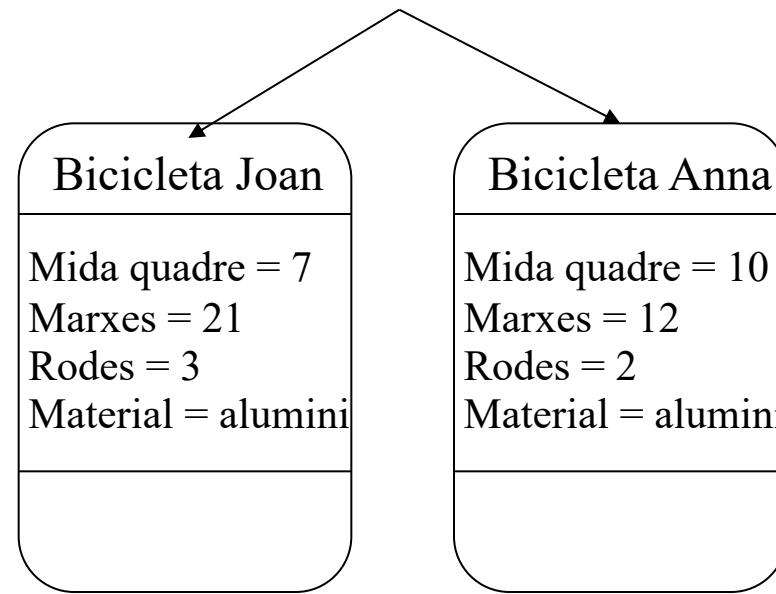
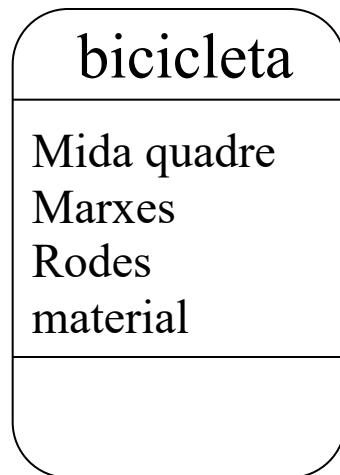
Una classe és un pla (o pla d'obra) d'un objecte

- La classe utilitza els mètodes per definir els comportaments de l'objecte.
- Es poden crear múltiples objectes d'una mateixa classe.
- Els objectes comparteixen el nom dels atributs i les operacions però cada objecte té un valor concret per cada atribut.
- La classe que conté el mètode **main** d'un programa JAVA o C++ representa el programa complet.

Objectes, classes i mètodes

Objectes i Classes

Múltiples casos de la mateixa classe



Objectes, classes i mètodes

Objectes i Classes

Una classe
(el concepte)



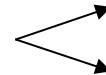
Un objecte
(la materialització,
la realització)

John's Bank Account
Balance: \$5,257

Bill's Bank Account
Balance: \$1,245,069

Mary's Bank Account
Balance: \$16,833

Múltiples objectes
de la mateixa classe



1.4 Característiques de la OO

Característiques de la OO

Abstracció

- Consisteix en agafar una informació i extreure'n les característiques més representatives

Encapsulament

- Amaga a l'usuari la implementació interna de l'objecte

Característiques de la OO

Herència

- Defineix una relació entre classes.
- Una classe (**superclasse**) defineix un conjunt de propietats comuns a altres classes (**subclasses**).
- Les classes es poden organitzar en jerarquies

Polimorfisme

- Canvi de comportament d'una operació dependent de l'objecte al qual s'aplica.

1.5 Tipus Abstractes de Dades (TAD)

- **Definició:**
 - **“Un TAD és un ens tancat i autosuficient, que no requereix d'un context específic perquè pugui ser utilitzat en un programa, la qual cosa garanteix portabilitat i reutilització del programari i minimitza els efectes que puguin produir un canvi a l'interior del TAD”**
 - Ocultar la representació del tipus de dades respecte de les aplicacions es coneix com a **encapsulament de les dades**
 - Aquesta manera de treballar té l'avantatge de la **reutilització de codi**
 - La implementació d'un TAD es compona de:
 - Les variables necessàries per definir les dades
 - Les rutines d'accés a aquestes variables

- **Exemple**

→ El TAD conjunt i els elements que el constitueixen (dades), té les operacions: AFEGIR ELEMENT, ELIMINAR ELEMENT, UNIÓ, INTERSECCIÓ i DIFERÈNCIA DE CONJUNTS

- **Especificació genèrica d'un TAD:**

- Nom del TAD
- Objecte abstracte
- Restriccions
- Llista d'Operacions
- Definició de cada operació

- **Exemple:**

- TAD Matriu
- Objecte abstracte:

		$x_{i,j}$	

- Restriccions: $n > 0, m > 0$
- Llista d'Operacions :
 - CrearMatriu(i, j)
 - AssignarMatriu($M, i, j, \text{ valor}$)
 - ObtenirDadaMatriu(M, i, j)
 - SumaMatriu(M_1, M_2)
 - MatriuNegativa(M)
 - RestarMatriu(M_1, M_2)
 - MatriuInversa(M)
 - MostrarMatriu(M)

Definició d'una classe

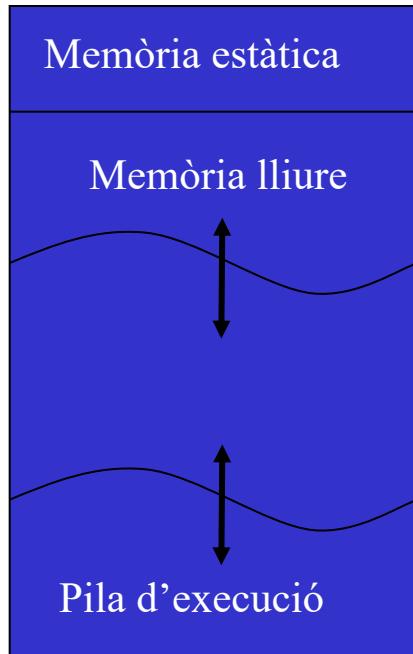
- Una classe és un tipus de dades especial (TAD) que defineix com construir un cert tipus d'objecte
- La "**classe**" també emmagatzema la part de les dades que són compartides per totes les instàncies d'aquesta classe
- "**Instàncies**" són objectes que es creen i que segueixen la definició donada en l'interior de la classe

Memòria

- Memòria d'emmagatzematge secundari
 - Les estructures de dades que s'emmagatzemen aquí les hi crida estructures de dades externes
- Memòria principal
 - Estructures de dades internes
- Classificació d'Estructures de Dades segons tipus de memòria:
 - Parlem d'una **Estructura de dades estàtica** quan se li assigna una quantitat fixa de memòria per a aquesta estructura abans de l'execució del programa. La quantitat d'espai assignat per a la memòria estàtica es determina durant la compilació i no varia
 - Parlem d'una **Estructura de dades dinàmica** quan se li assigna memòria a mesura que és necessitada, durant l'execució del programa. En aquest cas la memòria no queda fixa durant la compilació. L'acte d'assignar memòria durant l'execució es diu **assignació dinàmica de la memòria**

Memòria

- El sistema de la computadora manté una pila d'execució, la quantitat de la qual d'espai que ocupa, variarà durant l'execució del programa
 - Cada vegada que un procés és invocat en un programa, es crea un registre d'activació i s'emmagatzema en la pila d'execució.
 - El registre conté espai per a totes les variables declarades en un procediment i una còpia als paràmetres que estan sent passats a un procediment. A més de la indicació per continuar on ha de seguir l'execució del programa, una vegada es completi la funció
- La memòria dinàmica s'emmagatzema en la memòria lliure, la que creix cap a la zona baixa
- Es produeixen errors quan:
 - Massa memòria és assignada dinàmicament
 - Es creen massa registres d'activació



Tema 1 Introducció a les estructures de dades

Grau d'Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica,
Universitat de Barcelona