PROGRAMACIÓ ORIENTADA A OBJECTES: HERÈNCIA I POLIMORFISME EN C++

CONSTRUCCIÓ DE JERARQUIES DE CLASSES

ASSIGNATURA EDDO, GRESAUD

FATOS XHAFA, DEPT CS/ESEIAAT

CONTINGUTS DEL TEMA

1. CONCEPTE D'HERÈNCIA EN ORIENTACIÓ A OBJECTES:
   - D'ON SORGEIX
2. TIPUS D'HERÈNCIA
   - HERÈNCIA SIMPLE, MULTIPLE

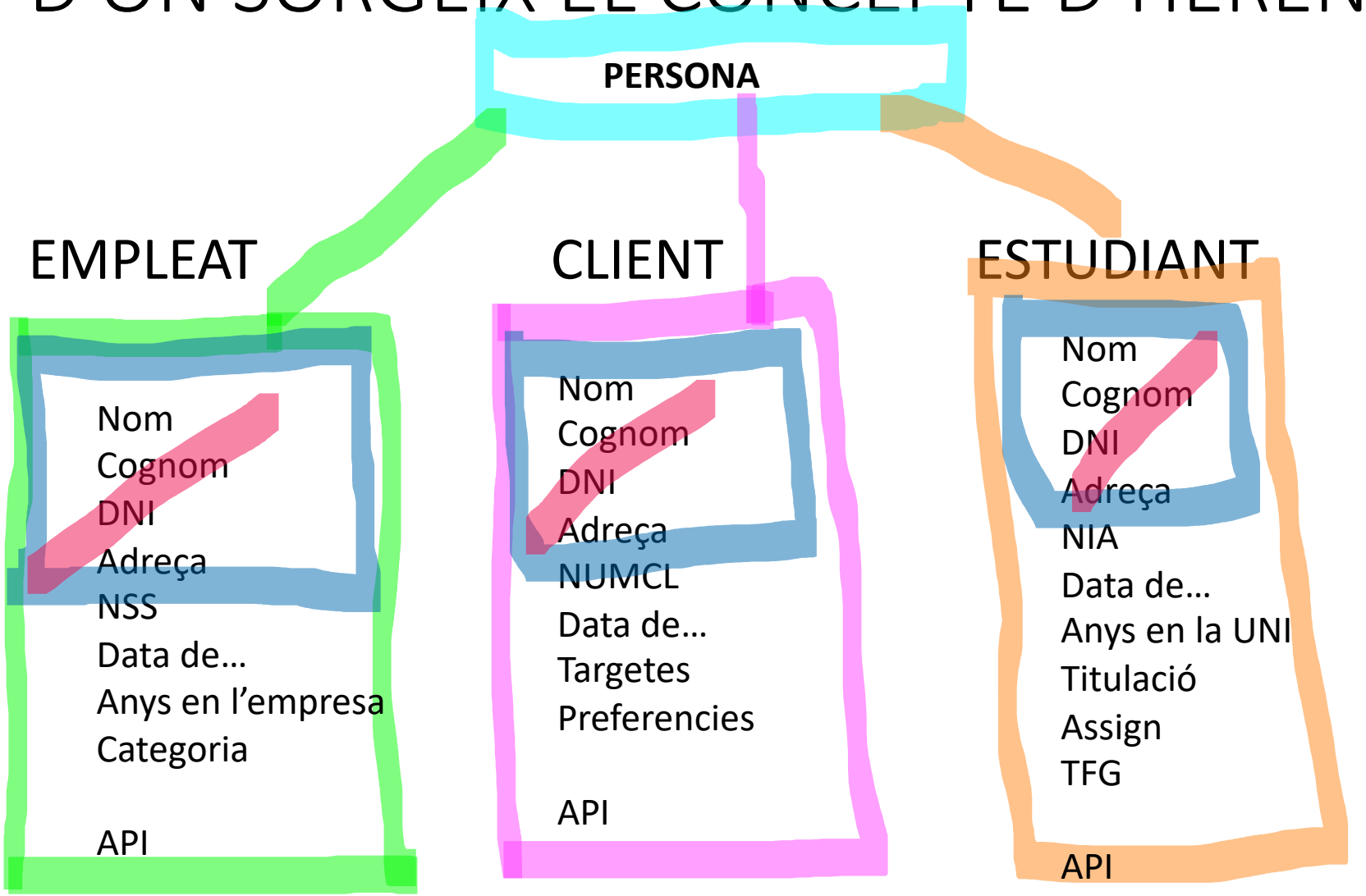
   ----------C++----------
3. SINTAXI D'HERÈNCIA EN C++
4. ACCÉS A LES DADES, DADES *PROTECTED*
5. CONSTRUCTORS I DESTRUCTOR
6. EXEMPLE: CLASSES DE JERARQUIA DE VEHICLES

   ---
7. MÈTODES VIRTUALS
8. POLIMORFISME

CONCEPTE D'HERÈNCIA EN ORIENTACIÓ A OBJECTES

- Fins are hem vist classes en C++ com a entitats separadament:
  - Cada classe té les seves dades i el seu API
- En general, en models reals, els objectes estan inter-relacionats (tenen un *solapament* en la informació). Considereu:
  - Figures geomètriques
  - Treballadors d'una empresa
  - Clients d'una empresa
  - Continguts audiovisuals
  - Vehicles
- Vegem les relacions que hi ha entre els objectes de cada cas.
- De pas, comentem el llenguatge diagramàtic UML (UNIFIED MODELLING LANGUAGE –*LLENGUATGE UNIFICAT DE MODELAT*)

D'ON SORGEIX EL CONCEPTE D'HERÈNCIA?



QUÈ GUANYEM?

LES DADES I L'API DE LA CLASSE PERSONA, NO CAL REPETIR EN LES CLASSES EMPLEAT, CLIENT I ESTUDIANT (***REUSE***)

HERÈNCIA ÉS UNA RELACIÓ “ÉS-UN” (“IS-A”)

- **Vehicle classe base**
  - ✓Autobus és un Vehicle
  - ✓Camió és Vehicle
  - ✓Cotxe és un Vehicle
  - ✓Ambulància és un Vehicle
  - X ~~Xofer NO és un Vehicle~~
- **Classe VehicleAeri**
  - ✓Avió és un VehicleAeri
  - ✓Drone és un VehicleAeri
  - ✓Helicopter és un VehicleAeri
- **Llibre**
- **Autor**
- ~~Llibre és un Autor ?? NO~~
- ~~Autor és un Llibre?? NO~~

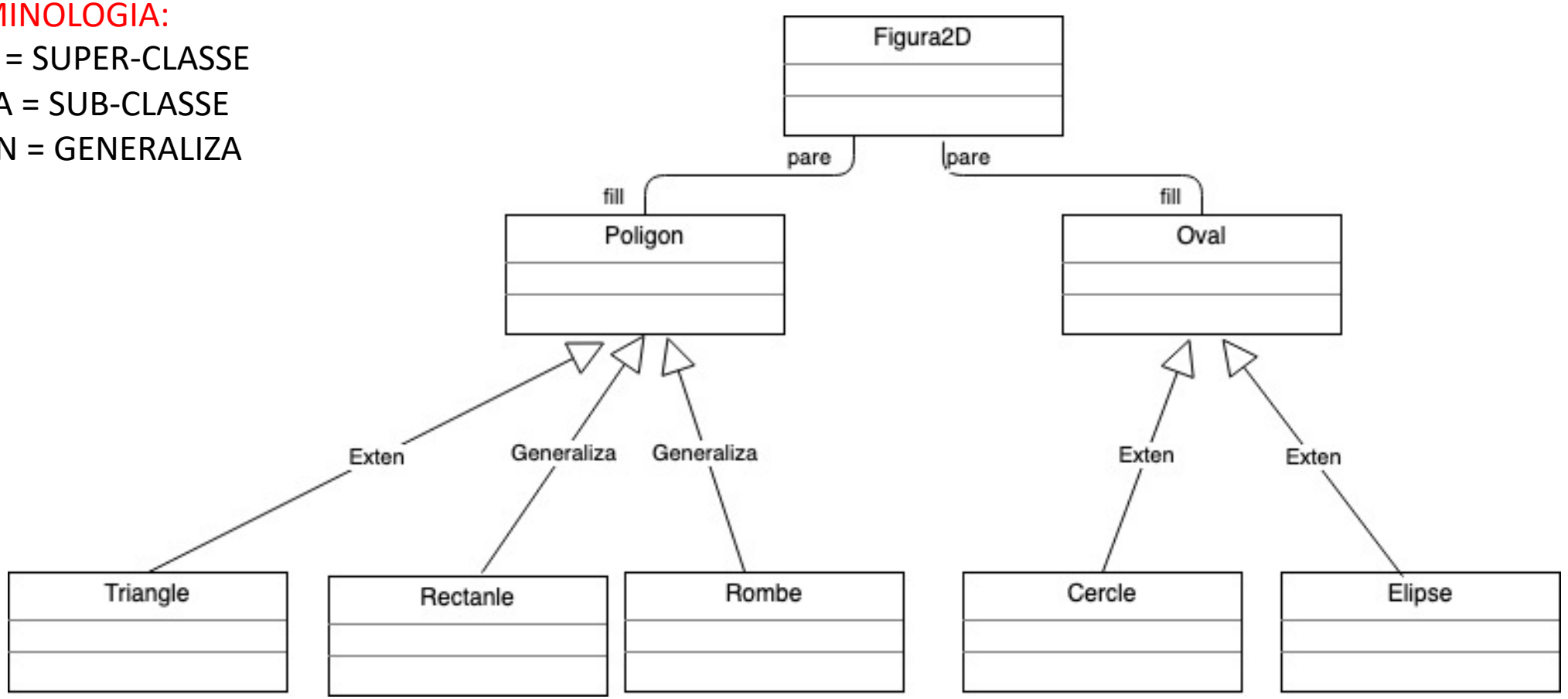
- **Classe Publicació**
  - ✓Llibre és una publicació
  - ✓Revista és un Publicació
  - ✓Diari és una Publicació
- **classe Persona //Classe base**
  - Autor és una Persona
  - Metge és una Persona
  - Pacient és una Persona
  - Infermer és una Persona
  - Lector és una Persona
  - Client és una Persona
  - ✓Treballador és una Persona

Altres exemples

- Vacuna //classe base
  - ✓VacunaAdenovirus
  - ✓Vacuna mRNA
- Aula //Classe base
  - ✓AulaLab
  - ✓Auditori
  - ✓AulaPisarra
- Publicació //Classe base
  - ✓Llibre
  - ✓Revista
  - ✓Diari
  - ✓…

FIGURES GEOMÈTRIQUES 2D

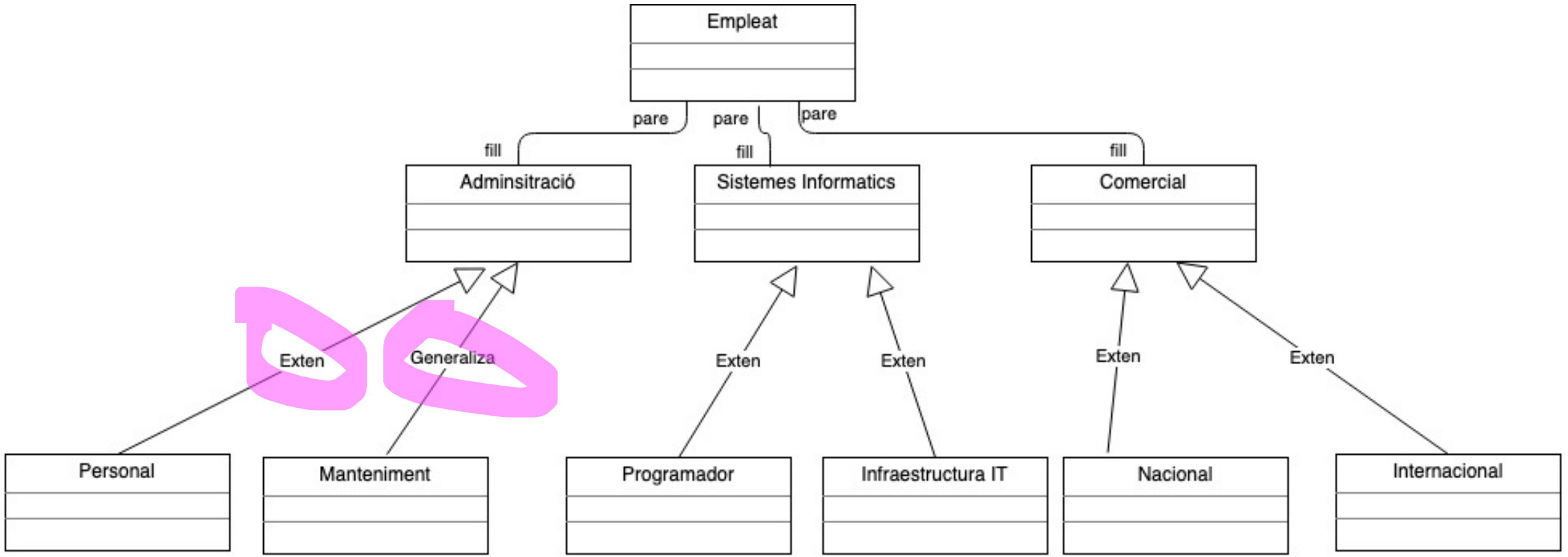
TERMINOLOGIA:
PARE = SUPER-CLASSE
FILL/A = SUB-CLASSE
EXTEN = GENERALIZA



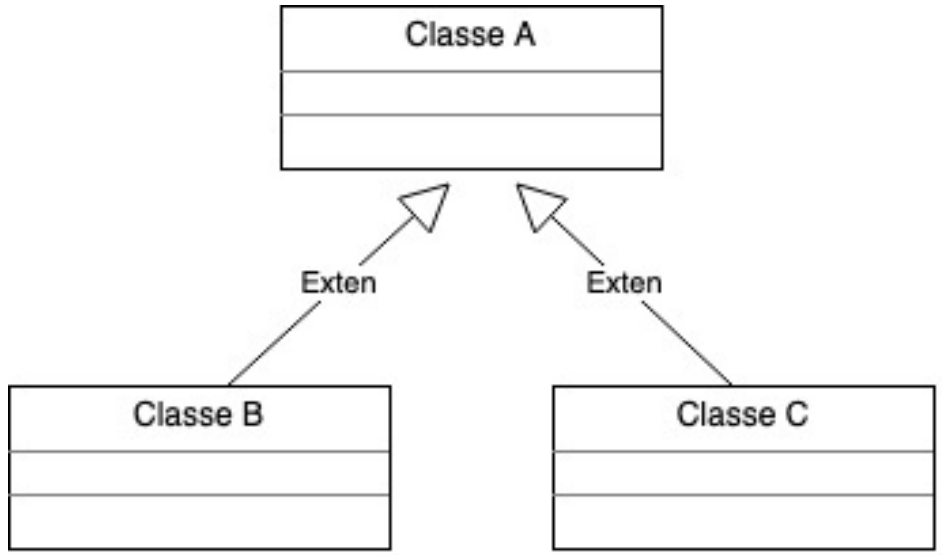
+ general

- general

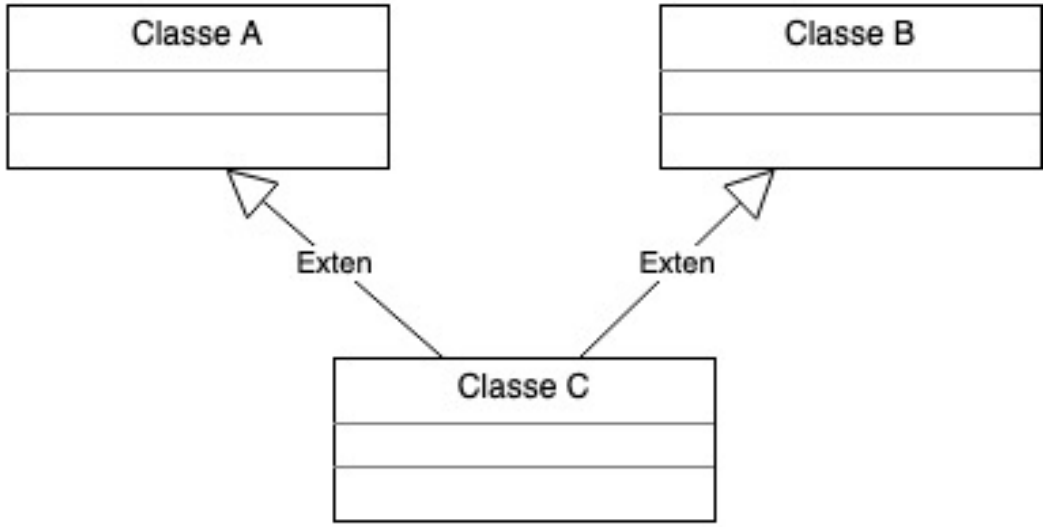
TREBALLADORS D'UNA EMPRESA



TIPUS D'HERÈNCIA: HERÈNCIA SIMPLE i MULTIPLE



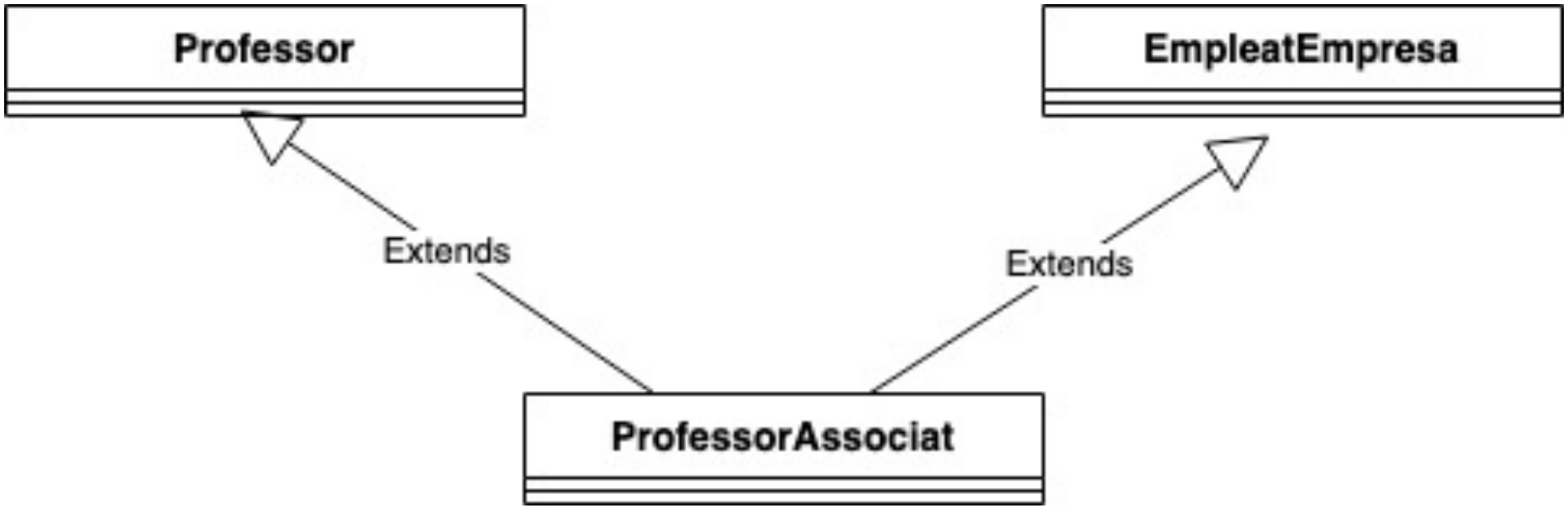
HERÈNCIA SIMPLE



HERÈNCIA MULTIPLE

Professor Associat (a Temps Parcial):

És un professor universitari i un empleat d'una empresa (no universitat)



SINTAXI D'HERÈNCIA SIMPLE EN C++

- Classe B (classe filla, o sub-classe) hereta de la classe A (classe pare/mare o super-classe)

```
class A
{
 …
…
};
```

```
class B : public  A {
            Implementació de la classe B
//LA CLASSE B NO REPETEIX NI DADES NI API DE LA CLASSE A
//JA QUE FORMEN PART DE LA CLASSE B A TRAVÉS DE L'HERÈNCIA
            …
            …
};
```

- TERMINOLOGIA:
  - Classe B: classe filla, sub-classe, classe derivada
  - Classe A: classe pare/mare, super-classe, classe base

class Comercial : public Empleat{…}

ACCÉS A LES DADES, DADES *PROTECTED*

- Les classe filles NO tenen accés als membres privats de les classes pare. Han d'accedir a través de l'API
- Per tal de garantir accés a les dades de la classe pare, els membres s'han de declarar *protected* (com veurem amb els exemples).

```
class A
{

   int a; //PRIVAT

   protected: //DÓNA ACCÉS A LES SUBCLASSES
       int b;
 …
…
};
```

```
class B : public A {
            Implementació de la classe B
            …
            …
};

//Mètodes de la classe B no poden accedir la dada a
//Mètodes de la classe B poden accedir la dada b
```

IMPLEMENTACIÓ DE CLASSES VIA HERÈNCIA: CONSTRUCTORS

- PRIMER S'APLICA EL CONSTRUCTOR DE LA CLASSE PARE, DESPRÉS ES COMPLETA LA CONSTRUCCIÓ DE LA PART DE LA CLASSE FILLA

nom_classe_filla(paràmetres) : nom_classe_mare(paràmetres)

{...resta d'implementació de l'objecte de la classe filla}

```
class A
{
   int a;
   string s;
public:
   A(){ a = 0;}
   A(int pa,string ps){
        a = pa;
        s = ps;
}

};
```

```
class B : public A {
     int c;
     double x;

public:
    B(){} //per la part de classe A crida el seu constructor per defecte
   B(int pa, string ps,int pc,double px) : A(pa,ps)
   {
       c = pc;
       x = px;
   }

};
```

CONSTRUCTOR DE LA classe A

RESTA DEL CONSTRUCTOR DE LA classe B

IMPLEMENTACIÓ DE CLASSE VIA HERÈNCIA: DESTRUCTORS

- PER DESTRUIR UN OBJECTE DE LA CLASSE FILLA:
  - PRIMER S'APLICA EL DESTRUCTOR DE LA CLASSE FILLA (DE LA SUB-CLASSE)
  - DESPRÉS ES CRIDA EL DESTRUCTOR DE LA CLASSE PARE (DE LA SUPER-CLASSE)

```
class A
{
   int* a;
   int N;
public:
  A(){ }
  A(int N){
     this -> N = N;
     a = new int[N];
      …
      …
}
~A()
{
  delete[] a;
}
…
};
```

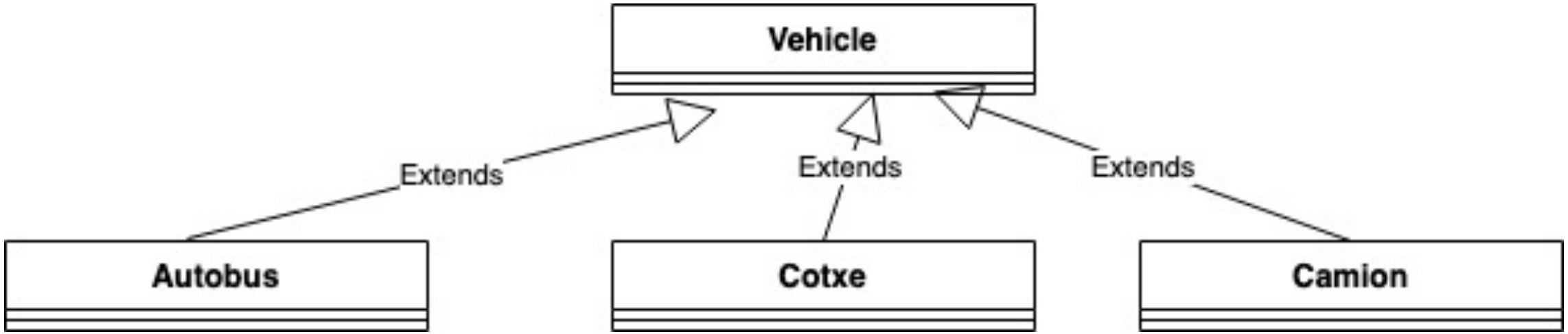
```
class B : public A {
   double* x;
   int M;
public:
   B(){} //per la part de classe A crida el seu constructor per defecte
  B(int N, int M) : A(N)
  {
     this -> M = M;
     x = new double[M];
    …
  }
~B()
{
  delete[] x;
}

};
```

CONSTRUCTOR DE LA classe A

RESTA DEL CONSTRUCTOR DE LA classe B

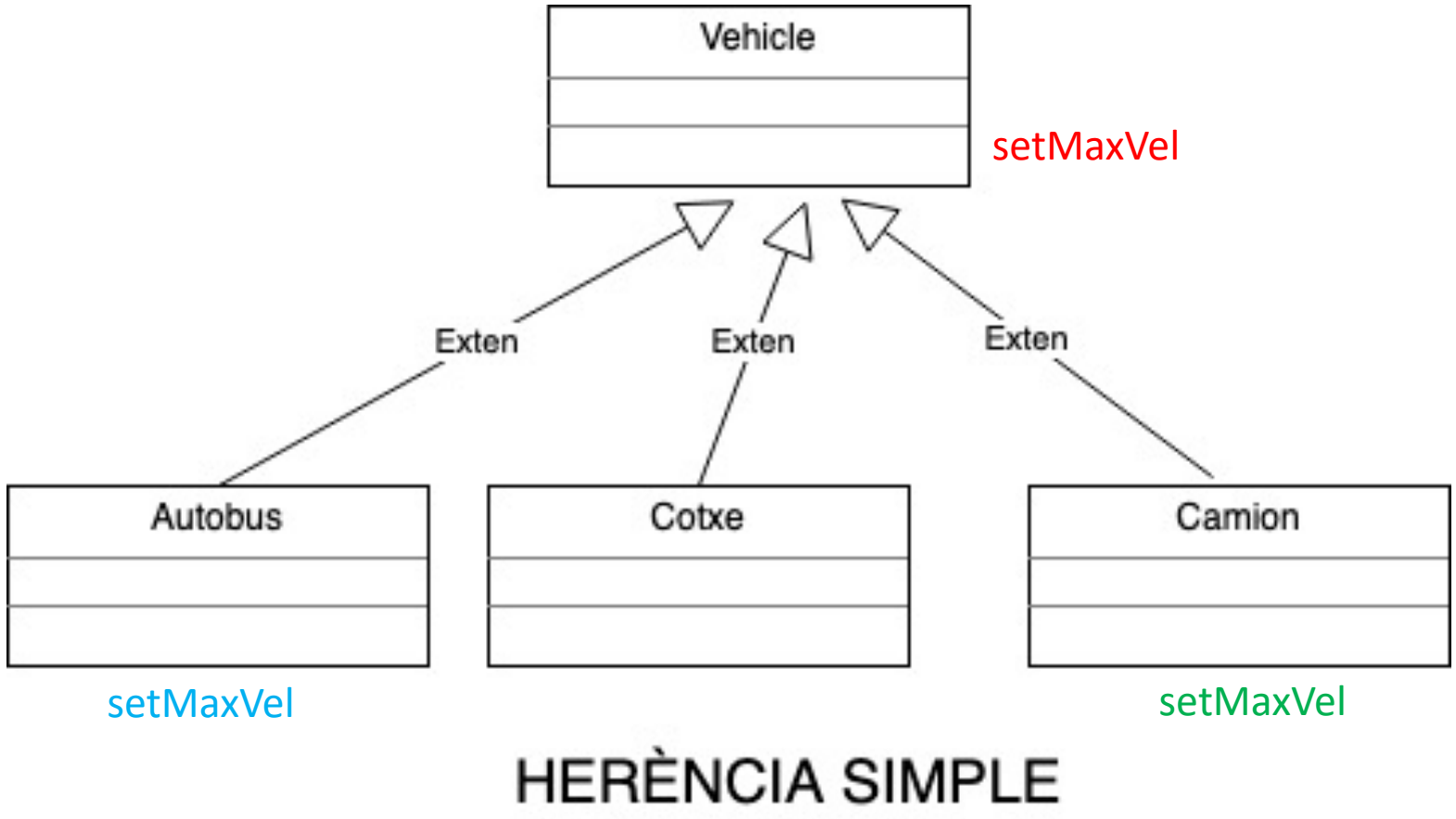
DESTRUCTOR DE LA classe B

PROGRAMA D'EXEMPLE D'HERÈNCIA SIMPLE: JERARQUIA VEHICLES



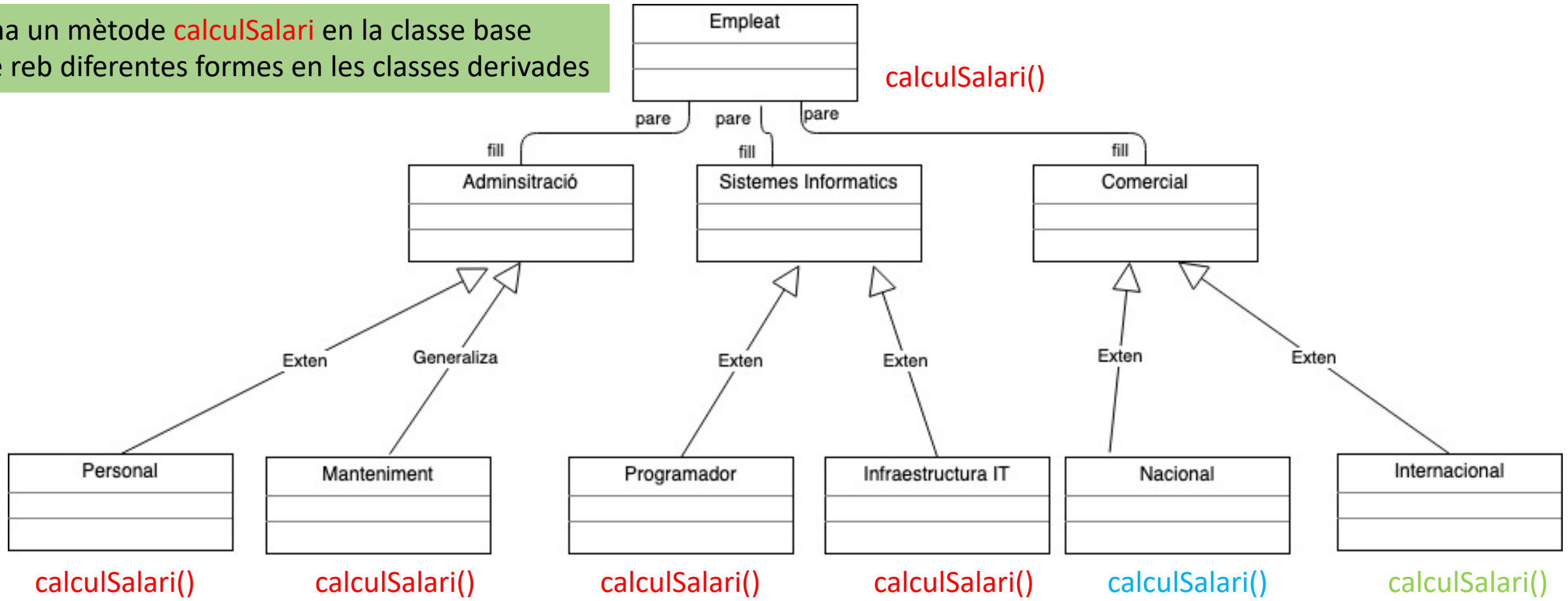
IMPLEMENTACIÓ DE CLASSES VIA HERÈNCIA: PROGRAMA D'EXEMPLE

- **Vehicle**:
  - NumBastidor
  - Matricula
  - Data Fin assegurança
  - Data Fin ITV
  - Velocitat màxima permesa
- **Autobus**:
  - Les dades d'un vehice
  - Capacitat passatgers
  - Tipus servei (urbà, etc.)
- **Camion**:
  - Les dades d'un vehice
  - TARA
  - Tipus mercaderia, perrillosa?
- **Cotxe**
  - Les dades d'un vehice
  - Etc.
- VEGEU SOLUCIÓ EN ATENEA



TREBALLADORS D'UNA EMPRESA: POLIMORFISME (IDEA)

Hi ha un mètode calculSalari en la classe base
que reb diferentes formes en les classes derivades



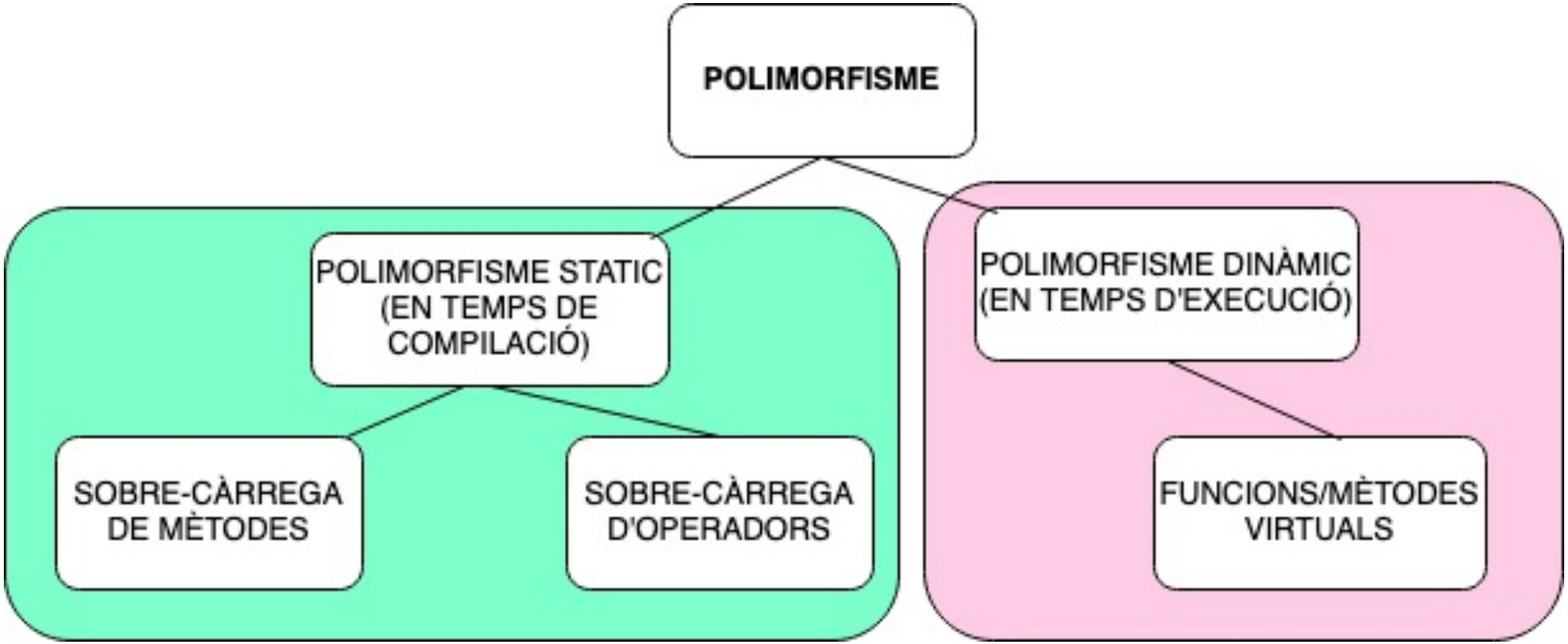
POLIMORFISME (MULTI-FORMA): UN MÈTODE IMPLEMENTAT DE DIVERSES FORMES

- CAS1: Un mètode definit en una classe, s'implementa de diferentes maneres dins la mateixa classe (*overloading* – sobrecàrrega)

```
class Suma{
//Dades
int suma(int a, int b){…} //2 paràmetres int
~~int suma(double u, double v){…}~~
int suma(int a, int b,int c){…} //3 paràmetres int
double suma(double x, double y){…} //2 paràmetres double
Complex suma(Complex c1, Complex c2){…} //2 paràmetres Complex
….
};
Suma S(…);
S.suma(2,5,7); //TEMPS DE COMPILACIÓ
S.suma(12,13); //TEMPS DE COMPILACIÓ
```

- CAS2: Un mètode definit en una super-classe, s'implementa de diferentes maneres en les sub-classes (*overriden* –sobre-implementat)
  - Àrea d'una figura
  - Sou dels empleats
- *HI HA VARIS TIPUS DE POLIMORFISE*

VARIANTS DE POLIMORFISME: STATIC i DINÀMIC



CONCEPTE DE POLIMORFISE ESTÀ RELACIONAT AMB *BINDING* (*STATIC BINDING, DYNAMIC BINDING*)

POLIMORFISME
PROBLEMA: COM SABER QUIN MÈTODE CRIDAR?

- CAS STATIC: SENZILL.
  - ✓ ES RESOL EN TEMPS DE COMPILACIÓ (*COMPILE TIME*)
  - ✓ Suma S(…);
  - ✓ S.suma(3,4);
  - ✓ S.suma(1.3,8.9);
  - ✓ S.suma(1,2,3);
  - ✓ Autobus A(…);
  - ✓ A.setMaxVel(90);
  - ✓ Camio C(…);
  - ✓ C.setMaxVel(60);
  - ✓ cout << A; //la clase Autobus HA de tenir sobre-carregat l'operador del ostream <<
  - ✓ cout << C; //la clase Camio HA de tenir sobre-carregat l'operador del ostream <<

- *CAS DINÀMIC:*
  - ✓ *ES RESOL EN TEMPS D'EXECUCIÓ (RUNTIME)*

Mètodes virtuals en una jerarquia d'herència

- Una funció (mètodes) *virtual* és una funció membre que es declara a la classe base mitjançant la paraula clau *virtual* i que la classe derivada la redefineix (re-implementa).
- Exemple: Jerarquia de Figura2D (simplificada)
- Vegeu programa

Mètodes virtuals purs (*Pure Virtual Functions*)

- Tenim un mètode virtual en una classe base però no la podem implementar (prendrà diferentes formes en les subclasses –polimorfisme).
- *P.ex.* No podem calcular l'àrea d'una Figura2D ja que depen de com sigui la figura (triangle, rectangle, trapezi… apliquen fórmules diferents per les seves àrees).
- En aquest cas, s'aplica la sintaxi del mètode virtual pur

  virtual double area() = 0; //NO S'IMPLEMENTA EN LA CLASSE BASE

- El compilador enten que aquest mètode no té implementació en la classe base (la tindrà –l'ha de tenir-- en les classes derivades)
- Vegeu el programa d'exemple.