Université de Biskra
Département d'Informatique

Relations entre classes (1)
Composition & Agrégation
Vidéo 18

L2-POO

Pr. Laid Kahloul

2019-2020

Plan de l’exposé

- Relation entre classes
- Composition & Agrégation: Exemple
- Composition: Définition, Exemple, Programmation en C++
- Agrégation
- Programmation en C++

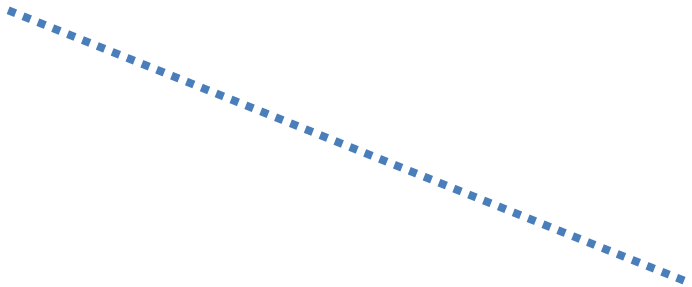
Relations entre classes

- Un programme orienté objet est souvent composés d'un ensemble de classes.
- Ces classes sont liées par des relations.
- On distingue entre trois types de relation: composition, agrégation et héritage

Composition & Agrégation

- Exemples: Un rectangle est composé de lignes

- Une ligne est composé de points



Composition (1)

- La composition est une relation entre une classe dite **Composite** d'une part et une ou plusieurs autres classes dites: **Composantes**.

- L'idée dans la composition est que les classes **Composantes vont disparaitre** si la classe composite disparait.

- L'existence des composantes **est liée étroitement** à la classe composante.

Composition (2)
Exemple: Banque

- Exemple: Une **banque** est vu comme un ensemble de **comptes bancaire**. Si la banque est détruite (faillite) alors les comptes vont disparaitre automatiquement.

Composition (3)
Exemple: Banque

- La classe Banque est la classe **composite**

- La classe Compte est une classe **composante** dans la classe Banque

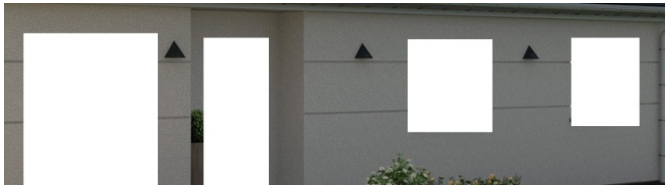
Composition (4)
Exemple: Maison

- Une **maison** est composée de **murs**, **portes**, fenêtres, … Si la maison est détruite, les murs, les fenêtres, etc vont aussi disparaitre

Composition (5)
Exemple: Maison

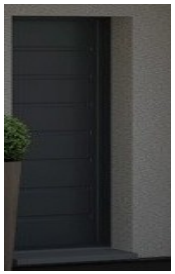
- Maison: classe **composite**

- **Mur: classe composante 1**



- **Fenêtre: classe composante 2**

- **Porte: classe composante 3**
- **…**



Programmation de Composition en C++ (1)

- La classe **composite** va avoir des **attributs** dont les types sont des classes **composantes**.

- Si A est **composé** de B, C, D par exemple alors on peut avoir le programme C++ suivant:

```
class A{
    B b; // composante 1
    C c; // composante 2
    D d; // composante 3
} ;
```

Programmation de Composition en C++ (2)

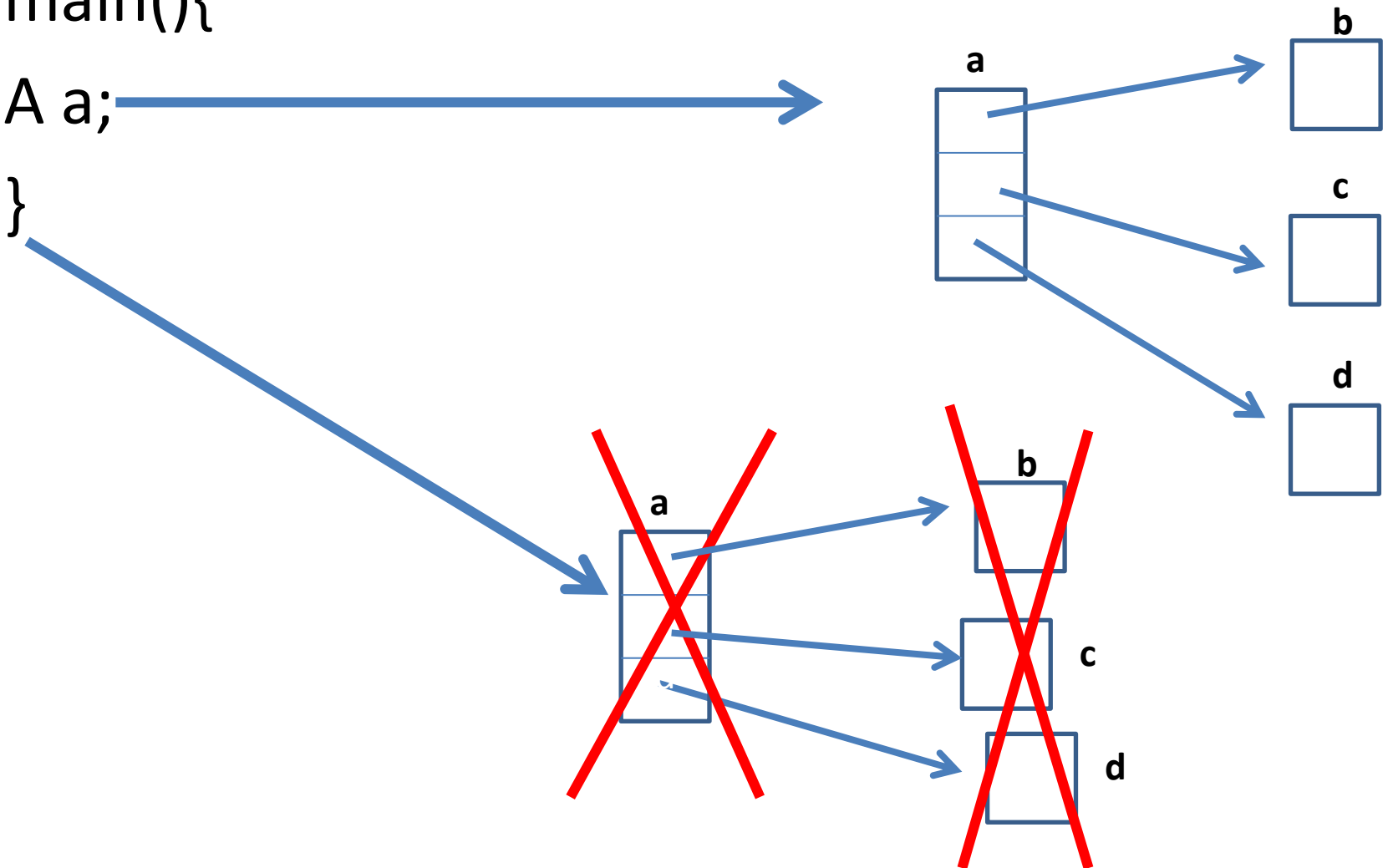
```
class A{
      B b; // composante 1
      C c; // composante 2
      D d; // composante 3
   A(B b, C, c, D d);
} ;

A::A(B b, C, c, D d){
this->b=b; this-> b=b; this-> b=b;
}
```

- La **création** d'une instance a de A **crée des instances** de b, c, d de B, C, et D
- La **destruction** de l'instance a de A **détruit automatiquement les instances** b, c et d
- D'où la composition est une **relation forte**

Programmation de Composition en C++ (3)

```
main(){
A a;
}
```



Programmation de Composition en C++ (4)

- Voir sur Eclipse

```
#include <iostream>
using namespace std;
class B{
        public:
        B();
        ~B();
};
B::B(){
        cout<<"objet de classe B créé"<<endl;
}
B::~B(){
        cout<<"objet de classe B détruit"<<endl;
}
```

Programmation de Composition en C++ (5)

```
class A{
        B b;
        public:
        A();
        ~A();
};
A::A(){
        cout<<"objet de classe A créé"<<endl;
}
A::~A(){
        cout<<"objet de classe A détruit"<<endl;
}
int main() {
        cout << "!!!Hello World!!!" << endl;
                // prints !!!Hello World!!!
        A a;
        return 0;
}
```

Programmation de Composition en C++ (6)

- Refaire le programme principale comme suit et voir les nouveaux résultats:

```
int main() {
      cout << "!!!Hello World!!!" << endl;
 // prints !!!Hello World!!!
      A a;
      a.~A();
      return 0;
}
```

Agrégation: C’est quoi?

- Une composition **faible**.
- Si la composite **disparait**, les composante ne doivent **pas disparaitre**.
- La relation n’est **pas directe par identité** mais plutôt **par référence**.

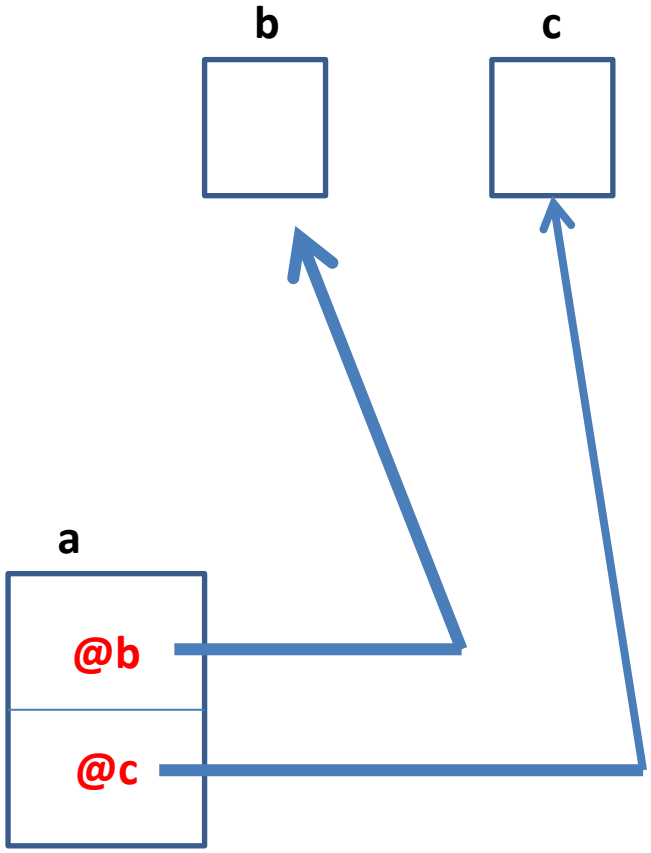
Agrégation: Exemple (1)

- Si par exemple **a** est composé de **b** et de **c**, donc a va avoir des références vers **b** et vers **c** et non pas les identité directes de **b** et de **c**

- **b** et **c** **existent indépendamment** de l'existence de **a**

- Dans un langage comme C++, ceci est implémenté **par des pointeurs** ajouté dans la classe **composite A** vers les deux classes **composantes B et C**

Agrégation: Exemple (2)



Agrégation: Exemple (1) en C++

```
class B{
    };
class C{
    };

class A{
    B*b;
    C*c;
};
```

Agrégation: remarque

- En effet, lorsqu'un objet **a** est créé **ceci ne crée pas** les objets **b** et **c**, mais **simplement une références est créée** vers ces objets qui peuvent exister déjà dans la mémoire.

- Quand l'objet **a est détruit**, **les références vers b et c sont détruites** mais **non les objets b et c**.

- D'où on parle d'une **composition faible** ou **agrégation**

Agrégation: Exemple (2) en C++

```
class B{
public:
        B();
        ~B();
};
B::B(){
        cout<<"objet B créé"<<endl;
}
B::~B(){
        cout<<"objet B détruit"<<endl;
}
class C{
public:
        C();
        ~C();
};
C::C(){
        cout<<"objet C créé"<<endl;
}
C::~C(){
        cout<<"objet C détruit"<<endl;
}
```

Agrégation: Exemple (2) en C++

```
class A{
        B*b;
        C*c;
public:
        A();
        ~A();
};
A::A(){
        cout<<"objet A créé"<<endl;
}
A::~A(){
        cout<<"objet A détruit"<<endl;
}
int main() {
        A a;
        cout << "!!!Hello World!!!" << endl;
         // prints !!!Hello World!!!
        return 0;
}
```

Agrégation: Exemple (2) en C++

Question:

compare est explique l'exécution de ce programme avec le programme de la composition déjà présenté.