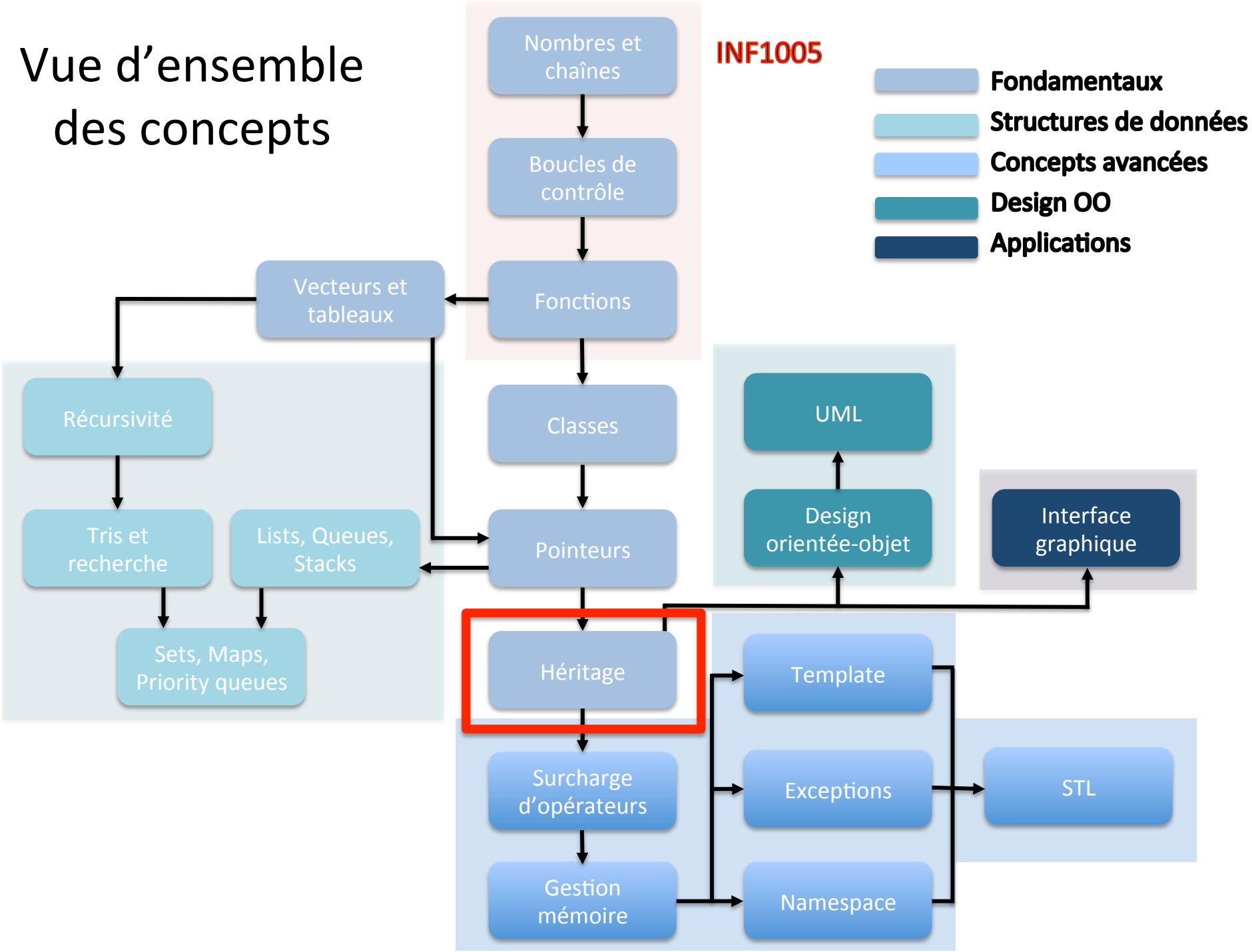
Programmation orientée objet

Héritage multiple

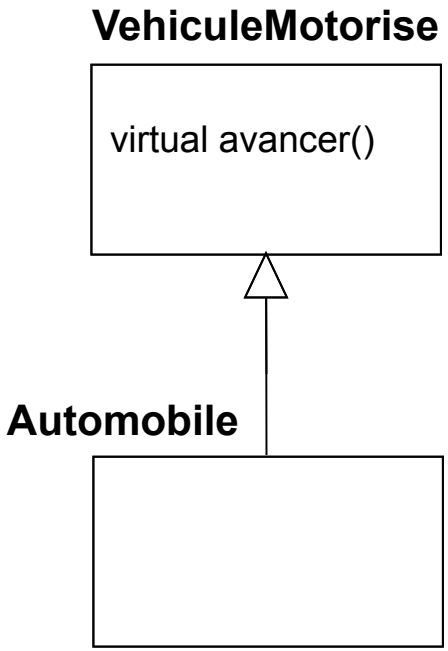
Michel Gagnon - École Polytechnique de Montréal

Vue d'ensemble des concepts



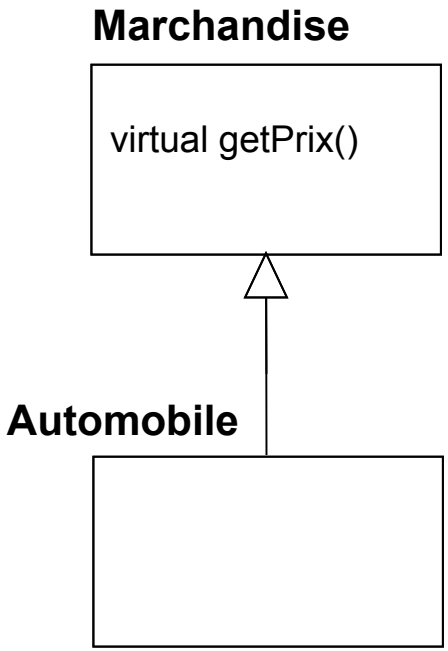
Motivation

- Reprenons notre exemple de véhicule motorisé
- Nous savons qu'une automobile est un véhicule motorisé:



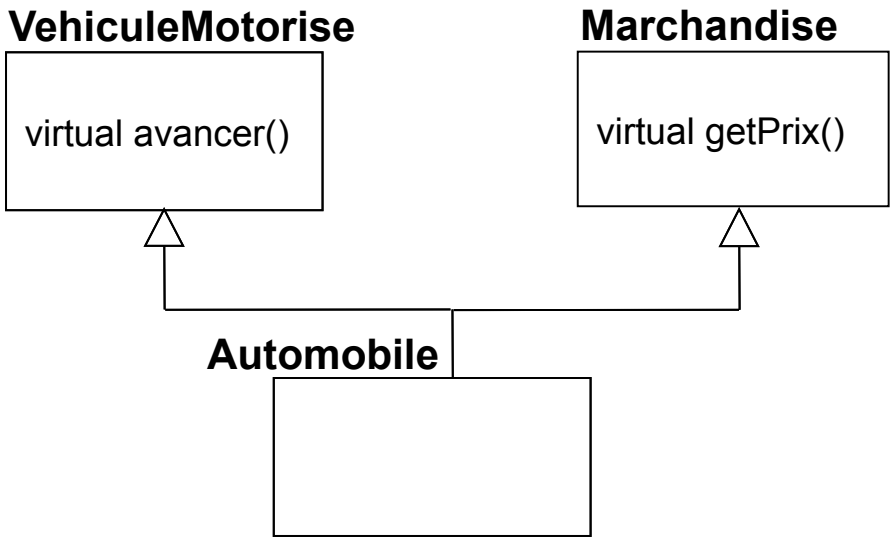
Motivation

- Mais une automobile est aussi une marchandise vendue dans un magasin



Motivation

- Cela signifie donc que la classe Automobile dérive de deux classes
- Certaines méthodes seront héritées d'une classe, d'autres seront héritées de l'autre classe



Héritage multiple en C++

- En C++, on peut faire dériver une classe de plus d'une classe:

```
class Automobile : public VehiculeMotorise,
                   public Marchandise
{
    ...
};
```

Héritage multiple et polymorphisme

- Le polymorphisme fonctionne de la même manière avec l'héritage multiple
- La seule différence avec l'héritage simple est qu'un même objet peut être pointé (ou référé, s'il s'agit d'une référence) par deux pointeurs (ou références) de deux classes de base différentes

Héritage multiple et polymorphisme (exemple)

```
int main()
{
  vector< VehiculeMotorise* > v;
  vector< Marchandise* > m;
  ...
  Automobile* a = new Automobile();
  ...
  v.push_back(a);
  m.push_back(a);
  ...
}
```

Le même objet peut être placé dans deux vecteurs de types différents.

Héritage multiple et polymorphisme (exemple)

```
bool estCher(const Marchandise& m)
{
   return m.getPrix() > 10000;
}
bool estUsage(const VehiculeMotorise& v)
{
   return v.getKilometrage() > 5000;
}
int main()
{
  Automobile a;
  if (estCher(a)) {
      ...
  }
  if (estUsage(a)) {
      ...
  }
}
```

Le même objet peut être passé à des fonctions dont les paramètres sont de types différents.

Ambiguïté de nom

- Et si les deux classes de base dont on hérite ont exactement la même méthode?
- Supposons par exemple que les classes VehiculeMotorise et Marchandise ont toutes les deux une méthode getId(), qui retourne un identificateur numérique
- Si un objet est de la classe Automobile, on ne pourra pas appeler la méthode getId() sur cet objet, puisque le compilateur ne saura pas quelle méthode appeler

Ambiguïté de nom (suite)

- Une manière de régler le problème consiste à indiquer explicitement la classe de la méthode appelée
- Supposons par exemple que **`a`** est un objet de la classe Automobile et que l'on veuille son identificateur de marchandise, on utilisera alors l'expresssion suivante:

```
a.Marchandise::getId()
```

Ambiguïté de nom (suite)

- Une meilleure solution consiste à cacher l'ambiguïté et redéfinir deux méthodes pour la classe Automobile:

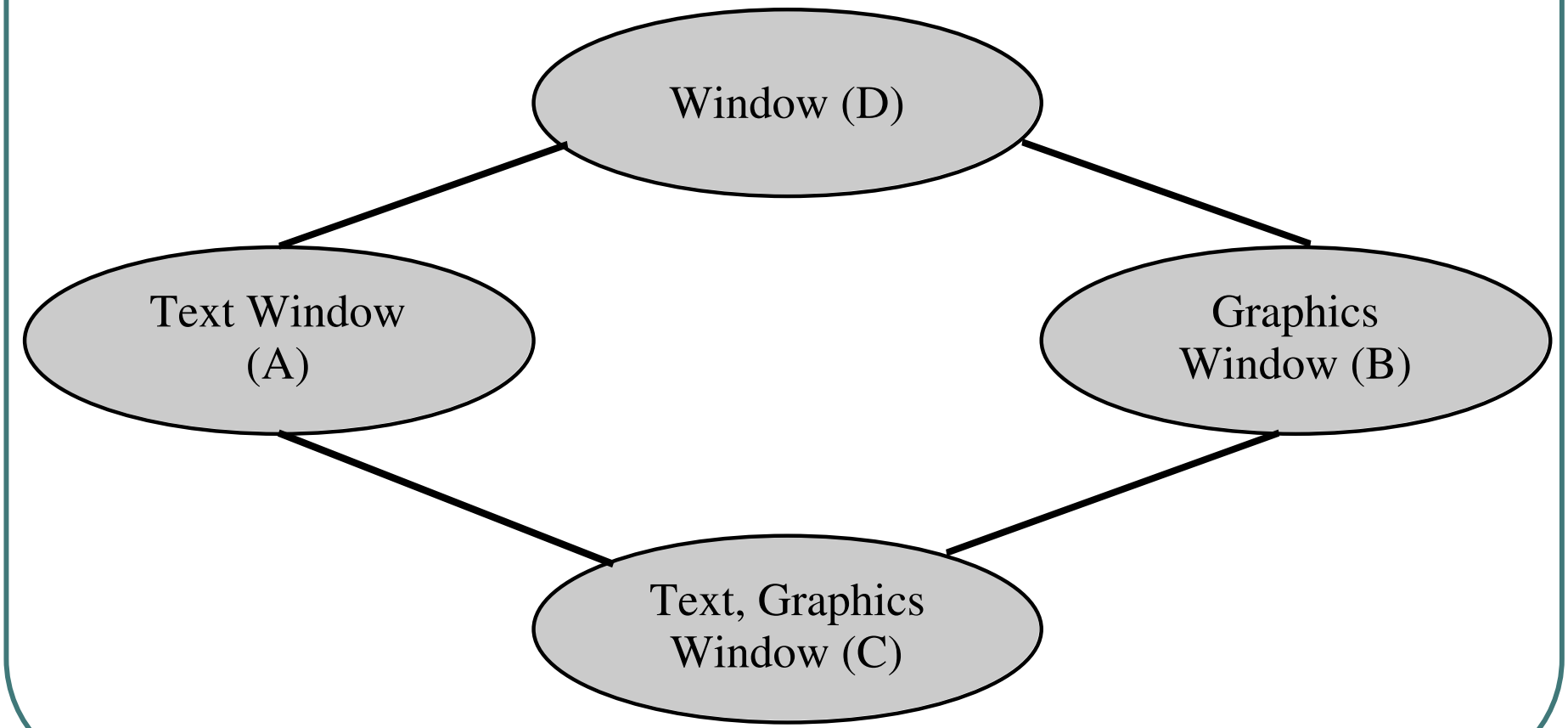
```
Automobile::getIdMarchandise() const
{
  return Marchandise::getId();
}

Automobile::getIdVehicule() const
{
  return VehiculeMotorise::getId();
}
```

Attention avec l'héritage multiple

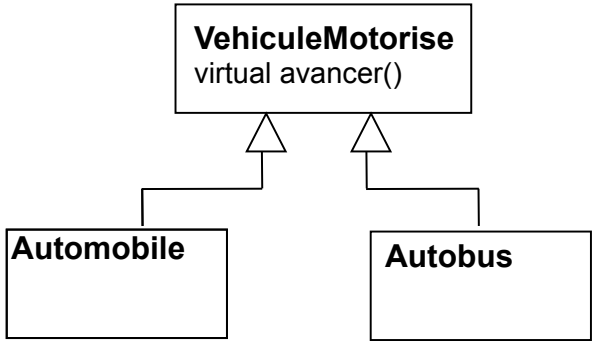
- Bien d'autres problèmes sont liés à l'héritage multiple
- L'héritage multiple peut compliquer substantiellement la compréhension d'un code
- Il faut donc l'éviter le plus possible

Exemple: Problème du diamant



Distinction entre type et interface

- En général une classe appartiendra à une seule hiérarchie de base:



Distinction entre type et interface

- À cette hiérarchie de base peuvent s'ajouter des interfaces (qui devraient toujours être virtuelles pures et ne devraient pas contenir d'implémentations):