Objectifs
Concepts
Etude de cas

Programmation Orientée Objet : Polymorphisme, 1re partie

Jean-Cédric Chappelier

Faculté I&C

©EPFL 2026
Jean-Cédric Chappelier
& Jamila Sam
EPFL

Organisation du travail (semestre)

		MOOC	déc.	cours 1 h Jeudi 8-9	exercices 2 h Jeudi 9-11
1	19.02.26		0	Intro + compil. séparée	
2	26.02.26	1. Intro POO	0	Intro POO	
3	05.03.26	2. Constructeurs/Dest	0	Constructeurs	
4	12.03.26	3. Surcharge des opé	0	Surcharge	
5	19.03.26	4. Héritage	0	Héritage	
6	26.03.26	5. Polymorphisme	0	Polymorphisme 1	
7	02.04.26		1	Polymorphisme 2 / Collections hétérogènes	
-	09.04.26		-	vacances Pâques	
8	16.04.26	6. Héritage multiple	1	Héritage multiple	
9	23.04.26		-	-	Midtem
10	30.04.26	(7. Etude de cas)	-	Templates	
11	07.05.26		-	Structure de données abstraites ; Bibliothèques	
12	14.05.26		-	(Ascension)	
13	21.05.26	(7. Etude de cas)	-	Bibliothèques (fin) + Révisions	
14	28.05.26		-	-	Examen

©EPFL 2026
Jean-Cédric Chappelier
& Jamila Sam
EPFL

Objectifs de la leçon d’aujourd’hui

- Concepts fondamentaux
- Étude de cas

©EPFL 2026
Jean-Cédric Chappelier
& Jamila Sam
EPFL

Concepts fondamentaux

- notion de polymorphisme
  1. notion **FONDAMENTALE !**
     - ☞ Attention aux « tests de type » : JAMAIS ! ! !
  2. en C++ : 2 ingrédients : ...
- méthodes virtuelles :
  `virtual` (transmis par héritage ; mais je vous conseille de le remettre à chaque fois pour mémoire)
  `override` (optionnel, mais conseillé)
- méthodes virtuelles *pures* et classes *abstraites*
  1. `virtual` *plus* `= 0`
  2. on **ne** peut **pas** créer d'instance de classe abstraite !
- (semaine prochaine : collections héterogènes)

NON

```
void bruit(Animal a)
{
  if (a.type() == chat)
    miauler()
  else (a.type() == chien)
    aboyer()
  :
}
```

OUI

```
a.bruit()
```

Polymorphismes

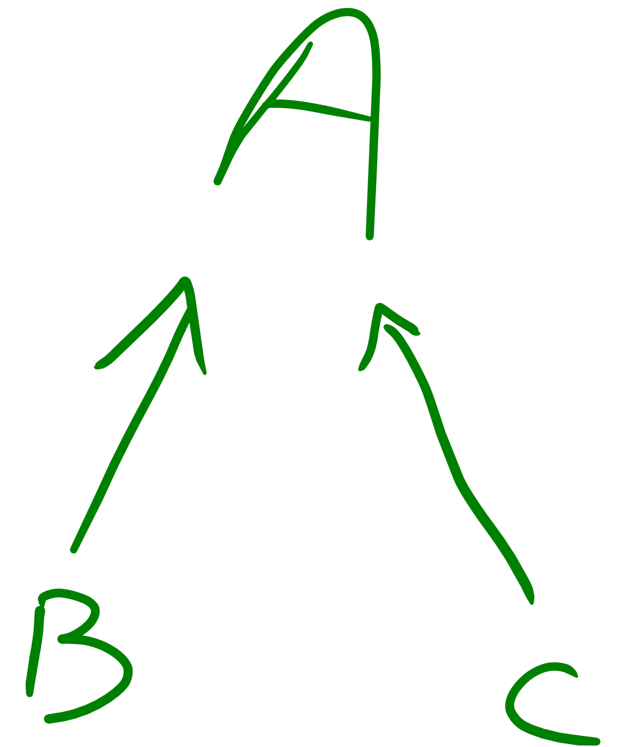
En programmation, on distingue deux types de polymorphismes :

- le **polymorphisme des traitements** (ou ad hoc)
  - ☞ mécanisme de **surcharge** des fonctions/méthodes : le même identificateur est utilisé pour désigner des séquences d'instructions différentes
- le **polymorphisme des données** (ou universel)
  - le polymorphisme **d'inclusion** :
    le même code peut être appliqué à des données de types différents liés entre eux par une relation de sous-typage
    ☞ hiérarchies de classes
  - le polymorphisme **paramétrique** :
    le même code peut être appliqué à n'importe quel type (généricité)
    ☞ Cours sur les *templates* dans quelques semaines

à venir

©EPFL 2026
Jean-Cédric Chappelier
& Jamila Sam
EPFL

```
A& a
  ⋮
cout << a ;
```



Exemple illustratif : l’affichage polymorphique

```
class Affichable {
public:
  virtual void affiche(ostream& flot) const = 0;
};

// -------------------------------------------------------
ostream& operator<<(ostream& flot, const Affichable& objet)
{
  objet.affiche(flot);
  return flot;
}

// =======================================================
class Machin: public Affichable
{
public:
// ...
  virtual void affiche(ostream& flot) const override {
    // ...
  }
// ...
};
```

1 seul

©EPFL 2026
Jean-Cédric Chappelier
& Jamila Sam
EPFL

Exercice 1 – Un séjour au soleil [sur 99 points]

On cherche ici à écrire un programme pour gérer le système de réservation[1] d'un hôtel. Une réservation portera sur trois choses : le type de chambre, la formule de repas (nombre par jour et régime alimentaire[2]) et les extras choisis (sauna, tennis, ...).

Nous allons commencer par nous intéresser aux extras, puis aux types de chambres, aux repas et l'on finira par les réservations proprement dites.

Question 1 – Prestations [sur 9 points]

est - un

Mais avant tout cela, toutes ces prestations[3] (chambre, repas, extra, réservation) ayant un prix, on souhaite les modéliser de façon commune : définissez ici une classe `Prestation` qui comprend une méthode `prix()` dont le comportement est propre à chaque prestation et n'a pas de définition générale. Cette méthode retournera un `double` et ne modifiera pas la prestation concernée.

= virtual

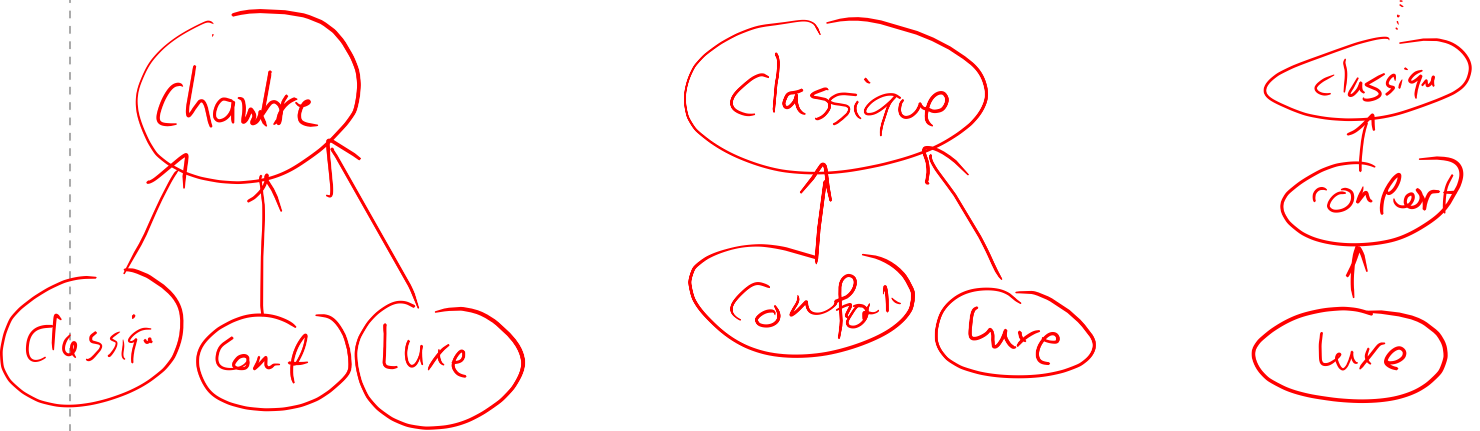
= pure

Question 2 – Prestations « extra » [sur 10 points]

L'hôtel souhaite offrir à ses clients des prestations « extra » en plus des chambres et des repas : piscine, sauna, tennis, golf, etc. Toutes ces prestations supplémentaires (« extra ») n'ayant pas de comportement particulier, il suffit de les modéliser par un nom et un prix (fixes et connus lors de leur construction).

En haut de la page suivante, définissez **complètement** la classe `Extra` permettant de représenter de telles « prestations supplémentaires ».

1. "*booking*", in English. 2. "*food diet*", in English. 3. "*service*", in English.



Question 3 – Chambres [sur 36 points]

Au niveau des chambres, l'hôtel souhaite offrir trois types de prestations : des chambres classiques, des chambres « confort » et des chambres de luxe.

Toutes ces chambres sont caractérisées par un nombre de lits et un nombre de salles de bains. Elles ont de plus accès à une liste d'« extra » (cf Question 2) qui leur est propre. Les chambres « confort » sont comme des chambres classiques, mais ont en plus un prix forfaitaire pour les prestations « extra » (son utilisation est expliquée ci-dessous). Les chambres « de luxe » sont exactement comme les chambres classiques, sauf que le calcul de leur prix est différent.

Lors de la création d'une chambre (quelle qu'elle soit), on précisera son nombre de lits (2 par défaut) et son nombre de salles de bains (1 par défaut). On ne fournira par contre aucun « extra » lors de la création d'une chambre. Les « extra » seront ajoutés (après construction) au moyen d'une méthode `ajoute_extra()`.

Pour le calcul des prix :

- le prix des chambres classiques vaut 60 fois le nombre de lits plus 40 fois le nombre de salles de bain, plus la somme des prix de toutes les prestations « extra » de la chambre ;
  les valeurs 60 et 40 ci-dessus devront être définies comme attributs de classe ;
- le prix des chambres « confort » est exactement le même que celui des chambres classiques, sauf que l'on rajoute le prix forfaitaire (dont on parlait ci-dessus) et que l'on soustrait le prix du premier « extra » (s'il existe, sinon on ne fait qu'ajouter le prix forfaitaire) ;
- le prix des chambres « de luxe » est simplement de 1000 francs.

Au dos de cette page et sur la suivante, définissez la ou les classe(s) permettant de représenter les chambres telles que décrit ci-dessus. Limitez-vous à ce que vous considérez être le *strict* nécessaire correspondant à la description donnée ci-dessus.