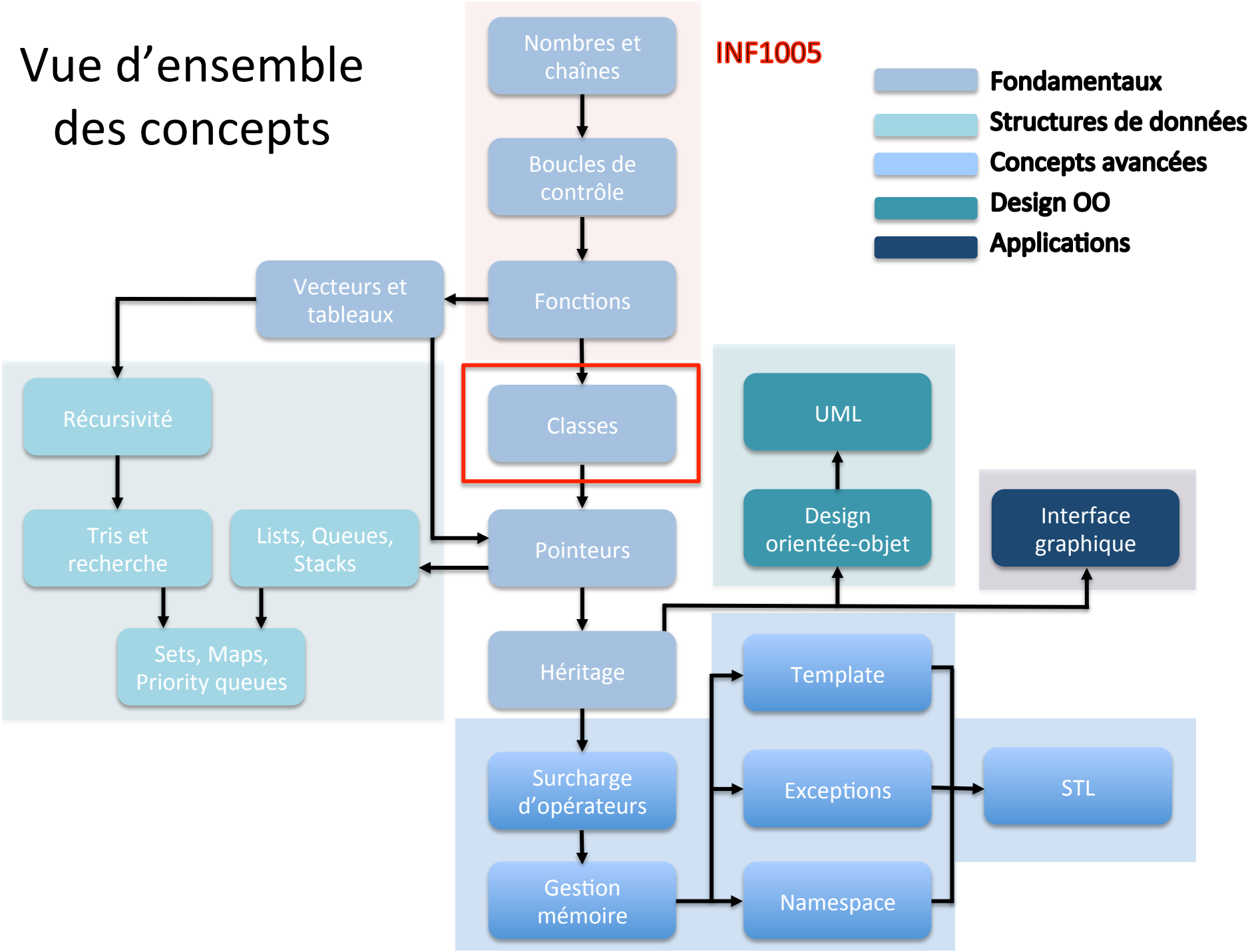
Programmation orientée objet

Méthodes constantes

Michel Gagnon

Vue d'ensemble des concepts



Motivation

- Nous avons vu que certaines méthodes ne servent qu'à accéder à des attributs d'un objet
- Pour éviter toute confusion, on aimerait que ces méthodes ne puissent effectuer aucune modification sur les attributs de l'objet

Implémentation d'une fonction d'accès

- Pour assurer que la méthode ne modifie aucun des attributs de l'objet, on ajoute le mot clé **`const`** après l'en-tête de la fonction:
  - dans la définition de la classe
  - dans l'implémentation de la fonction

Exemple de fonction d'accès

```
double Point::getX() const
{
    return x_;
}
```

Le compilateur retournera un message d'erreur si la fonction contient une instruction qui tente de modifier la valeur d'un attribut de l'objet **ou si la fonction utilise une méthode qui n'a pas été déclarée constante**.

Exemple de code erroné avec const

```
double Point::getX() const
{
    ++x_;
    return x_;
}
```

On n'a pas le droit de modifier la valeur d'un attribut de l'objet.

Autre exemple de code erroné avec const

```
class Commande
{
public:
   ...
   void afficher() const;
    int getAtt1();
    int getAtt2();
private:
    ...
    int att1_;
    int att2_;
};
```

```
int Commande::getAtt1()
{
   return att1_;
}

int Commande::getAtt2()
{
   return att2_;
}

void Commande::afficher() const
{
   cout << getAtt1() << endl;
   cout << getAtt2() << endl;
}
```

Erreur!
Pourquoi?

Autre exemple de code erroné avec const

```
class Commande
{
public:
   ...
   void afficher() const;
    int getAtt1() const;
    int getAtt2() const;
private:
    ...
    int att1_;
    int att2_;
};
```

```
int Commande::getAtt1() const
{
   return att1_;
}

int Commande::getAtt2() const
{
   return att2_;
}

void Commande::afficher() const
{
   cout << getAtt1() << endl;
   cout << getAtt2() << endl;
}
```

Encore un exemple de code erroné avec const

```
class Produit
{
   ...
   void print();
    ...
};
```

```
class Commande
{
public:
   ...
   void afficher() const;
private:
    ...
    Produit article_;
};
```

```
void Commande::afficher() const
{
    ...
    article_.print();
    ...
}
```

Erreur!!
Pourquoi?

Encore un exemple de code erroné av

```
class Produit
{
   ...
   void print() ;
    ...
}
```

```
 Commande
{
public:
   ...
   void afficher() const;
private:
    ...
    Produit article_;
}
```

Cette méthode n'est pas déclarée constante.

Cette méthode n'a pas le droit de modifier un attribut.

```
void Commande::afficher() const
{
    ...
    article_.print();
    ...
}
```

Puisque la méthode `print` n'est pas déclarée constante, elle pourrait donc modifier l'attribut `article` (même si elle ne le fait pas en pratique). Le compilateur refusera donc de compiler ce code.

Encore un exemple de code erroné avec const

```
class Produit
{
   ...
   void print() const;
    ...
}
```

```
class Commande
{
public:
   ...
   void afficher() const;
private:
    ...
    Produit article_;
}
```

```
void Commande::afficher() const
{
    ...
    article_.print();
    ...
}
```