

Programmation orientée objet

Méthodes constantes

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

```
string Employee::getName() const
{
    return name_;
}
```

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

```
string Employee::getName() const
{
    name_ = "pierre";
    return name_;
}
```

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

Autre exemple de code erroné avec const

```
class Employee
{
public:
    Employee(string name =
        "unknown", double salary
        = 0);
    double getSalary();
    string getName();
    ...
    void print() const;

private:
    string name_;
    double salary_;
};
```

```
string Employee::getName()
{
    return name_;
}

double Employee::getSalary()
{
    return salary_;
}

void Employee::print() const
{
    cout << getName() << endl;
    cout << getSalary() << endl;
```

Erreur!
Pourquoi?

Autre exemple de code erroné avec const

```
class Employee
{
public:
    Employee(string name =
        "unknown", double salary
        = 0);
    double getSalary() const;
    string getName() const;
    ...
    void print() const;

private:
    string name_;
    double salary_;
};
```

```
string Employee::getName() const
{
    return name_;
}

double Employee::getSalary() const
{
    return salary_;
}

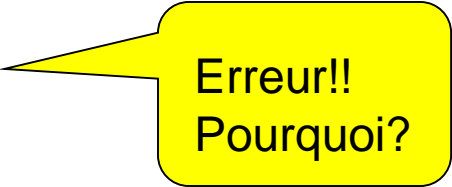
void Employee::print() const
{
    cout << getName() << endl;
    cout << getSalary() << endl;
}
```

Encore un exemple de code erroné avec const

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

```
class Company
{
public:
    ...
    void print() const;
private:
    ...
    Employee president_;
};
```

```
void Company::print() const
{
    ...
    president_.print();
    ...
}
```



Erreur!!
Pourquoi?

Encore un exemple de code erroné avec

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

Cette méthode n'est pas déclarée constante.

```
class Company
{
public:
    ...
    void print() const;
private:
    ...
    Employee president_;
};
```

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

```
void Company::print() const
{
    ...
    president_.print();
    ...
}
```

Puisque la méthode `print` n'est pas déclarée constante, elle pourrait donc modifier l'attribut `president` (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 Employee
{
    ...
    void print() const;
    ...
};
```

```
class Company
{
public:
    ...
    void print() const;
private:
    ...
    Employee president_;
};
```

```
void Company::print() const
{
    ...
    president_.print();
    ...
}
```