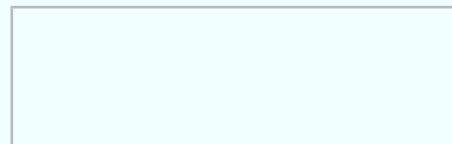


Factorisation : $6x^2+20x = 4x (4x+5)$

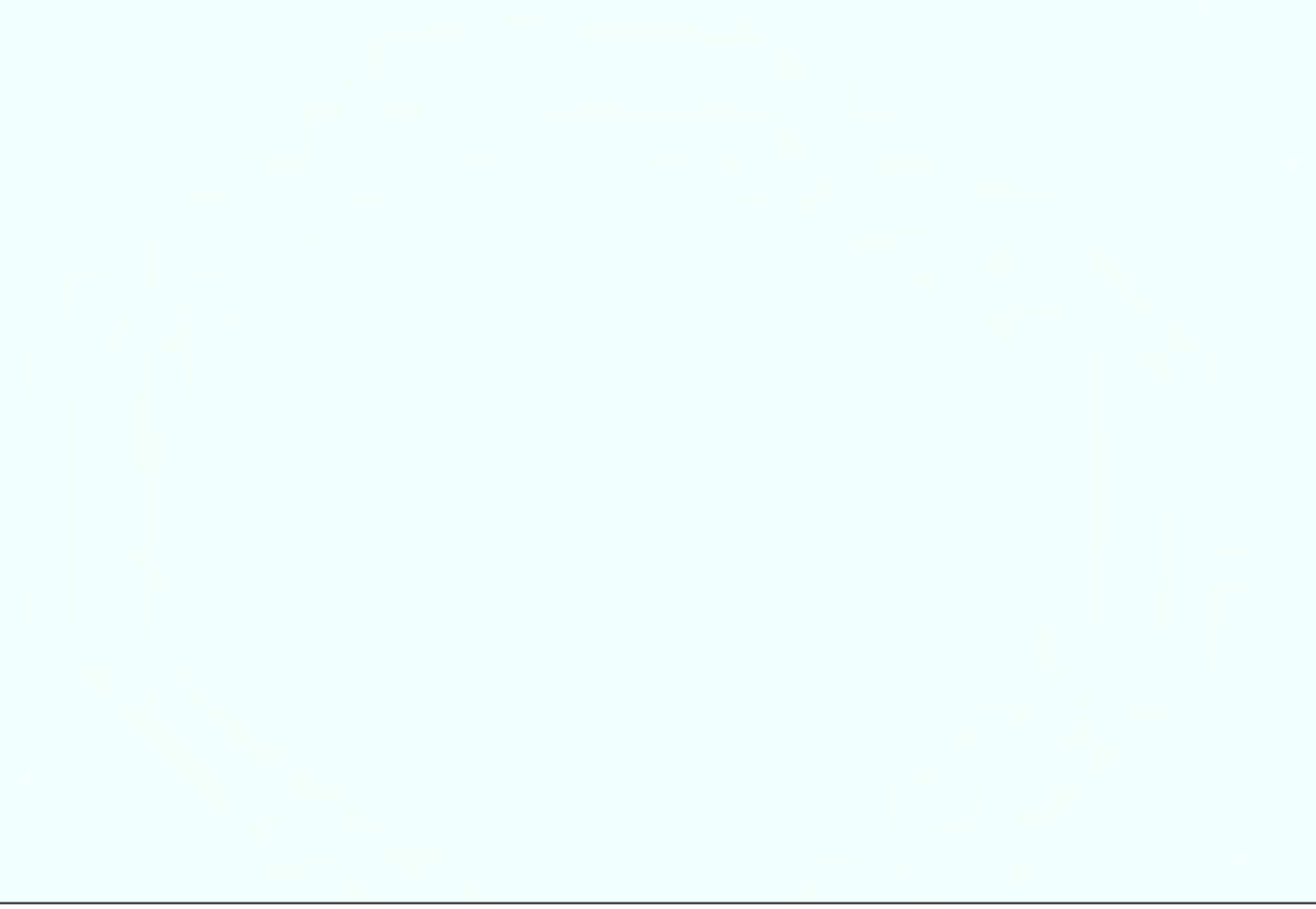









i.e.

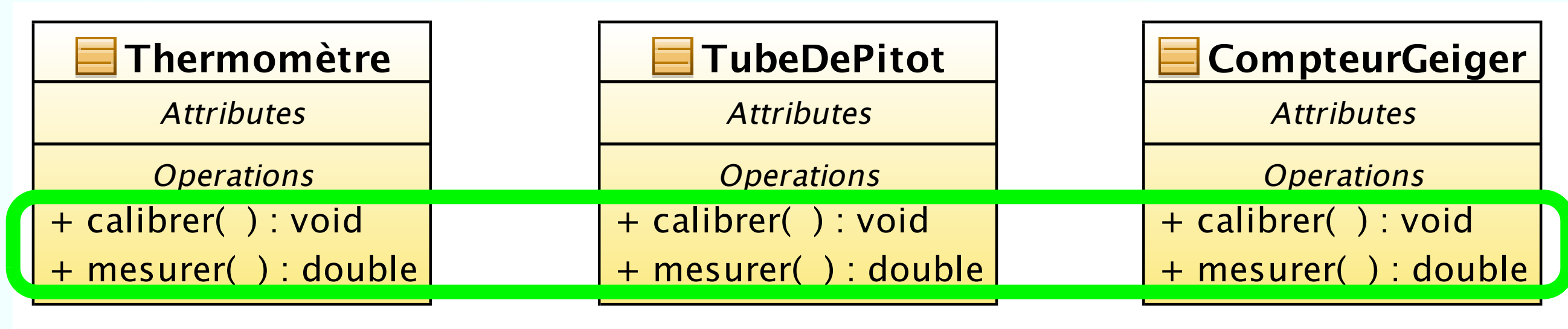
i.e.

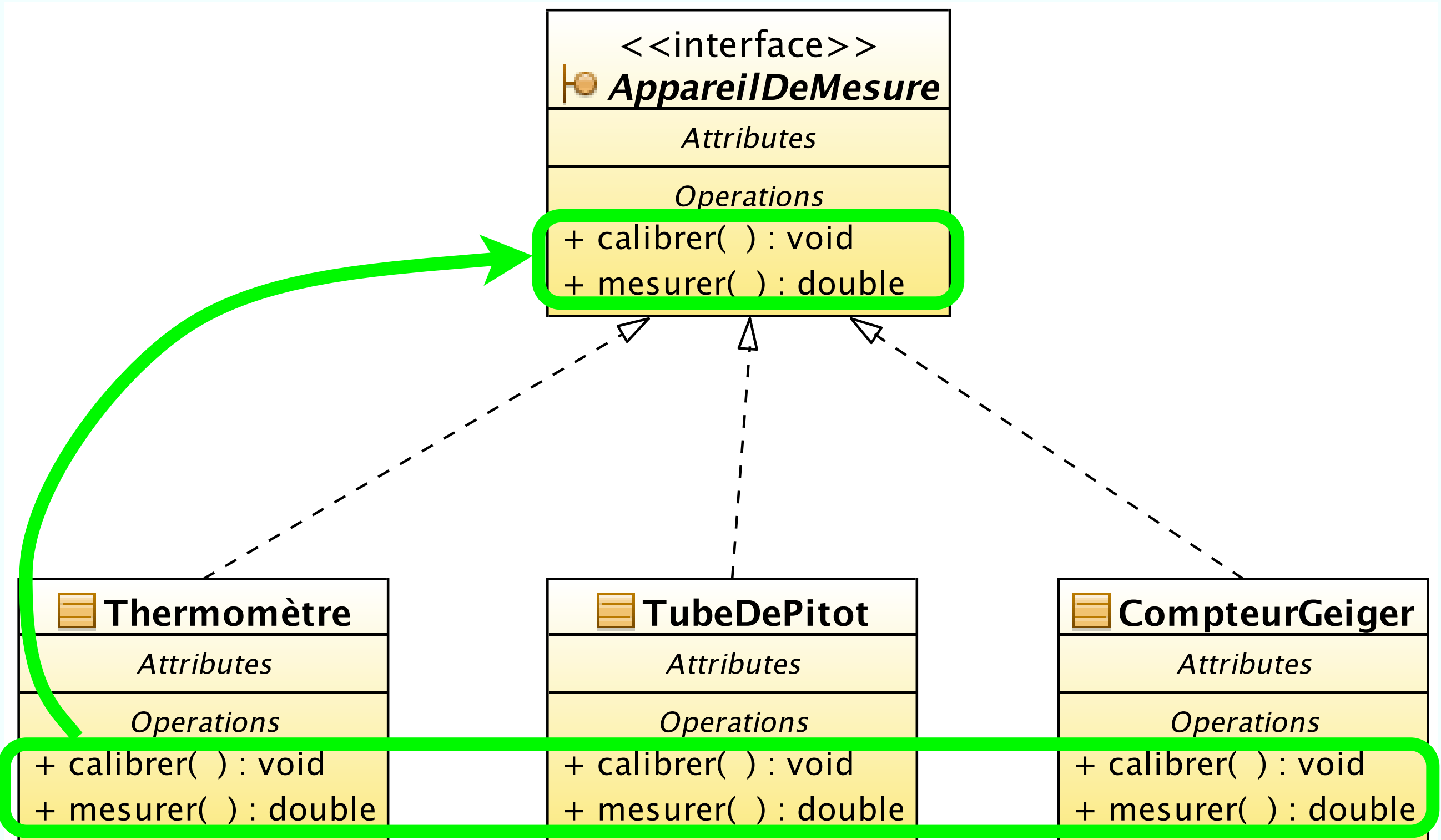


 Thermomètre
<i>Attributes</i>
<i>Operations</i> + calibrer() : void + mesurer() : double

 TubeDePitot
<i>Attributes</i>
<i>Operations</i> + calibrer() : void + mesurer() : double

 CompteurGeiger
<i>Attributes</i>
<i>Operations</i> + calibrer() : void + mesurer() : double





```
class AppareilDeMesure {  
    public:  
        virtual void calibrer()=0; // équiv. C++ du abstract de Java  
        virtual double mesurer()=0;  
};
```

```
class CompteurGeiger : public AppareilDeMesure {  
    public:  
        virtual void calibrer() { /* remettre à zéro */ }  
        virtual double mesurer() { /* compter les particules */ }  
};
```

```
class TubeDePitot : public AppareilDeMesure {  
    public:  
        virtual void calibrer() { /* remettre à zéro */ }  
        virtual double mesurer() { /* soustraire des pressions */ }  
};
```

```
class Thermomètre : public AppareilDeMesure {  
    public:  
        virtual void calibrer() { /* laisser refroidir */ }  
        virtual double mesurer() { /* attendre la stabilisation */ }  
};
```

• `virtual type identificateur(type identificateur, ...) = 0`



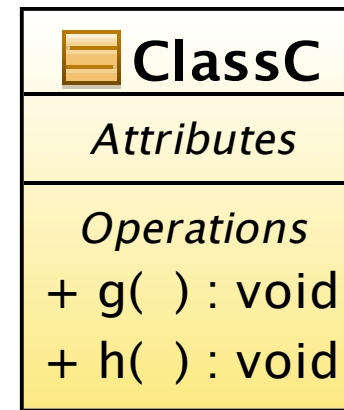
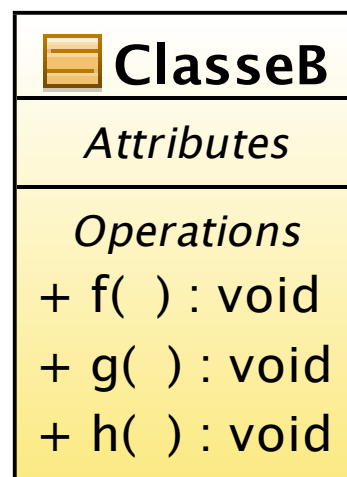
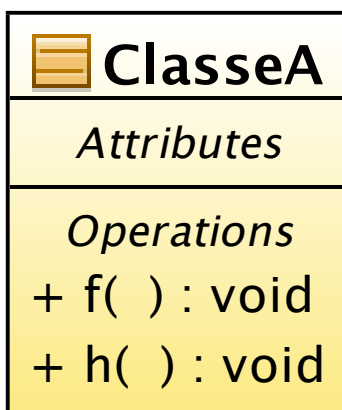
```
void faitFonctionner(AppareilDeMesure &a) {  
    a.mesurer();  
}
```

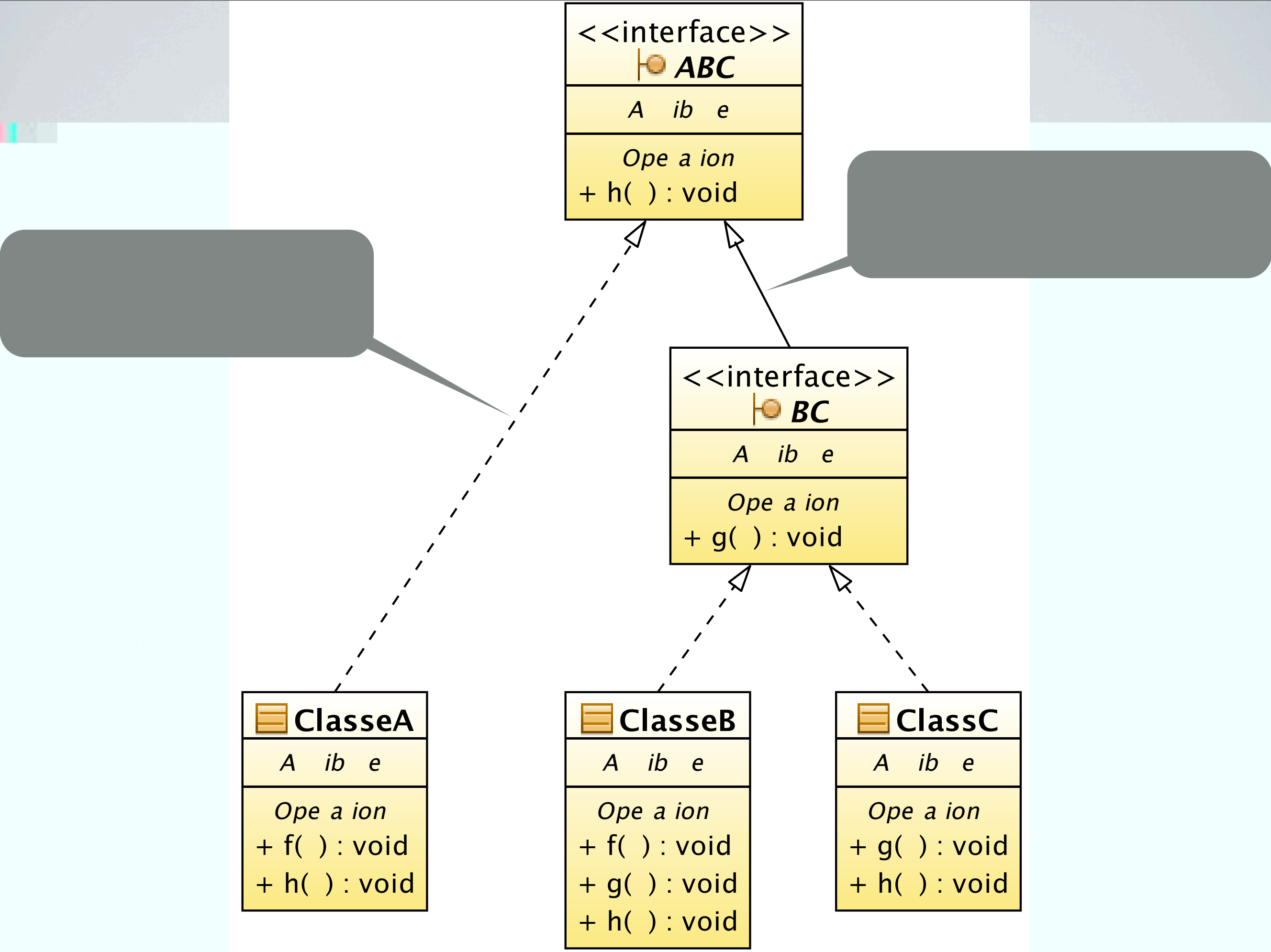
```
void faitFonctionner(AppareilDeMesure *pa) {  
    pa->mesurer();  
}
```

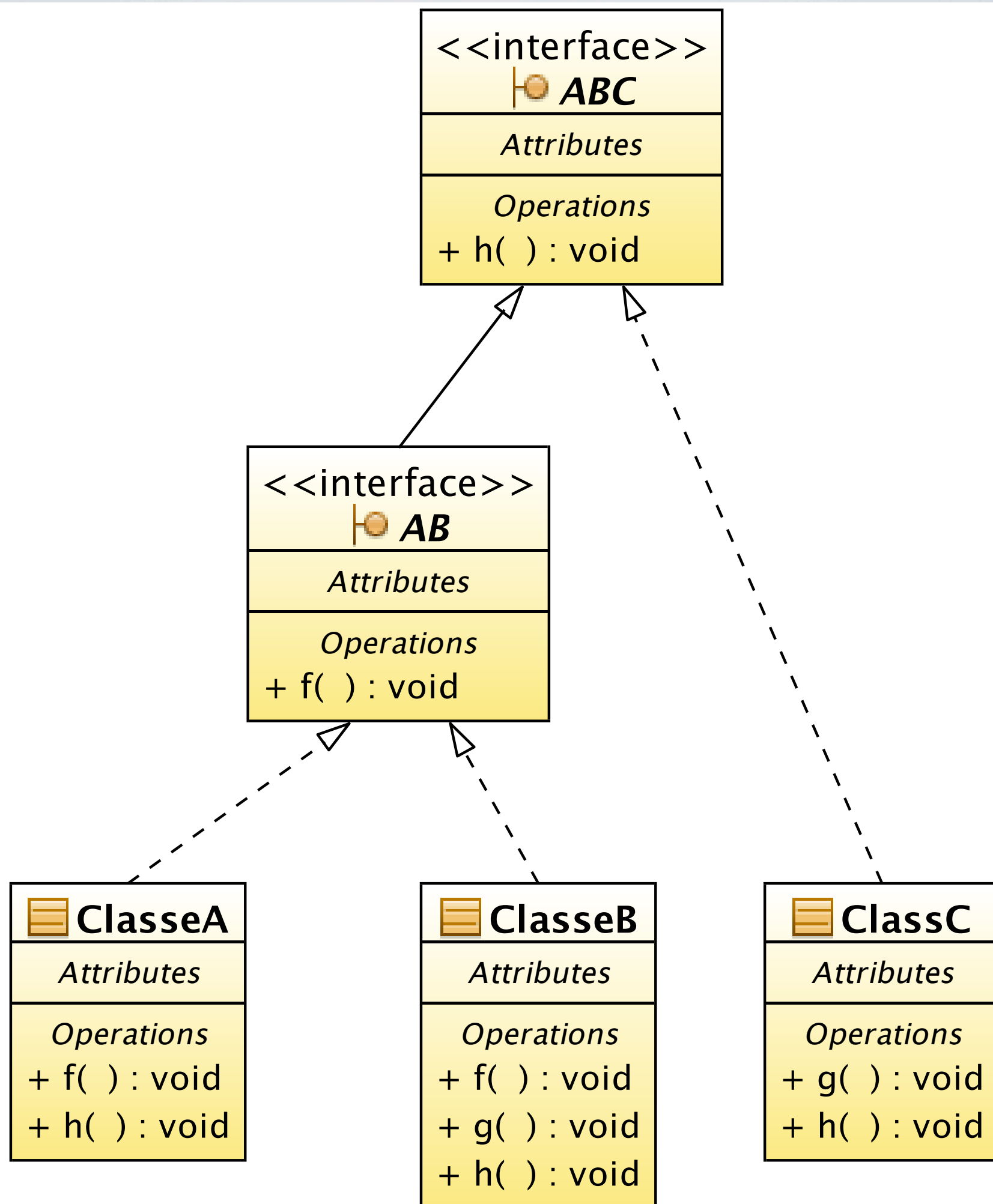
```
int main() {  
    Thermomètre t;  
    faitFonctionner(t);  
    faitFonctionner(&t);  
    SondePitot s;  
    faitFonctionner(s);  
    faitFonctionner(&s);  
    return 0;  
}
```

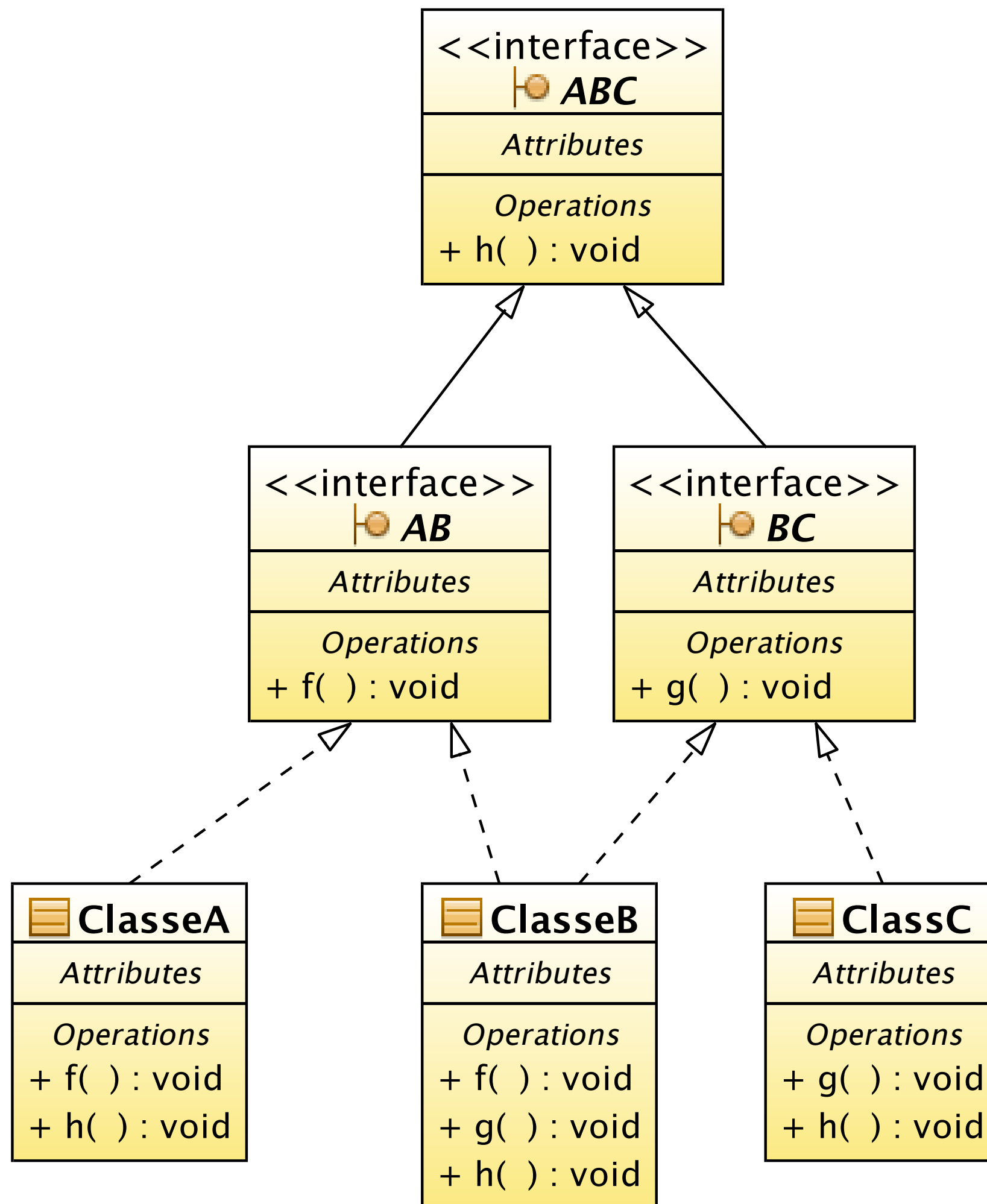
```
[yunes] ./main  
Thermo  
Thermo  
Pitot  
Pitot  
[yunes]
```

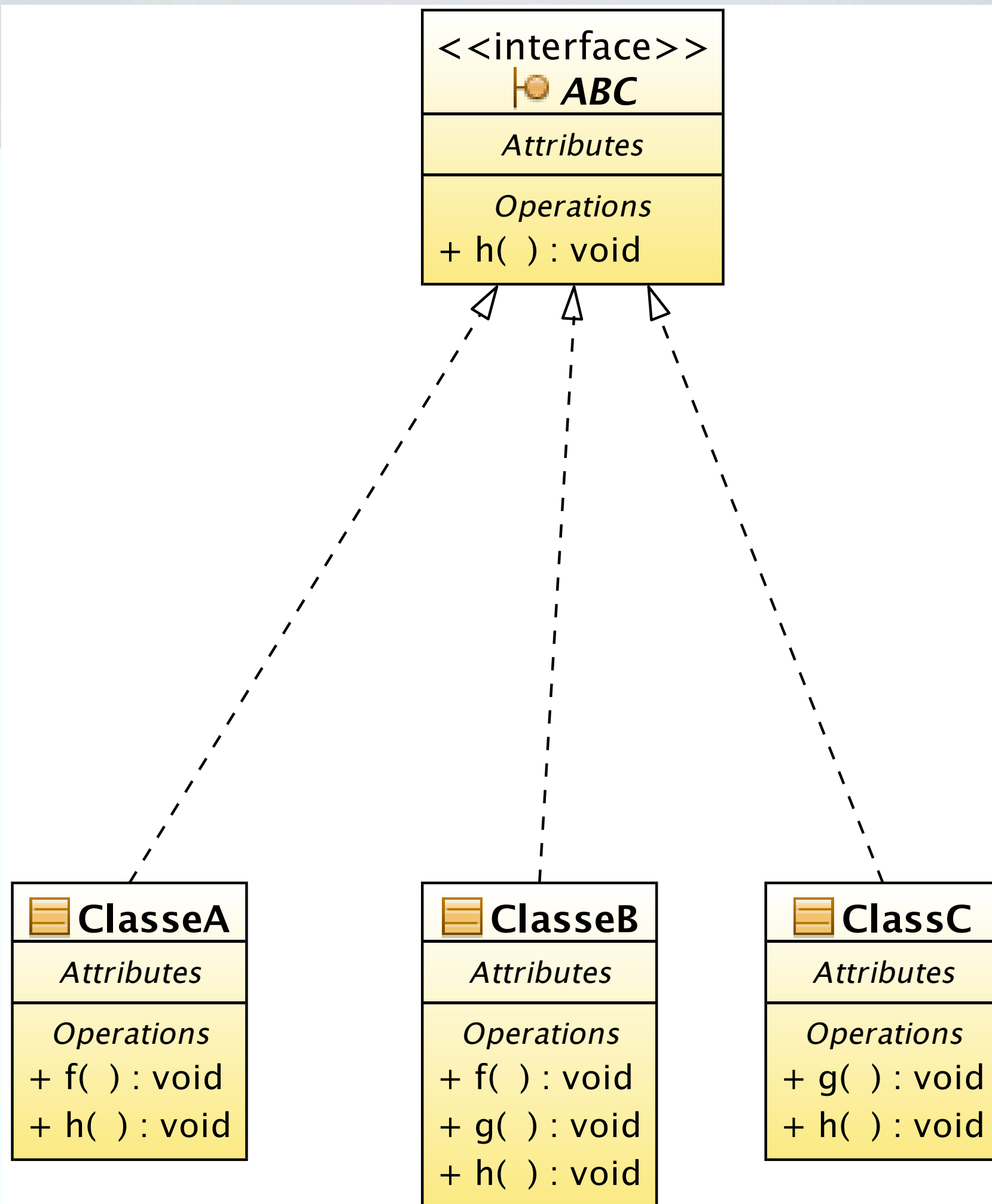






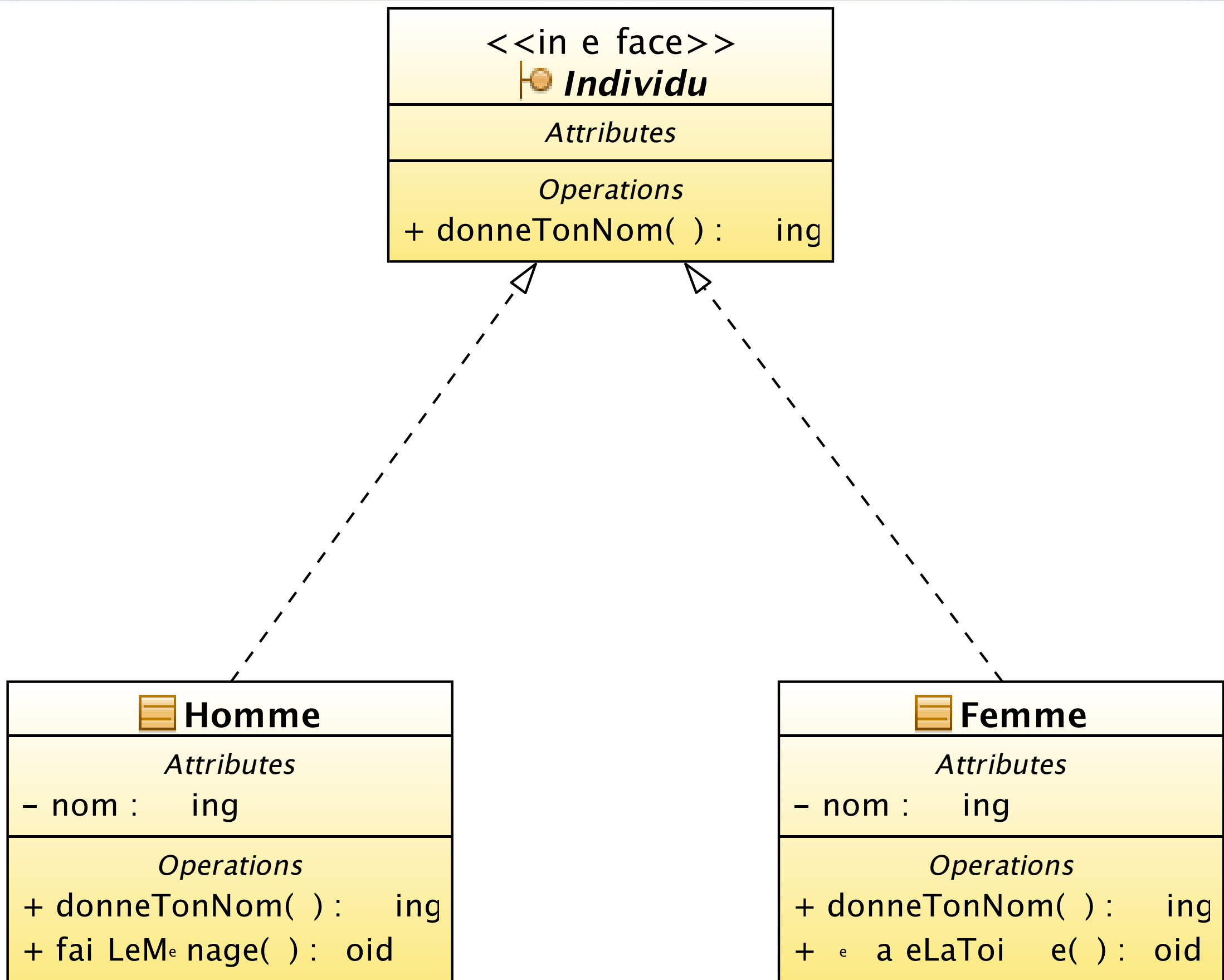












```
class Individu {  
    public:  
        virtual string donneTonNom()=0;  
};
```

```
class _Individu : public Individu {  
    private:  
        string nom;  
    public:  
        virtual string donneTonNom() { return nom; }  
};
```

```
class Homme : public _Individu {  
    public:  
        void faitLeMénage() { ... }  
};
```

```
class Femme : public _Individu {  
    public:  
        void répareLeToit() { ... }  
};
```

i.e. donneTonNom()

```

class Individu {
    public:
        virtual string donneTonNom()=0;
        virtual void ditTonNom()=0;
};

class _Individu : public Individu {
    private:
        string nom;
    public:
        virtual string donneTonNom() { return nom; }
};

class Homme : public _Individu {
    public:
        virtual void ditTonNom() {
            synthétiseurVocal.setMode(GRAVE);
            synthétiseurVocal.synthétise(donneTonNom());
        }
};

class Femme : public _Individu {
    public:
        virtual void ditTonNom() {
            synthétiseurVocal.setMode(AIGU);
            synthétiseurVocal.synthétise(donneTonNom());
        }
};


```

```
class Individu {
    public:
        virtual string donneTonNom()=0;
        virtual void ditTonNom()=0;
};

class _Individu : public Individu {
    private:
        string nom;
    public:
        virtual string donneTonNom() { return nom; }
        virtual void ditTonNom() { synthétiseurVocal.synthétise(donneTonNom()); }
};

class Homme : public _Individu {
    public:
        virtual void ditTonNom() {
            synthétiseurVocal.setMode(GRAVE);
            _Individu::ditTonNom();
        }
};

class Femme : public _Individu {
    public:
        virtual void ditTonNom() {
            synthétiseurVocal.setMode(AIGU);
            _Individu::ditTonNom();
        }
};
```



```
class Individu {  
    public:  
        virtual string donneTonNom()=0;  
        virtual void ditTonNom()=0;  
};
```

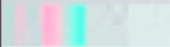
```
class _Individu : public Individu {  
    private:  
        string nom;  
    public:  
        virtual string donneTonNom() { return nom; }  
        void ditTonNom() { synthétiseurVocal.synthétise(donneTonNom()); }  
};
```

```
class Homme : public _Individu {  
    public:  
        virtual void ditTonNom() {  
            synthétiseurVocal.setMode(GRAVE);  
            _Individu::ditTonNom();  
        }  
};
```

```
class Femme : public _Individu {  
    public:  
        virtual void ditTonNom() {  
            synthétiseurVocal.setMode(AIGU);  
            _Individu::ditTonNom();  
        }  
};
```

virtual

_Individu



virtual



doivent être

virtual

```

class Individu {
    public:
        virtual string donneTonNom()=0;
        ~Individu() { cout << "~Individu()" << endl; };
};
class Femme : public Individu {
    public:
        Femme(string nom) : Individu(nom) {};
        ~Femme() { cout << "~Femme(" << donneTonNom() << ")" << endl; }
};

void libere(Individu *pi) {
    delete pi;
}

int main()
{
    Femme f("Georgette");
    Femme *pf = new Femme("Pascale");
    libere(pf);
    return 0;
}

```

```

[yunes] ./main
~Individu( )
~Femme(Georgette)
~Individu( )
[yunes]

```

```

class Individu {
    public:
        virtual string donneTonNom()=0;
        virtual ~Individu() { cout << "~Individu()" << endl; };
};
class Femme : public Individu {
    public:
        Femme(string nom) : _Individu(nom) {};
        virtual ~Femme() { cout << "~Femme(" << donneTonNom() << ")" << endl; };
};

void libere(Individu *pi) {
    delete pi;
}

int main()
{
    Femme f("Georgette");
    Femme *pf = new Femme("Pascale");
    libere(pf);
    return 0;
}

```

```

[yunes] ./main
~Femme(Pascale)
~Individu()
~Femme(Georgette)
~Individu()
[yunes]

```



`dynamic_cast<type>`

type

i.e.

downcast

0

`bad_cast`

```
class A {
public:
    virtual void f() { cout << "A::f()" << endl; }
};

class B : public A {
public:
    virtual void f() { cout << "B::f()" << endl; }
};

void f(A *a) {
    B *p = dynamic_cast<B *>(a);
    cout << p << endl;
}

int main() {
    B b;
    f(&b); // ca va marcher
    A a;
    f(&a); // ca va rater...
    return 0;
}
```

```
[yunes] ./main
0x7fff5fbff680
0
[yunes]
```


Downcast

```
class A {
public:
    virtual void f() { cout << "A::f()" << endl; }
};

class B : public A {
public:
    virtual void f() { cout << "B::f()" << endl; }
};

void f(A &a) {
    B &b = dynamic_cast<B &>(a);
    b.f();
}

int main() {
    B b;
    f(b); // ca va marcher
    A a;
    f(a); // ca va rater...
    return 0;
}
```

```
[yunes] ./main
B::f()
terminate called after throwing an
instance of 'std::bad_cast'
    what():  std::bad_cast
Abort
[yunes]
```







`typeid(type)`

`typeid(expression)`

`type_info`

type

expression

type_info

type_info

==

!=

const char *name() const;