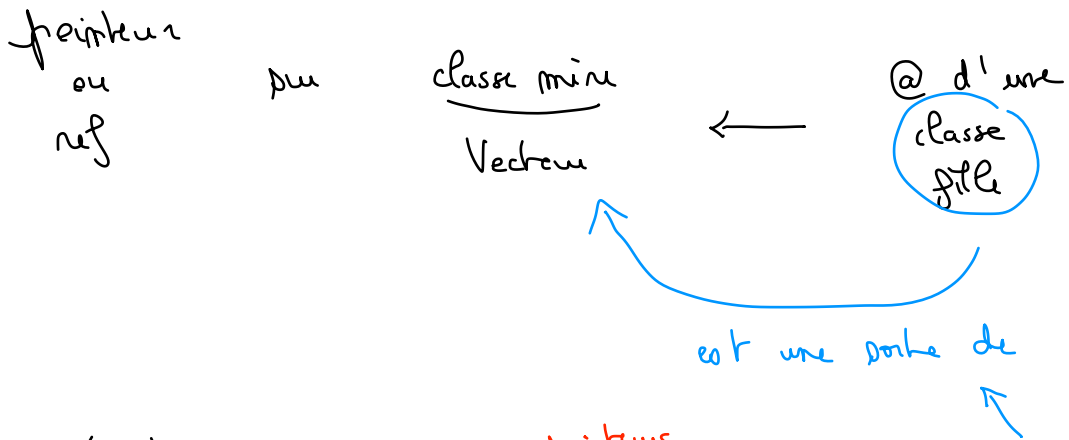
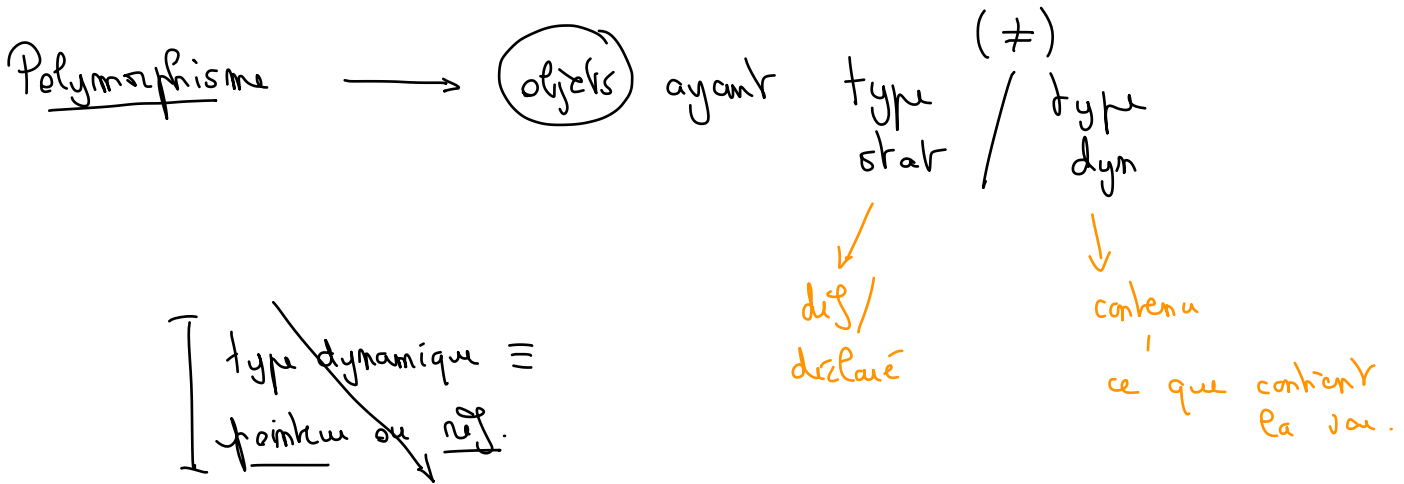


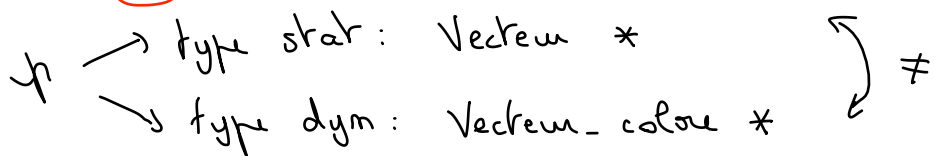
```
class Vecteur {
  ...
  public:
    void afficher ();
}
```

```
class Vecteur_couleur : public Vecteur
{
  private:
    char_couleur [3]; // RGB
  public:
    void afficher ();
}
```



Polymorphisme (ex): ⚠ dynamique → pointeurs / refs.

```
Vecteur * p = new Vecteur_couleur (255, 0, 10);
```



$\varphi \rightarrow$ afficheur()
!

par défaut: celle du type stat
(ici: Vecteur)

si la méth. est virtuelle dans la classe
mine
on utilise celle du type dynamique
(Vecteur-couleur).

ex:

main: Vecteur * tab[3]; // tab tableau de 3
ptr Vecteur.

tab[0] = new Vecteur(10);
tab[1] = new Vecteur-couleur(20, 255, 0, 0);
tab[2] = new Vecteur(20);
for (int i=0; i<3; i++)
tab[i] → afficheur();

Vecteur $\sigma =$ Vecteur-couleur(20, 250, 10, 2);

const. par défaut

une sorte de
Vecteur

partie Vecteur affectée

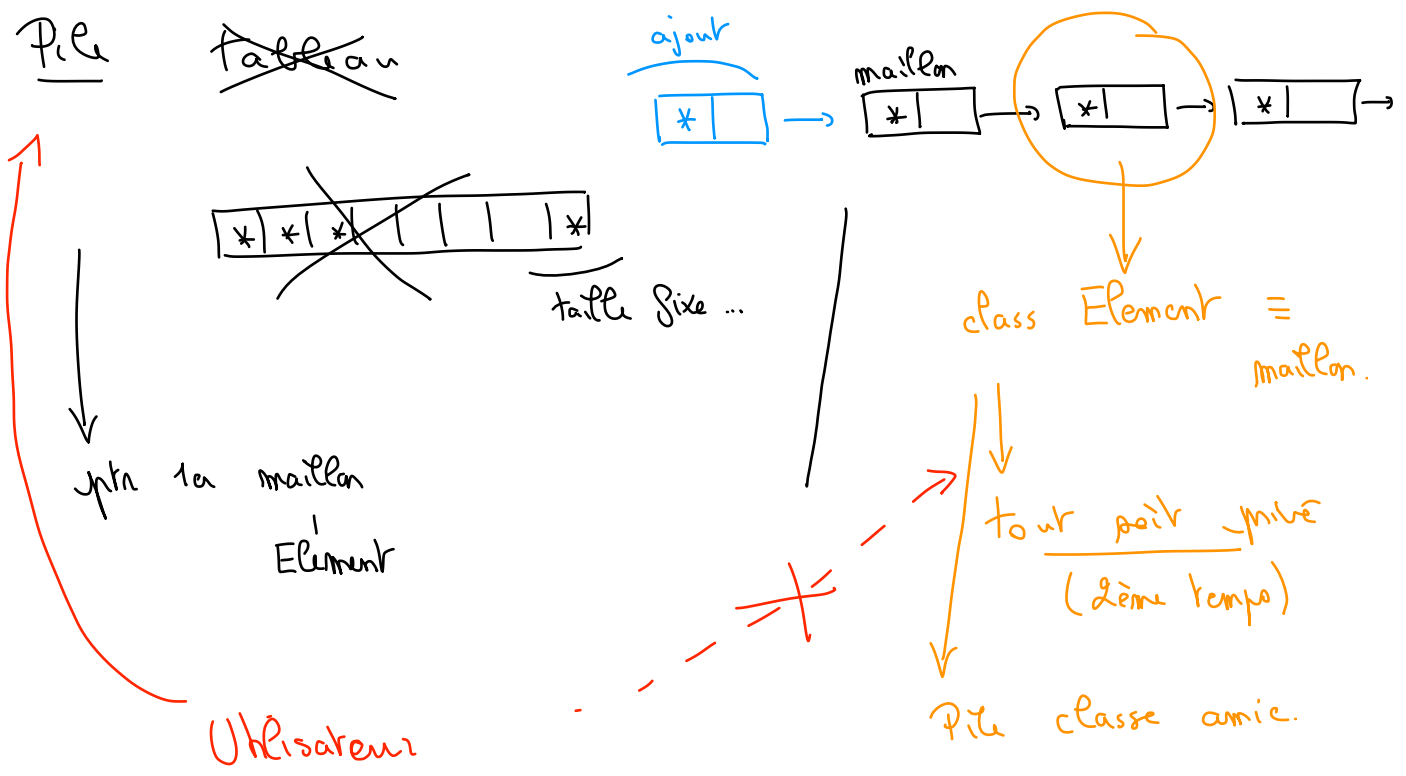
~~-couleur~~

mieux
const. par copie

Vecteur σ (Vecteur-couleur(20, 250, 10, 2));

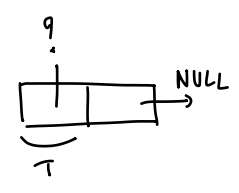
une sorte de Vecteur

σ n'a qu'un type statique → pas de polymorphisme.



```

template <typename T>
class Element {
    T -data;
    Element <T> * -next;
    // Pas de constr. par défaut
};
  
```



comme fait rien mais les données membre
-data, -next
sont construites vs constr. par défaut
-data celui de T
-next celui des ptr & NULL

```

... Pile {
    Element * -data;
    string -nom;
    int -taille;
  }
  
```

```

... Elem {
private:
    T -data;
    Element * -suiv;
  }
  
```

```

-> pile vide
Pile (string s = "toto")
{

```

```

    -data = NULL;
    -nom = s;
    -taille = 0;
}

```

pas de new

push ~ new.

destructeur:

```

~ Pile ()

```

détruit tous les maillons

delete - data

⊕ simple en récursif

détruite tous les maillons

Element (T data, Element * m = NULL)

```

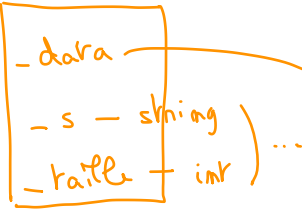
{
    - dat = data;
    - suiv = m;
}

```

pas de new

~ Element () {}

pas de delete



Element *
-data

ptr 1er maillon.
Element

delete - data

détruit l'élément
à l'@ - data
tête ...



⚠ Element n'est pas une classe

↓
Element <T>

```

template <typename T>

```

```

Element_id : public Element <T> {

```

private:

```

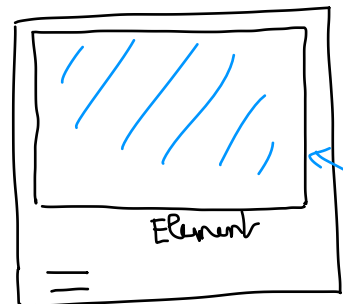
    int - id;

```

```

    static int - cpt;

```



Element_id

```

}

```

Element_id → accès à -data?
 ∈ Element } → si protégé : ek.
 privé ds Element
 ↓
 pas d'accès



```
template <typename Type>
class Element {
private:
    Type _elt;
    Element<Type> *_next;

    Element(const Type &);
    Element(const Element &);
    ~Element();
    template <typename T> friend class PileGen;
};
```

The screenshot shows a VS Code editor with the following code in `Vector15.h`:

```
1 #ifndef VECTOR15
2 #define VECTOR15
3
4 #include "Vector.h"
5
6 class Vector15 : public Vector
7 {
8
9 private:
10     int _coul;
11
12 public:
13     // Constructeur par défaut
14     Vector15(int couleur = 0) : Vector(15){_coul = couleur;};
15     // Constructeur par copie
16     Vector15(const Vector15 & v) : Vector(v){_coul = v._coul;};
17     // Destructeur
18     ~Vector15();
19 };
20 #endif
```

Handwritten annotations in red and green:

- `Vector15` is circled in green.
- `Vector15(int couleur = 0) : Vector(15){_coul = couleur;};` is circled in red, with "spécif. const. Vector" written above it.
- `Vector15(const Vector15 & v) : Vector(v){_coul = v._coul;};` has "avant" written above it.
- `~Vector15();` is circled in green, with "distr. Vect15" written below it.
- A diagram on the right shows a box labeled "Vecteur" containing a smaller box with diagonal lines labeled "const. 'autre'", and a label "Vecteur15" with an arrow pointing to the box.
- Below the diagram, "Vecteur15" is written in green with an arrow pointing to it.
- At the bottom, "distr. Vecteur" is written in green with an arrow pointing to it.
- At the bottom left, "d'execute" is written in green with an arrow pointing to it.

The terminal at the bottom shows the following output:

```
Vector 2 = {0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
exe(2708, 0x111618e00) malloc: *** error for object 0x7f9775504080: pointer being freed was not allocated
exe(2708, 0x111618e00) malloc: *** set a breakpoint in malloc_error_break to debug
zsh: abort ./exe
(base) hugo@MBP-de-Hugo tp2 %
```