Analyse et Programmation Orientées Objets / C++

Classe Pile (V1)

Généricité – Exemple classe Pile

➡ **Spécifications fonctionnelles**

Gestion d'une file LIFO

- Taille maximum fixée à l'exécution
- Indépendance vis à vis du type des objets gérés
- Contrôle de la validité
- Accesseurs cardinal et taille
- Méthodes empiler, depiler et vider
- Libération mémoire
- Gestion intégrée des erreurs

La classe Pile - V1 (1)

```
#include <stdio.h>
class Pile {
private :
void** m_pT;
int    m_sommet;
int    m_taille;

public :
--- Cf description transparent suivant

};
```

La classe Pile - V1 (2)

Pile();

Pile(int);

~Pile();

--- Cf description transparent suivant

La classe Pile - V1 (3)

```
inline void veto()   {
    m_pT=NULL;
    m_sommet=-1;
    m_taille= -1;
}
```

--- Cf description transparent suivant

La classe Pile - V1 (4)

```
inline bool neutre() {
   return m_taille==0;
}
```

--- Cf description transparent suivant

La classe Pile - V1 (5)

```
inline bool ok()     {
   return m_pT !=NULL  &&
              m_sommet >= 0 &&
              m_taille >= 0;
}
```

--- Cf description transparent suivant

La classe Pile - V1 (6)

```
inline bool nok()    {return !ok();}
```

--- Cf description transparent suivant

La classe Pile - V1 (7)

```
inline int  cardinal() {return m_sommet;}
inline int  taille()   {return m_taille;}
```

--- Cf description transparent suivant

La classe Pile - V1 (8)

```
void* empiler(void*);
void* depiler ();
void   vider();
```

--- Cf description transparent suivant

La classe Pile - V1 (9)

```
Pile :: Pile () {
    m_pT       = new void*[1];
    m_sommet = 0;
    m_taille    = 0;
}
```

La classe Pile - V1 (10)

```
Pile :: Pile (int taille) {
    if (taille <0) throw -2.1;
    m_pT       = new void*[taille];
    m_sommet = 0;
    m_taille    = taille;
}
```

La classe Pile - V1 (11)

```
void* Pile :: empiler (void* pE) {
    if (nok()) throw -2.0;
    if (pE==NULL) throw -2.1;
    if (m_sommet == taille) throw -3.0;
    m_pT [m_sommet++] = pE;
    return pE;
}
```

La classe Pile - V1 (12)

```
void* Pile :: depiler () {
    if (nok()) throw -2.0;
    if (m_sommet == 0) throw -3.0;
    return  m_pT [--m_sommet] ;
}
```

La classe Pile - V1 (13)

```
void Pile :: vider () {
    if (nok()) return;
    while (cardinal() > 0) delete depiler();
}
```

La classe Pile - V1 (14)

```
void main () {
Pile p1(10);
RxR *pZ1= new RxR(1, -2), *pZ2= new RxR(-7, 11.5f);
RxR *pW;
      p1.empiler(pZ1);
      p1.empiler(pZ2) ;
     ---
      pW= p1.depiler();
      ---
}
```

La classe Pile - V1 (15)

➡ **Inconvénients de la version actuelle**

- Impossibilité de contrôler le type des objets pointés
- Impossibilité d'accéder à la valeur des objets
- Impossibilité de surcharger les opérateurs (=, ==, …)
- Impossibilité d'assurer la persistance