Analyse et Programmation Orientées Objets / C++

Classe Pile (V2)

Objectifs de la version V2

➡ **Cahier des charges**

- Spéc. fonctionnelles initiales reconduites
- Mise en place de la persistance
- Surcharge des opérateurs <<, = et ==
- Spéc. d'interfaces inchangées
- Contraintes de réalisation inchangées

La classe Pile – V2 (1)

```
#include <stdio.h>
#define _Pile_
template <class G> class Pile {
private :
void** m_pT;
int      m_sommet;
int      m_taille;
public :
--- Cf description transparent suivant

};
```

La classe Pile – V2 (2)

Pile();

Pile(int);

~Pile();

--- Cf description transparent suivant

La classe Pile – V2 (3)

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

--- Cf description transparent suivant

La classe Pile – V2 (4)

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

--- Cf description transparent suivant

La classe Pile – V2 (5)

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

--- Cf description transparent suivant

La classe Pile – V2 (6)

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

--- Cf description transparent suivant

La classe Pile – V2 (7)

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

--- Cf description transparent suivant

La classe Pile – V2 (8)

G empiler(G);

G depiler ();

void vider();

--- Cf description transparent suivant

La classe Pile – V2 (9)

```
#ifndef _Pile_
# include ”Pile.h”
#endif
template <class G>
Pile <G>:: Pile () {
    m_pT        = new G[1];
    m_sommet = 0;
    m_taille    = 0;
}
```

La classe Pile – V2 (10)

```
template <class G>
Pile <G>:: Pile (int taille) {
    if (taille <0) throw -2.1;
    m_pT        = new G[taille];
    m_sommet = 0;
    m_taille    = taille;
}
```

La classe Pile – V2 (11)

```
template <class G>
G Pile <G>:: empiler (G 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 – V2 (12)

```
template <class G>
G Pile <G>:: depiler () {
    if (nok()) throw -2.0;
    if (m_sommet == 0) throw -3.0;
    return  m_pT [--m_sommet] ;
}
```

La classe Pile – V2 (13)

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

La classe Pile – V2 (14)

```
# include ”Pile.h”
# include ”Pile.cpp”
# include ”Pile_2.cpp”

void main () {
Pile <RxR> p1(10);
---
}
```

La classe Pile – V2 (15)

```
---
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 - V2 (16)

➡ **Avantages de la version V2**

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

La classe Pile - V2 (17)

➡ **Inconvénients de la version V2**

- Taille du code généré
- Complexification des codes sources