2. tester si une pile est vide,
3. ajouter un premier élément à une pile,
4. accéder au premier élément de la pile,
5. retirer le premier élément de la pile.

Le premier élément d'une pile s'appelle le *sommet* et non la tête de la pile, car on représente d'habitude les piles verticalement, comme des piles d'assiettes. L'opération consistant à retirer le sommet d'une pile peut être comparée à l'opération consistant à accéder à la queue d'une liste, mais la différence est que l'instruction `l2 = l1.tl ;` ne modifie pas la liste `l1`, alors que l'instruction `pop(p) ;`, qui retire le sommet de la pile `p`, modifie cette pile.

Puisque cette instruction modifie son argument, il est nécessaire de représenter une pile, non comme une liste, mais comme une liste enveloppée

```
class List {
 int hd;
 List tl;

 List (final int x, final List y) {this.hd = x; this.tl = y;}}

class Pile  {
 List c;

 Pile (final List x) {this.c = x;}}
```

Et on programme les cinq opérations ci-dessus ainsi

```
static Pile empty () {
 return new Pile(null);}

static boolean testempty (final Pile l) {
 return l.c == null;}

static void push (final int a, final Pile l) {
 l.c = new List(a,l.c);}

static void pop (final Pile l) {
 l.c = l.c.tl;}

static int top (final Pile l) {
 return l.c.hd;}
```

On peut ensuite utiliser ces opérations, par exemple dans le programme suivant