

# Premier rapport - Semaine 2

Groupe 2.2

Vendredi 25 septembre 2014

## Question 1

## Question 2

## Question 3

## Question 4

## Question 5

## Question 6 : Arnaud Dethise

Soient deux piles,  $p_1$  et  $p_2$ .

push(A) : ajouter A dans la liste non-vide ( $p_1$ ).

pop() : vider tous les éléments de la liste non-vide ( $p_1$ ) dans la liste vide ( $p_2$ ), à l'exception du dernier élément. Retourner le dernier élément.  $p_2$  est maintenant la liste non-vide,  $p_1$  est la liste vide.

Ainsi, les opérations suivantes :

```
1 push(1)
  push(2)
3 push(3)
  a = pop()
5 b = pop()
```

se traduisent en utilisant des files par :

```
1 p1.enqueue(1)           % p1: [1]      p2: []
  p1.enqueue(2)           % p1: [2,1]    p2: []
3 p1.enqueue(3)           % p1: [3,2,1]   p2: []
  p2.enqueue(p1.dequeue()) % p1: [3,2]   p2: [1]
5 p2.enqueue(p1.dequeue()) % p1: [3]     p2: [2,1]
  a = p1.dequeue()        % p1: []       p2: [2,1]
7 p1.enqueue(p2.dequeue()) % p1: [1]     p2: [2]
  b = p2.dequeue()        % p1: [1]     p2: []
```

La complexité de la fonction push est alors de  $\Theta(1)$  et celle de pop est de  $\Theta(N)$  où N est le nombre d'élément dans la pile. Il est possible d'implémenter la pile de telle sorte que les complexités de push et pop soient inversées.

Cette solution est donc moins efficace que les implémentations normales d'une pile, dont la complexité est en temps constant pour push() et pop().

Nous avons également discuté en séance de la possibilité d'implémenter une file à partir de deux piles. La solution, pour laquelle la fonction push a une complexité constante  $\Theta(1)$  et la fonction pop a une complexité moyenne  $\Theta(1)$ , est indiquée ci-dessous.

```
1 pile A
  pile B
4 function enqueue(A) :
  A.push(A)
6
```

```

8   function dequeue() :
    if B.isEmpty() :
        while not A.isEmpty() :
            B.push(A.pop())
    return B.pop()
10

```

## Question 7 : Gil De Grove, Romain Henneton

Fonction d'écriture :

```

1  public static int WriteInFile(String filePath, String toWrite)
3  {
5      Path file = Paths.get(filePath);
    if (! file.toFile().exists())
7      {
        try {
9            Files.createFile(file);
        } catch (IOException e) {
11           e.printStackTrace();
        }
13     }
    if (! file.toFile().canWrite()){
15         return -1;
17     }
19     try {
        Files.write(file, toWrite.getBytes("US-ASCII"));
21     } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
23     } catch (IOException e) {
        e.printStackTrace();
25     }
    return 0;
27 }

```

Fonction de lecture :

```

1  public static List<String> ReadFromFile(String filePath)
    {
3      List<String> text = new ArrayList<String>();
        Path file = Paths.get(filePath);
5        if(! file.toFile().exists())
        {
7            return text;
        }
9
        try {
11           text = Files.readAllLines(file, Charset.forName("US_ASCII"));
        } catch (IOException e) {
13           e.printStackTrace();
        }
15     return text;
    }
17

```