



# BDDA

TP n°2  
Semaine 39



# Roadmap

## Finir le TP n°1

Créer l'architecture du DiskManager (API = squelette !)

Créer l'architecture pour le BufferManager

Coder et enrichir les fonctions

# Informations générales

Les TP de 17-20h commenceront à 18h (vous pouvez commencer dès 17h)

Pas de changement des méthodes de l'API, garder la dénomination des TPs !

# Gestion de l'espace disque

Stockage : Data\_n.fr , n entier > 0

Méthodes de lecture/écriture dans un fichier binaire...

... Rappel en JAVA , usage des streams !

```
0000000 0000 0001 0001 1010 0010 0001 000
0000010 0000 0010 0000 0020 0000 0010 000
0000020 0000 0001 0004 0000 0000 0000 000
0000030 0000 0000 0000 0010 0000 0000 000
0000040 0004 8384 0084 c7c8 00c8 4748 004
0000050 00e9 0a09 0009 a8a9 00a9 2828 002
0000060 00fc 1819 0019 9898 0098 d9d8 00d
0000070 0057 7b7a 007a bab9 00b9 3a3c 003
0000080 8888 8888 8888 8888 288e be88 888
0000090 3b83 5788 8888 8888 7007 778e 882
00000a0 d01f 7abd 8818 8888 407c 585f 881
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88b
00000c0 8a18 880c e841 c988 b328 0871 088
00000d0 a948 5802 5884 7e81 3788 1ab4 5a8
00000e0 3d86 dcb8 5cbb 8888 8888 8888 888
00000f0 8888 8888 8888 8888 8888 8888 888
0000100 0000 0000 0000 0000 0000 0000 000
*
0000130 0000 0000 0000 0000 0000 0000 000
000013e
```

# Gestion de l'espace disque



```
00000000 0000 0001 0001 1010 0010 0001 000
0000010 0000 0010 0000 0020 0000 0010 000
0000020 0000 0001 0004 0000 0000 0000 000
0000030 0000 0000 0000 0010 0000 0000 000
0000040 0004 8384 0084 c7c8 00c8 4748 004
0000050 00e9 0a09 0009 a8a9 00a9 2828 002
0000060 00fc 1819 0019 9898 0098 d9d8 00d
0000070 0057 7b7a 007a bab9 00b9 3a3c 003
0000080 8888 8888 8888 8888 288e be88 888
0000090 3b83 5788 8888 8888 7007 778e 882
00000a0 d01f 7abd 8818 8888 407c 585f 881
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88b
00000c0 8a18 880c e841 c988 b328 0871 088
00000d0 a948 5802 5884 7e81 3788 1ab4 5a8
00000e0 3d86 dc88 5cbb 8888 8888 8888 888
00000f0 8888 8888 8888 8888 8888 8888 888
0000100 0000 0000 0000 0000 0000 0000 000
*
0000130 0000 0000 0000 0000 0000 0000 000
000013e
```

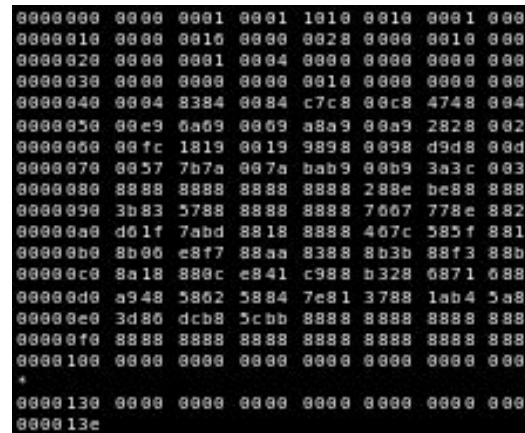
# Gestion de l'espace disque

```
1 import java.io.*;
2
3 /**
4  * Copy one file to another using low level byte streams, one byte at a time.
5  * @author www.codejava.net
6  */
7 public class CopyFiles {
8     public static void main(String[] args) {
9         if (args.length < 2) {
10             System.out.println("Please provide input and output files");
11             System.exit(0);
12         }
13
14         String inputFile = args[0];
15         String outputFile = args[1];
16
17
18         try {
19             InputStream inputStream = new FileInputStream(inputFile);
20             OutputStream outputStream = new FileOutputStream(outputFile);
21         } {
22
23             int byteRead;
24
25             while ((byteRead = inputStream.read()) != -1) {
26                 outputStream.write(byteRead);
27             }
28
29         } catch (IOException ex) {
30             ex.printStackTrace();
31         }
32     }
33 }
```

```
00000000 0000 0001 0001 1010 0010 0001 000
00000010 0000 0010 0000 0020 0000 0010 000
00000020 0000 0001 0004 0000 0000 0000 000
00000030 0000 0000 0000 0010 0000 0000 000
00000040 0004 8384 0084 c7c8 00c8 4748 004
00000050 00e9 0a09 0009 a8a9 00a9 2828 002
00000060 00fc 1819 0019 9898 0098 d9d8 00d
00000070 0057 7b7a 007a bab9 00b9 3a3c 003
00000080 8888 8888 8888 8888 288e be88 888
00000090 3b83 5788 8888 8888 7007 778e 882
000000a0 d01f 7abd 8818 8888 407c 585f 881
000000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88b
000000c0 8a18 880c e841 c988 b328 0871 088
000000d0 a948 5802 5884 7e81 3788 1ab4 5a8
000000e0 3d80 dcb8 5cbb 8888 8888 8888 888
000000f0 8888 8888 8888 8888 8888 8888 888
00001000 0000 0000 0000 0000 0000 0000 000
*
00001030 0000 0000 0000 0000 0000 0000 000
0000103e
```

# Gestion de l'espace disque (1.8+)

```
1 import java.io.*;
2 import java.nio.file.*;
3
4 /**
5  * Copy one file to another using low level byte streams, one byte at a time.
6  * @author www.codejava.net
7  */
8 public class CopyFilesNIO {
9     public static void main(String[] args) {
10         if (args.length < 2) {
11             System.out.println("Please provide input and output files");
12             System.exit(0);
13         }
14
15         String inputFile = args[0];
16         String outputFile = args[1];
17
18
19         try {
20             long start = System.currentTimeMillis();
21
22             byte[] allBytes = Files.readAllBytes(Paths.get(inputFile));
23             Files.write(Paths.get(outputFile), allBytes);
24
25             long end = System.currentTimeMillis();
26             System.out.println("Copied in " + (end - start) + " ms");
27         } catch (IOException ex) {
28             ex.printStackTrace();
29         }
30     }
31 }
```



Using Files API !

# Rappel Buffer Manager

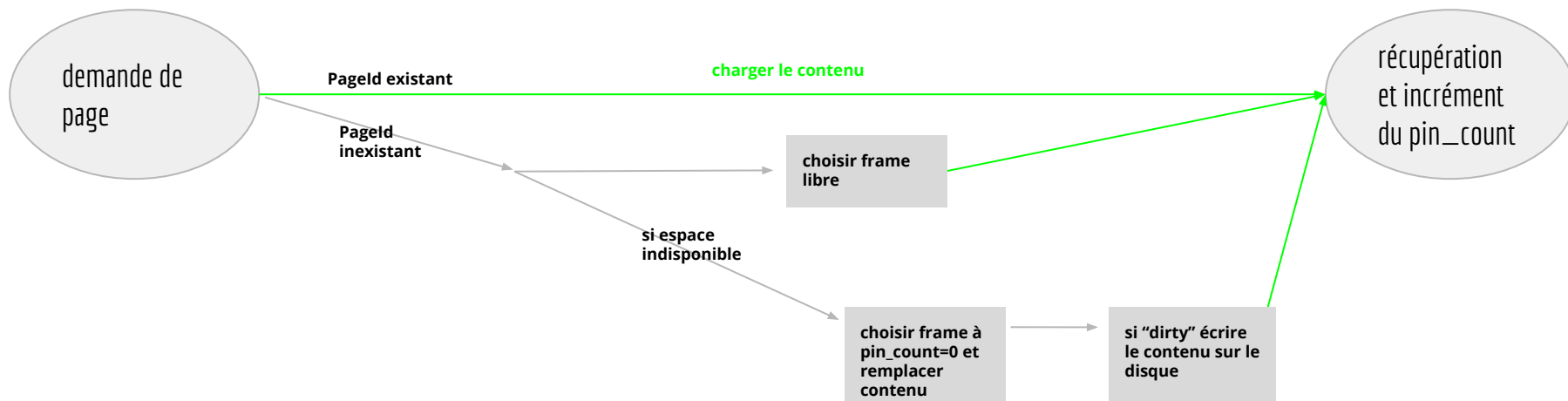
responsable du transfert des pages entre le disque et la RAM

- Partition de la mémoire
- Remplissage des partitions en fonction des demandes (buffer pool)
- Stocke le *PageId* lorsque occupé, vérifie que la page existe avant chargement
- *Flag dirty* : écriture sur disque
- Limitation de la mémoire = Obligation de remplacement (*pin\_count*) / **page remplaçable ssi *pin\_count*=0**



# Rappel Buffer Manager

- Conservation d'une table d'informations  
    < PageId, pin\_count, dirty >



# Rappel Politique de remplacement

Quand ? Lorsque les cases sont pleines et qu'un remplacement intervient...

## LRU : Least Recently used

noter le dernier passage à `pin_count` nul `t_unpin`

remplacer le contenu qui a été unpinned le moins récemment / recherche du `max(t_unpin)`

Problème de flooding, intervient à chaque scan...

## Clock

Notion de reference bit

Lorsque `pin_count` nul, `ref_bit` à 1

Boucler sur les pages, si (0,1) passer `ref_bit` à 0 et si (0,0) on choisit cette page en remplacement

arrêt lorsque la page est choisi !