FILE

DESCRIPTION :

Une file d’attente est une suite d’objets de même type, possédant un ordre bien précis, et dont le nombre est variable

(Une file peut être vide !)

L’ajout se fait en queue (fin)

Le retrait se fait en tête (début)

INTERFACE (AVEC JAVADOC):

public interface File<E> {

/\*\*

\* renvoie le nombre d'elements qui se trouvent dans la file

\* @return nombre d'elements

\*/

public **int taille();**

/\*\*

\* verifie si la file est vide

\* @return true si la file est vide, false sinon

\*/

public **boolean estVide();**

/\*\*

\* ajoute un element en fin de file (queue)

\* @param element l'element a ajouter

\*/

public **void enfile(E element)**;

/\*\*

\* renvoie l'element qui se trouve en tete de file et l'enleve

\* @return l'element en tete

\* @throws FileVideException si la file est vide

\*/

public **E defile() throws FileVideException**;

/\*\*

\* renvoie l'element qui se trouve en tete de file sans l'enlever

\* @return l'element en tete

\* @throws FileVideException si la file est vide

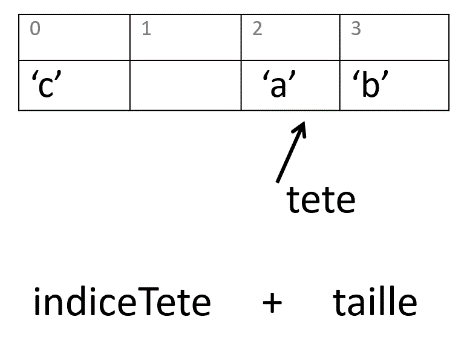
\*/

public **E premier()throws FileVideException**;

}

Implémentation retenue :

Table circulaire redimensionnable



JEU DE TESTS (sous forme d’un scénario avec au départ une file vide):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N°  test | Instruction | Exception  ? | return  attendu | file.taille() attendu | file.toString()  attendu |
| 1 | file.enfile('a') |  |  | 1 | a |
| 2 | file.enfile('b') |  |  | 2 | a b |
| 3 | file.premier() |  | a | 2 | a b |
| 4 | file.defile() |  | a | 1 | b |
| 5 | file.defile() |  | b | 0 |  |
| 6 | file.defile() | FileVideException |  |  |  |
| 7 | file.premier() | FileVideException |  |  |  |
| 8 | file.enfile('c'); |  |  | 1 | c |

COUTS DES METHODES :

|  |  |
| --- | --- |
| taille() | O(1) |
| estVide() | O(1) |
| enfile() | O(1) |
| defile() | O(1) |
| premier() | O(1) |