Cours 8 : Exceptions, un peu de graphique

1. Traiter les exceptions usuelles
2. Créer ses propres exceptions
3. Exemples: les files.
4. Quelques éléments sur les graphiques

Exceptions

- Une exception est lancée quand une situation anormale se produit
- Une exception est un objet de la classe `Exception`

Exemples:

1. Division par 0: `ArithmeticException`
2. Sortir des limites de la taille d'un tableau: `IndexOutOfBoundsException`
3. Appliquer une méthode à la référence vide:: `NullPointerException`

Exemple

```
public static void calculSimple (int a, int b){
     int res = a/b;
     System.out.println(res);
     System.out.println(a*b);
    }
```

```
Exception in thread "main" java.lang.ArithmeticException:
                        / by zero
        at Division.calculSimple(Division.java:18)
        at Division.main(Division.java:32)
```

Exemple

```
public static void calcul (int a, int b){
        int res;
        try {res = a/b;}
        catch (Exception e) {
            System.out.println("attention division par 0");
            res = -99999999;}
        System.out.println(res);
        System.out.println(a*b);
    }

% java Calculer 12 0

attention division par 0
-99999999
0
```

Forme générale

```
try{

}
catch( ArithmetiqueException e) {

   }
catch (IndexOutOfBoundsException e) {

 }
```

Forme plus générale

```
try{

}
catch(Exception e) {

   }
finally  {

 }
```

Lecture d'un fichier

IOException

```
static BufferedReader entree;

static void ouvrir (String f){
    // ouvre le fichier qui prend le nom entree
    try{
        entree = new BufferedReader(new FileReader(f));
    }
    catch (IOException e) {
          System.out.println("erreur: fichier"
             + f + "non existant");}
    }
```

```
do {
     try {
         i++;
         mesLignes[i] = entree.readLine();
      }
      catch (IOException e) {
            mesLignes[i]  = null;};
       }
 while (!mesLignes[i].equals(""));
 return mesLignes;
```

Exceptions définies par le programmeur

- On peut définir une nouvelle classe formées d'exceptions
- C'est une classe qui doit hériter de la classe `Exception`
- Il faut préciser quand cette exception est soulevée (lancée) `throw`
- Une fonction , méthode qui lance une exception doit être déclarée comme telle

```
int chercher() throws Exception {
```

Exemple

```
class FIFOExcp extends Exception {
      FIFOExcp (String x){
              System.out.println(x );}
}
```

Les Files

Constante: La file vide F_0

Opérations

Ajouter:	File	$\times$ Entier	$\rightarrow$	File
Supprimer	File		$\rightarrow$	File
Valeur	File		$\rightarrow$	Entiers
Vide?	File		$\rightarrow$	Booléens

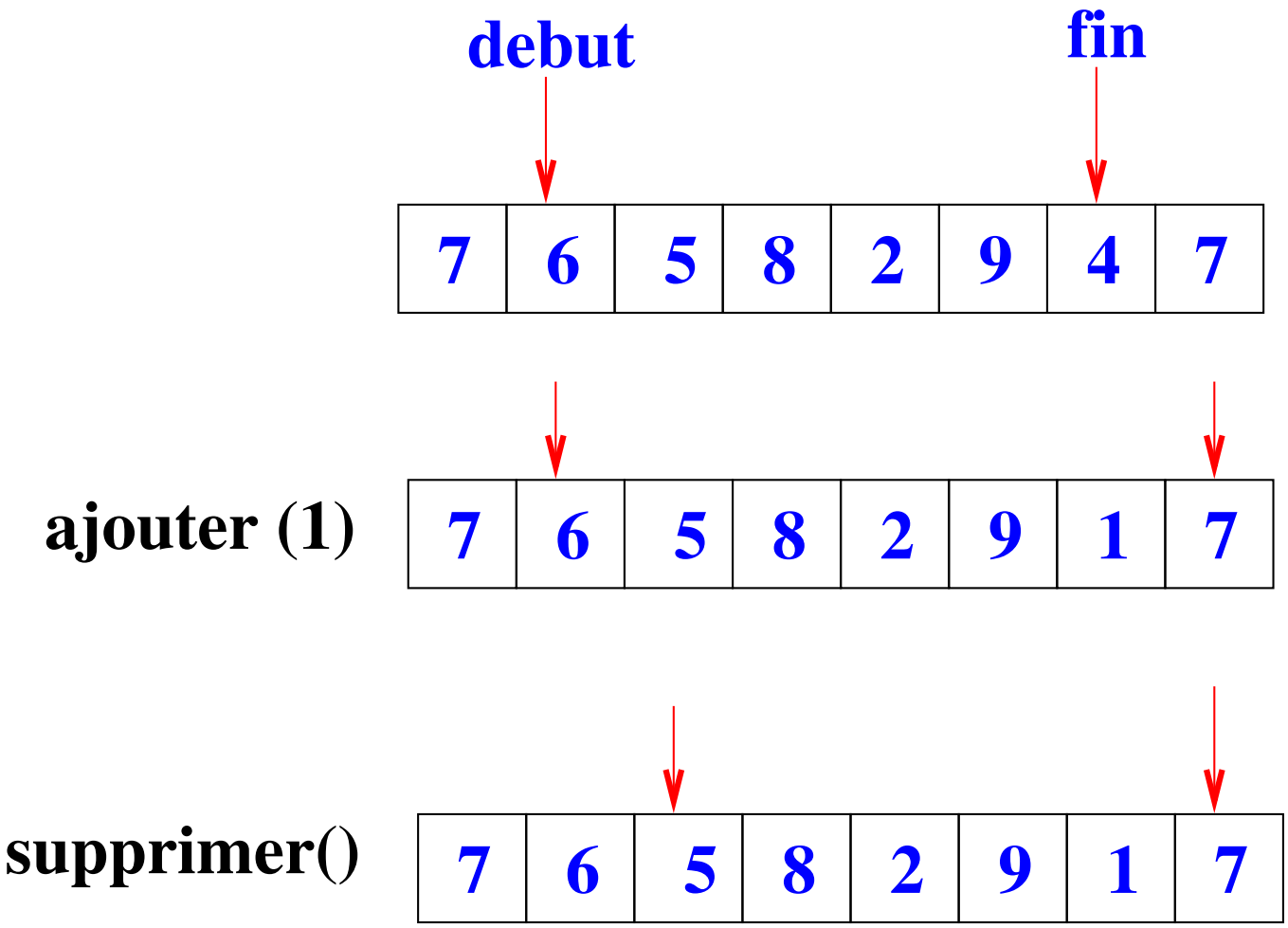
Valeur, Supprimer ne sont pas définis pour F_0

Equations des Files

Vide? (F_0)	=	true
Vide? (Ajouter(F, a))	=	false
Supprimer (Ajouter(F,a))	=	Ajouter(Supprimer(F), a) Si F $\neq F_0$
Valeur (Ajouter (F,a))	=	Valeur (F) Si F $\neq F_0$
Supprimer (Ajouter(F_0, a))	=	F_0
Valeur (Ajouter (F_0,a))	=	a

Technique de représentation

- On utilise un tableau et deux entiers
- `debut` donne la position du premier élément de la file si elle n'est pas vide
- `fin` est égal à $1+$ la position du dernier élément de la file, (c'est à dire la position où on pourra ajouter un élément si la file n'est pas pleine).
- On réutilise la place libérée en calculant modulo la taille du tableau
- `pleine, vide` sont des booléens qui permettent de savoir si la file est pleine ou vide


debut
fin
7 6 5 8 2 9 4 7
ajouter (1)
7 6 5 8 2 9 1 7
supprimer()
7 6 5 8 2 9 1 7

Réalisation des Files en Java

```
class FIFO {
    private int          debut, fin;
    private boolean      pleine, vide;
    private int[]        contenu;
    private int          longueur;
    FIFO (int n) {
        debut = 0; fin = 0;
        pleine = false; vide = true;
        longueur = n;
        contenu = new int[n];
        }
```

```
int valeur () throws FIFOExcp{
    if (vide)
       throw new FIFOExcp("Valeur sur file vide");
    else return  contenu[debut];
}
```

```
private static int Successeur(int i) {
   return  (i+1) % longueur;
}

static FIFO vide () {
   return new FIFO();}

boolean estVide() {
   return  vide;}

boolean estPleine() {
   return  pleine;}
```

```
void ajouter (int x) throws FIFOExcp {
    if (pleine)
        throw new FIFOExcp ("ajout dans file pleine");
    else{
        contenu[fin] = x;
        fin = successeur(fin);
        vide = false;
        pleine = (fin == debut);
    }
}
```

```
    void supprimer () throws FIFOExcp {
         if (vide)
             throw new FIFOExcp("Suppresion sur File Vide");
         else {
             debut =  successeur(debut);
             vide = (fin == debut);
             pleine = false;
         }
     }

}
```

```
public static void main (String[] args){
    FIFO fil = new FIFO(5);
    try{
    fil.ajouter(1);fil.ajouter(2); fil.ajouter(3);
    System.out.println (fil.valeur());
    fil.supprimer(); System.out.println (fil.valeur());
    fil.supprimer(); System.out.println (fil.valeur());
    fil.supprimer(); System.out.println (fil.valeur());
    }
    catch (FIFOExcp e) {};
    System.out.println(" je continue quand meme");

}
```

Quelques fonctions graphiques

Le paquetage `java.awt` Abstract Window Toolkit

- Fenêtres
- Graphiques
- Exemple : le triangle de Sierpinski

Fenêtres

- Classe `Window, Frame`
- Rend une plage graphique où on peut dessiner
- Voir toutes les fonctions possibles dans la documentation

Graphique

- Dans une fenêtre ou un applet
- Voir documentation
- Tracer des lignes

Triangle de Sierpinski

- Taille 1: Un triangle equilatéral
- Taille n : Trois triangles de taille n-1
- Fonction récursive

```
import java.awt.*;



class Sierpinski{

  static Graphics initFenetre(){
       int largeur = 650, hauteur = 500;
       Frame  fenetre = new Frame("Sierpinski");
       fenetre.setBounds(1000,1000, largeur, hauteur);
       fenetre.setBackground(Color.black);
       fenetre.setForeground(Color.red);
       fenetre.setVisible(true);
       return fenetre.getGraphics();
    }
```

```
static void  sierpDessin(int x, int y, int a, int n, Graphics g)
     double rac3 = Math.sqrt(3);
     int b =  (int) rac3*a/2;
     if (n == 1) {
         g.drawLine (x,y, x- a/2, y+b);
         g.drawLine (x-a/2, y+b, x +a/2, y+b);
         g.drawLine(x+a/2, y+b, x, y);
         }
     else {
         int a1 =  a/2, a2 =  a1/2, b1 =  b/2;
         sierpDessin(  x,   y  ,   a1, n-1, g);
         sierpDessin(x-a/4,  y+b/2  ,   a1, n-1, g);
         sierpDessin(x+a/4, y+b/2,   a1, n-1, g);
     }
}
```

```
public static void main(String[] args) {

    int n = Integer.parseInt(args[0]);
    Graphics gg = initFenetre();
    sierpDessin(325, 40, 600, n, gg);

   }
}
```