

Fms Academy 2022

Formation Java Spring Angular

Module 7 : Programmation avancée avec Java

Sommaire

2

- Exceptions
- Flux E/S
- Threads
- Expressions Lambdas
- Expressions régulières
- Stream
- Synthèse sur la Poo
- Design Pattern
- Next
- Ressources

Que s'est-t-il passé ?

```
1 package fr.ldnr.Exception;
2
3 public class TestException {
4     public static void main(String[] args) {
5         int [] tablo = {2,45,6,78,95};
6
7         System.out.println(tablo[0]);
8         System.out.println(tablo[5]);
9         System.out.println(tablo[4]);
10    }
11 }
12
```

Console

```
<terminated> TestException [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe
2
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at fr.ldnr.Exception.TestException.main(TestException.java:8)
```

Où est passé ma 3ème instruction ?

En bref, comment éviter un arrêt du programme tout en prenant en compte ce qui s'est passé ?

Exceptions

```
3 public class TestException {
4     public static void main(String[] args) {
5         int [] tablo = {2,45,6,78,95};
6
7         try {
8             System.out.println(tablo[0]);
9             System.out.println(tablo[5]);
10            System.out.println(tablo[4]);
11        }
12        catch(ArrayIndexOutOfBoundsException e) {
13            System.out.println("youpi nous avons capturé l'exception ici, elle est de type :\n " + e);
14            System.out.println("mais le programme s'est quand même arrêter sans avoir afficher 95 !");
15        }
16    }
17 }
18
```

Exception de dépassement de l'indice du tableau, capturée par nos soins ici

Console

<terminated> TestException [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (11 mai 2021 à 16:58:13)

2

youpi nous avons capturé l'exception ici, elle est de type :
java.lang.ArrayIndexOutOfBoundsException: 5
mais le programme s'est quand même arrêter sans avoir fini le job !

```
7 int [] tablo = { 2, 45, 6, 78, 95};
8 //indice : 0 1 2 3 4      tablo[0] contient la valeur 2 ...
9 int val = 0;
10 Scanner scan = new Scanner(System.in);
11
12 while(val++ < 5) {
13     System.out.println("selectionner un indice de 0 à 4 pour voir la valeur correspondante ");
14     int indice = scan.nextInt(); //nous devrions vérifier si l'indice est dans l'intervalle
15     try {
16         System.out.println("valeur à l'indice " + indice + " : " + tablo[indice]);
17     }
18     catch(ArrayIndexOutOfBoundsException e) {
19         System.out.println(e);
20     }
21     System.out.println("on passe toujours par ici :)");
22 }
23
```

Console

<terminated> TestException [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (11 mai 2021 à 17:38:48)

selectionner un indice de 0 à 4 pour voir la valeur correspondante

0

valeur à l'indice 0 : 2

on passe toujours par ici :)

selectionner un indice de 0 à 4 pour voir la valeur correspondante

-1

java.lang.ArrayIndexOutOfBoundsException: -1

on passe toujours par ici :)

selectionner un indice de 0 à 4 pour voir la valeur correspondante

f

Exception in thread "main" java.util.InputMismatchException

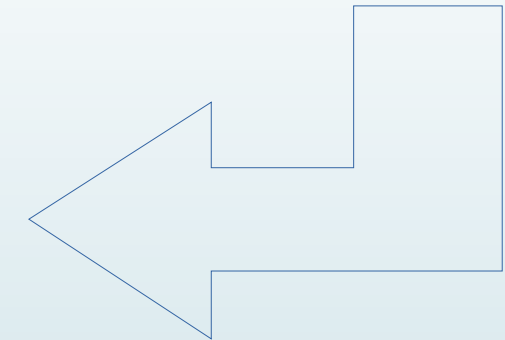
at java.util.Scanner.throwFor(Unknown Source)

at java.util.Scanner.next(Unknown Source)

at java.util.Scanner.nextInt(Unknown Source)

at java.util.Scanner.nextInt(Unknown Source)

at fr.lidnr.Exception.TestException.main(TestException.java:14)



On résous un pb
et il en apparaît un autre...

Exceptions

```
13 while(val++ < 5) {
14     System.out.println("selectionner un indice de 0 à 4 pour voir la valeur correspondante ");
15     try {
16         int indice = scan.nextInt(); //nous devrions vérifier si l'indice est dans l'intervalle
17         System.out.println("valeur à l'indice " + indice + " : " + tablo[indice]);
18     }
19     catch(ArrayIndexOutOfBoundsException e) {
20         System.out.println("vous avez saisi une valeur en dehors des indices du tableau -> " + e.getMessage());
21     }
22     catch(InputMismatchException e) {
23         System.out.println("vous avez saisi une valeur inattendue ici, pb de type !");
24         e.printStackTrace();
25     }
26     finally {
27         System.out.println("on passe toujours par ici :)");
28         scan.nextLine();
29     }
30 }
31 scan.close();
32 }
33 }
```

Console

TestException [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (11 mai 2021 à 18:32:11)

selectionner un indice de 0 à 4 pour voir la valeur correspondante

-1

vous avez saisi une valeur en dehors des indices du tableau -> -1

on passe toujours par ici :)

selectionner un indice de 0 à 4 pour voir la valeur correspondante

h

vous avez saisi une valeur inattendue ici

[java.util.InputMismatchException](#)

```
at java.util.Scanner.throwFor(Unknown Source)
at java.util.Scanner.next(Unknown Source)
at java.util.Scanner.nextInt(Unknown Source)
at java.util.Scanner.nextInt(Unknown Source)
at fr.lidnr.Exception.TestException.main(TestException.java:16)
```

on passe toujours par ici :)

selectionner un indice de 0 à 4 pour voir la valeur correspondante

Exceptions

Comment gérer ce problème sans arrêter le programme ?

```
32 public static void main(String[] args) {
33     System.out.println(5/0);
34 }

Console

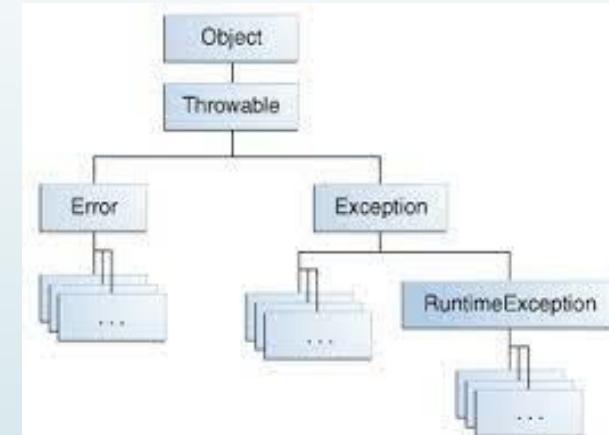
<terminated> TestThrow [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe
Exception in thread "main" java.lang.ArithmeticException: / by zero
at fr.lidnr.Exception.TestThrow.main(TestThrow.java:33)
```

```
TestThread.java TestRunnable.java TestException.java TestThrow.java

10 public class TestThrow {
11     public static void main(String[] args) {
12         //1ère manière : anticiper et générer une exception
13         // double a = 5, b = 0; //en effet, on ne doit surtout pas laisser faire cette opération
14         // if(b == 0) throw new RuntimeException("Impossible de diviser par zéro !");
15         //malheureusement le programme s'arrête ici
16
17         //2ème manière : c'est une méthode qui peut générer une exception que nous capturons
18         try {
19             Operations(0, 0);
20         } catch (java.lang.ArithmeticException ae) {
21             System.out.println("C'est une exception arithmétique ! " + ae.getMessage());
22         } catch (Exception e) {
23             System.out.println("C'est une exception ! " + e.getMessage());
24         }
25         finally {
26             System.out.println("on passe toujours par ici");
27         }
28         System.out.println("le programme peut continuer sans problème ...");
29     }
30 }
31
32 public static void Operations(double a, double b) throws ArithmeticException , Exception {
33     double c = 0;
34     if(b == 0) throw new ArithmeticException(" div par zéro ");
35     else if(a == 0 && b == 0) throw new Exception(" 0 / 0 donne aussi l'infini");
36 }
37

Console

<terminated> TestThrow [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (14 mai 2021 à 12:51:37)
C'est une exception arithmétique ! div par zéro
on passe toujours par ici
le programme peut continuer sans problème ...
```



Les Flux d'entrées sorties

Que fait ce programme ?

```
13 File f = new File("test.txt");
14 System.out.println("Chemin absolu du fichier : " + f.getAbsolutePath());
15 System.out.println("Nom du fichier : " + f.getName());
16 System.out.println("Est-ce qu'il existe ? " + f.exists());
17 System.out.println("Est-ce un répertoire ? " + f.isDirectory());
18 System.out.println("Est-ce un fichier ? " + f.isFile());
19 System.out.println("Affichage des lecteurs à la racine du PC : ");
20 for(File file : File.listRoots()) {
21     System.out.println(file.getAbsolutePath());
22     try {
23         int i = 1;
24         for(File nom : file.listFiles()){
25             System.out.print("\t\t\t" + ((nom.isDirectory()) ? nom.getName()+"-/" : nom.getName()));
26             if((i%4) == 0){ // opérateur ternaire : condensé de if/else
27                 System.out.print("\n");
28             }
29             i++;
30         }
31         System.out.println("\n");
32     }
33     catch (NullPointerException e) {
34         e.printStackTrace();
35     }
36 }
```

Console

<terminated> TestFile [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (12 mai 2021 à 10:27:22)

Chemin absolu du fichier : C:\Users\moham\JavaLdnr\Advanced\test.txt

Nom du fichier : test.txt

Est-ce qu'il existe ? true

Est-ce un répertoire ? false

Est-ce un fichier ? true

Affichage des lecteurs à la racine du PC :

C:\

\$AV_ASW-/

data-/

Forqan Group-/

pagefile.sys

ProgramData-/

System Volume Information-/

Windows-/

\$RECYCLE.BIN-/

Documents and Settings-/

hiberfil.sys

PerfLogs-/

Recovery-/

tools-/

Wondershare UniConverter-/

\$SysReset-/

DumpStack.log.tmp

Intel-/

Program Files-/

Recovery.txt

Users-/

\$WinREAgent-/

OneDriveTemp-/

Program Files (x86)-/

swapfile.sys

wamp64-/

eSupport-/

D:\

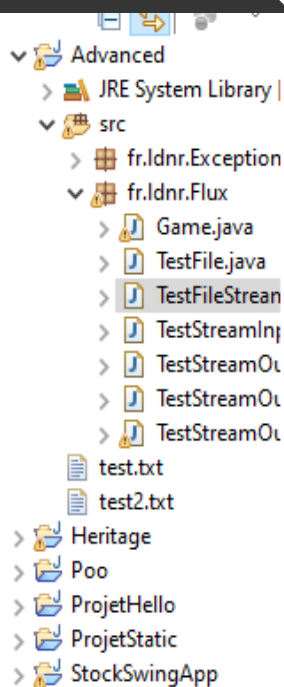
[java.lang.NullPointerException](#)

at fr.ldnr.Flux.TestFile.main(TestFile.java:27)

Les Flux d'entrées sorties

	Flux d'octets	Flux de caractères
Flux d'entrée	InputStream	Reader
Flux de sortie	OutputStream	Writer

Les Flux d'entrées sorties



```
12 FileInputStream fis = null;
13 FileOutputStream fos = null;
14 try {
15     fis = new FileInputStream(new File("test.txt")); // l'objet fis va lire dans le fichier test.txt
16     fos = new FileOutputStream(new File("test2.txt")); // l'objet fos va écrire dans le fichier test2.txt
17     byte[] buf = new byte[8]; //buffer contenant 8 octets/bytes d'informations lues à chaque tour de boucle
18
19     while ((fis.read(buf)) >= 0) { //tant qu'on peut lire le fichier / -1 fait sortir
20         // On écrit dans le fichier de destination
21         fos.write(buf);
22         // On affiche sur la console ce qu'on a lu
23         for (byte octet : buf) {
24             if(octet != 0) System.out.print((char) octet);
25         }
26         //for (byte octet : buf) System.out.print("(" + octet + ")");
27
28         //on réinitialise le buffer sans quoi on risque d'afficher des données déjà lues
29         for (int i = 0 ; i<buf.length ; i++) buf[i] = 0;
30     }
31     System.out.println("Copie terminée !");
32
33     } catch (FileNotFoundException e) {
34         // Cette exception est levée si l'objet FileInputStream ne trouve aucun fichier
35         e.printStackTrace();
36
37     } catch (IOException e) {
38         // Celle-ci se produit lors d'une erreur d'écriture ou de lecture
39         e.printStackTrace();
40     } finally {
41         // On ferme nos flux de données dans un bloc finally pour s'assurer
42         // que ces instructions seront exécutées dans tous les cas même si une exception est levée !
43         try {
44             if (fis != null) fis.close();
45         } catch (IOException e) {
46             e.printStackTrace();
47         }
48     }
```

Console

```
<terminated> TestFileStream [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (12 m
voici une ligne de test
et puis une encore
et encore une nouvelle
Copie terminée !
```

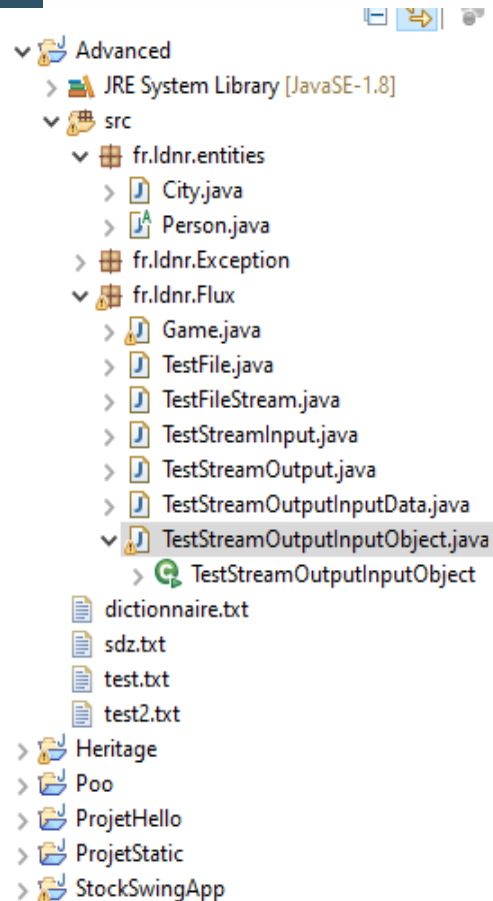
test.txt

```
1 voici une ligne de test
2 et puis une encore
3 et encore une nouvelle
4
```

test2.txt

```
1 voici une ligne de test
2 et puis une encore
3 et encore une nouvelle
4
```

Les Flux d'entrées sorties



```
15 public static void main(String[] args) {
16     ObjectInputStream ois;
17     ObjectOutputStream oos;
18     try {
19         oos = new ObjectOutputStream( new BufferedOutputStream( new FileOutputStream( new File("stars.txt"))));
20
21         //Nous allons écrire chaque objet Game dans le fichier
22         oos.writeObject(new Star("Potter", "Harry", "Poudlard"));
23         oos.writeObject(new Star("Andersen", "Neo", "Matrix"));
24         oos.writeObject(new Star("De Funes", "Louis", "Vadrouille"));
25         oos.close();
26
27         //On récupère maintenant les données !
28         ois = new ObjectInputStream( new BufferedInputStream( new FileInputStream( new File("stars.txt"))));
29
30         try {
31             System.out.println("Affichage des célébrités :");
32             System.out.println(((Star)ois.readObject()).toString());
33             System.out.println(((Star)ois.readObject()).toString());
34             System.out.println(((Star)ois.readObject()).toString());
35         } catch (ClassNotFoundException e) {
36             e.printStackTrace();
37         }
38         ois.close();
39     } catch (FileNotFoundException e) {
40         e.printStackTrace();
41     } catch (IOException e) {
42         e.printStackTrace();
43     }
44 }
45
46 private static class Star implements Serializable { //définition d'une classe interne forcément statique
47     private String lastName;
48     private String firstName;
49     private String celebrity;
```

Dans ce dernier exemple, l'intérêt ici est d'écrire des objets dans des fichiers, pour ce faire, ils doivent être :

Il existe plusieurs solutions pour optimiser la lecture et l'écriture d'un fichier selon la taille, à tester !

Console

```
<terminated> TestStreamOutputInputObject [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (12 mai 2021 à 12:17:04)
Affichage des célébrités :
Star [lastName=Potter, firstName=Harry, celebrity=Poudlard]
Star [lastName=Andersen, firstName=Neo, celebrity=Matrix]
Star [lastName=De Funes, firstName=Louis, celebrity=Vadrouille]
```

```

public class TestBufferedWriter {
    public static void main(String args[]) {
        try {
            int nombre = 12345;
            BufferedWriter fichier = new BufferedWriter(new FileWriter("source.txt"));

            fichier.write("bonjour le groupe Java SE LDNR 2021");
            fichier.newLine();
            fichier.write("Nous sommes le " + new Date());
            fichier.write("\nle pwd est " + nombre);

            fichier.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

public class TestBufferedReader {
    public static void main(String args[]) {
        try {
            String ligne ;
            BufferedReader fichier = new BufferedReader(new FileReader("source.txt"));
            //fileReader lit des caractères qui sont stockés dans un buffer ou tampon grâce au filtre BufferedReader
            //delors on peut appeler des méthodes qui lisent des lignes comme ici
            while ((ligne = fichier.readLine()) != null) {
                System.out.println(ligne);
            }

            fichier.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

public class TestFileWriter {
    public static void main(String[] args) {
        try {
            FileWriter myWriter = new FileWriter("print3.txt");
            myWriter.write("ça en fait des manières de travailler sur des fichiers en Java");
            myWriter.close();
        }
        catch (IOException e) {
            System.out.println("PROBLEM !");
            e.printStackTrace();
        }
    }
}

```

C'est quoi un Thread ?



C'est quoi un processus ?

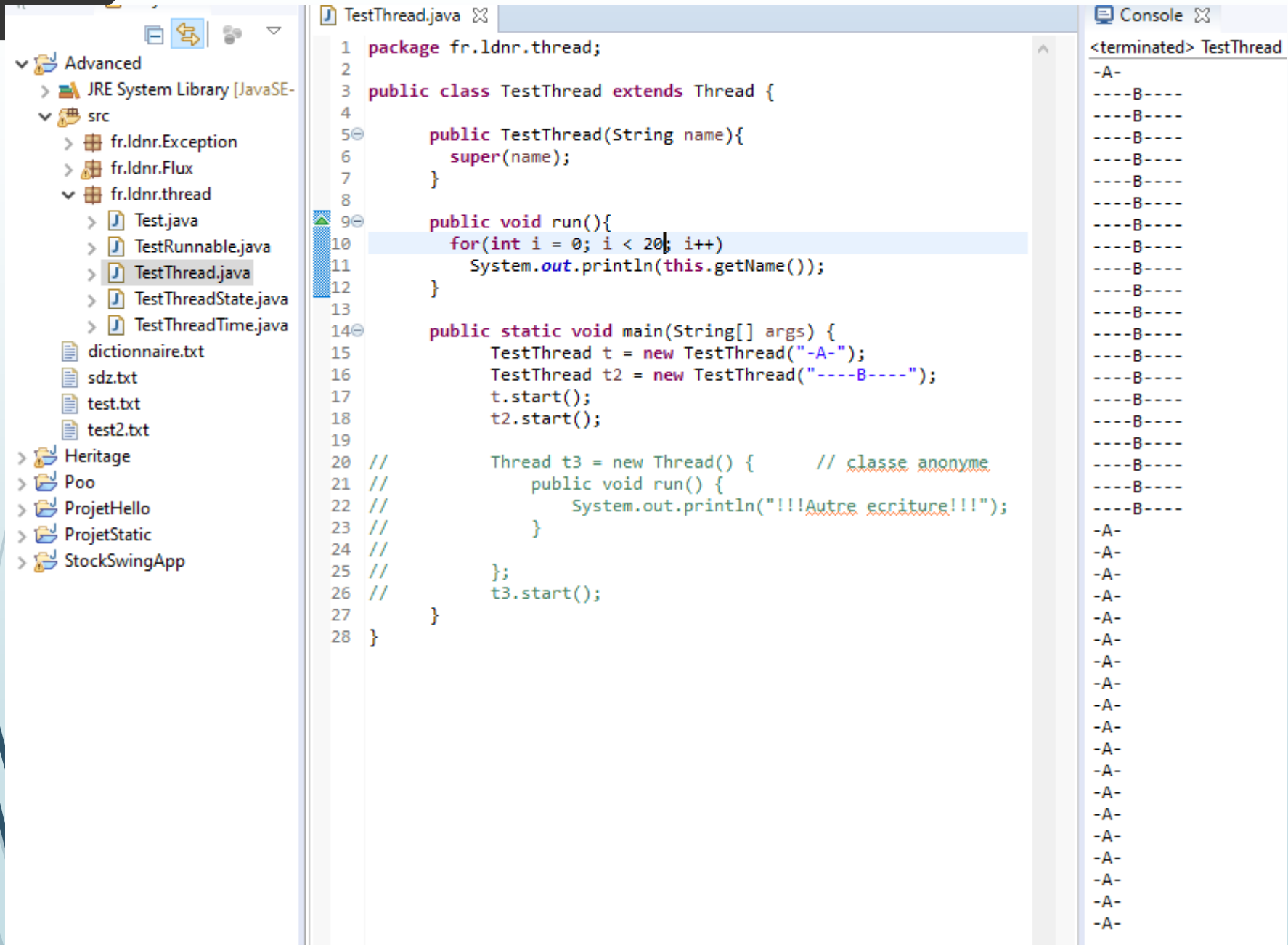
Un processus est un programme en exécution

```
Test.java
1 package fr.ldnr.thread;
2
3 public class Test {
4
5     public static void main(String[] args) {
6
7         System.out.println("Le nom du thread principal est " + Thread.currentThread().getName());
8     }
9 }
10
11 }
12 }
```

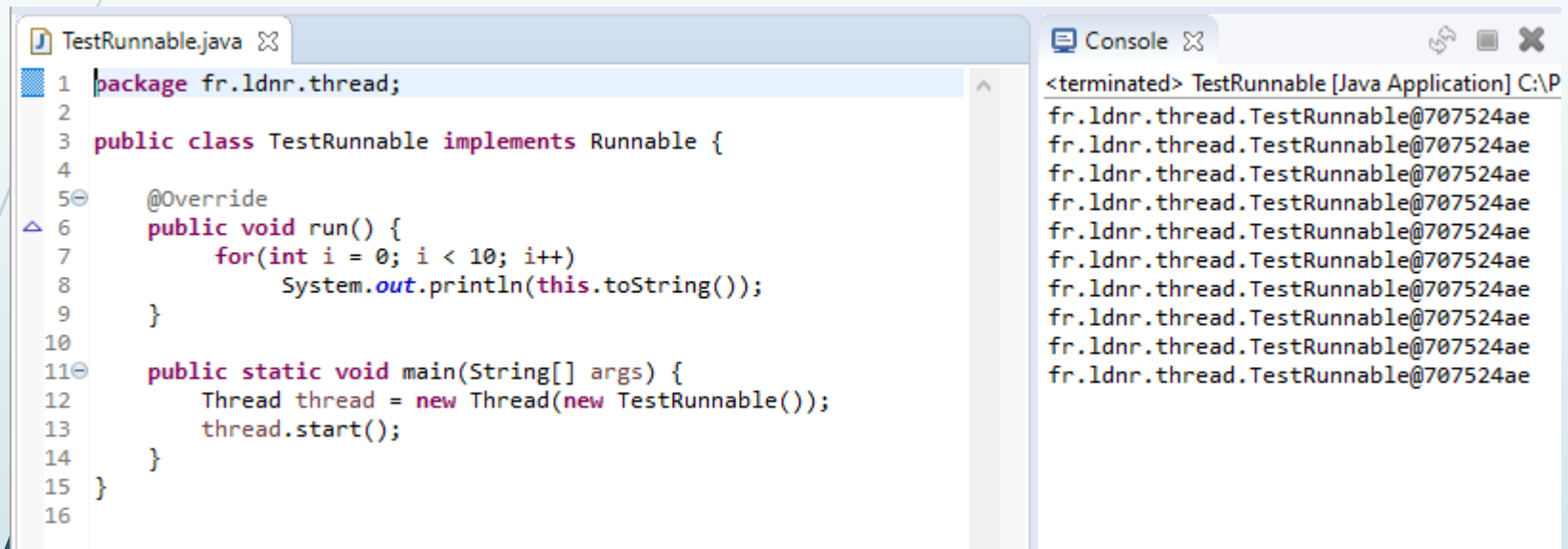
Nom	Statut	11% Processeur	62% Mémoire	3% Disque	0% Réseau
Applications (6)					
eclipse		4,6%	1 066,0 Mo	0 Mo/s	0 Mbits/s
JavaLdnr - Advanced/src/fr/Ld...					
Explorateur Windows		1,7%	46,5 Mo	0 Mo/s	0 Mbits/s
Firefox (6)		1,1%	1 087,4 Mo	0,1 Mo/s	0,1 Mbits/s
Gestionnaire des tâches		0%	27,5 Mo	0 Mo/s	0 Mbits/s
LibreOffice (32 bits) (2)		0%	76,4 Mo	0 Mo/s	0 Mbits/s

```
Console
<terminated> Test [Java Application] C:\Program
Le nom du thread principal est main
```

Première manière d'ajouter un thread



Deuxième manière d'ajouter un thread



The screenshot displays an IDE with two panels. The left panel shows the source code for `TestRunnable.java`, and the right panel shows the console output.

```
TestRunnable.java
1 package fr.ldnr.thread;
2
3 public class TestRunnable implements Runnable {
4
5     @Override
6     public void run() {
7         for(int i = 0; i < 10; i++)
8             System.out.println(this.toString());
9     }
10
11     public static void main(String[] args) {
12         Thread thread = new Thread(new TestRunnable());
13         thread.start();
14     }
15 }
16
```

The console output shows the execution of the program, displaying the memory address of the `TestRunnable` object ten times, indicating that the thread successfully executed the `run()` method ten times.

```
Console
<terminated> TestRunnable [Java Application] C:\P
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
fr.ldnr.thread.TestRunnable@707524ae
```

Nous avons un petit problème ici, lequel ?

The screenshot displays an IDE with three Java files and a console window.

BankAccount.java

```
1 package fr.lndr.threadJob;
2 // Ressources compte bancaire
3 public class BankAccount {
4     private int balance = 20; //Solde
5
6     public int getBalance(){
7         if(this.balance < 0)
8             System.out.println("Vous êtes à découvert !");
9         return this.balance;
10    }
11
12    public void retrait(int retrait){
13        balance = balance - retrait;
14        System.out.println("Solde compte = " + balance);
15    }
16 }
```

Test.java

```
1 package fr.lndr.threadJob;
2
3 public class Test {
4     public static void main(String[] args) {
5         BankAccount cb = new BankAccount();
6         //CompteEnBanque cb2 = new CompteEnBanque();
7
8         Thread pedro = new Thread(new UserBankAccount(cb, "pedro"));
9         Thread sancho = new Thread(new UserBankAccount(cb, "sancho"));
10        pedro.start();
11        sancho.start();
12    }
13 }
```

UserBankAccount.java

```
1 package fr.lndr.threadJob;
2
3 //utilisateur de compte bancaire
4 public class UserBankAccount implements Runnable {
5     private BankAccount cb;
6     private String name;
7
8     public UserBankAccount(BankAccount cb, String name){
9         this.cb = cb;
10        this.name = name;
11    }
12
13    public void run() {
14        for(int i = 0; i < 15; i++){
15            if(cb.getBalance() > 0){
16                cb.retrait(2);
17                System.out.println("Retrait effectué par " + this.name);
18            }
19        }
20    }
21 }
```

Console

```
<terminated> Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (15 mai 2022)
Solde compte = 16
Solde compte = 18
Retrait effectué par pedro
Retrait effectué par sancho
Solde compte = 14
Solde compte = 12
Retrait effectué par sancho
Retrait effectué par pedro
Solde compte = 10
Retrait effectué par sancho
Solde compte = 8
Retrait effectué par pedro
Solde compte = 4
Retrait effectué par pedro
Solde compte = 2
Retrait effectué par pedro
Solde compte = 0
Retrait effectué par pedro
Solde compte = 6
Retrait effectué par sancho
```


Lorsque 2 threads ou plus ont un accès à une ressource commune, il faut mettre un verrou sur cette ressource pour empêcher l'ordonnanceur du SE (responsable de dispatcher l'utilisation du processeur) de retirer la main au 1^{er} servi jusqu'à ce qu'il ai fini !



BankAccount.java

```
1 package fr.lidnr.threadJob;
2 // Ressources compte bancaire
3 public class BankAccount {
4     private int balance = 20; //Solde
5
6     public int getBalance(){
7         if(this.balance < 0)
8             System.out.println("Vous êtes à découvert !");
9         return this.balance;
10    }
11
12    public synchronized void retrait(int retrait , String name){
13        balance = balance - retrait;
14        System.out.println("Solde compte = " + balance);
15        System.out.println("Retrait effectué par " + name);
16    }
17 }
```

UserBankAccount.java

```
1 package fr.lidnr.threadJob;
2 //utilisateur de compte bancaire
3 public class UserBankAccount implements Runnable {
4     private BankAccount cb;
5     private String name;
6
7
8     public UserBankAccount(BankAccount cb, String name){
9         this.cb = cb;
10        this.name = name;
11    }
12
13    public void run() {
14        for(int i = 0; i < 15; i++){
15            if(cb.getBalance() > 0){
16                cb.retrait(2,this.name);
17            }
18        }
19    }
20 }
```

Test.java

```
1 package fr.lidnr.threadJob;
2
3 public class Test {
4     public static void main(String[] args) {
5         BankAccount cb = new BankAccount();
6         //CompteEnBanque cb2 = new CompteEnBanque();
7
8         Thread pedro = new Thread(new UserBankAccount(cb, "pedro"));
9         Thread sancho = new Thread(new UserBankAccount(cb, "sancho"));
10        pedro.start();
11        sancho.start();
12    }
13 }
```

Console

<terminated> Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (15 mai 202

```
Solde compte = 18
Retrait effectué par pedro
Solde compte = 16
Retrait effectué par pedro
Solde compte = 14
Retrait effectué par pedro
Solde compte = 12
Retrait effectué par pedro
Solde compte = 10
Retrait effectué par pedro
Solde compte = 8
Retrait effectué par pedro
Solde compte = 6
Retrait effectué par sancho
Solde compte = 4
Retrait effectué par sancho
Solde compte = 2
Retrait effectué par sancho
Solde compte = 0
Retrait effectué par sancho
Solde compte = -2
Retrait effectué par pedro[
Vous êtes à découvert !
Vous êtes à découvert !
Vous êtes à découvert !
Vous êtes à découvert !
Vous êtes à découvert !
```


Expressions Lambdas

Derrière cette notion se cache en réalité la volonté de réduire le code ou nous faciliter les choses dans certaine circonstance telle que **l'implémentation d'une interface ayant qu'une seule méthode, on parle d'interface fonctionnelle**

```
public static void main( String [] args ) {  
    List<String> collection = new ArrayList<>();  
    collection.add( "A380" );  
    collection.add( "a320" );  
    collection.add( "a350" );  
    collection.add( "a400M" );  
    collection.add( "A330" );  
  
    collection.sort(null);           //A330  A380  a320  a350  a400M  
  
    collection.sort( new Comparator<String>() {           //interface anonyme  
        @Override public int compare( String str1, String str2 ) {  
            return str1.compareToIgnoreCase( str2 );  
        }  
    });  
  
    for ( String airbus : collection ) System.out.println( airbus );  
  
    collection.sort( (m1, m2) -> m1.compareToIgnoreCase( m2 ) );  
}
```

Expressions Régulières

Une expression régulière est un **ensemble de caractères particuliers appelé format (pattern)** permettant de décrire un ensemble de chaînes de caractères à reconnaître. Une fois définie, permet : Matching, Substitution, Extraction de donnée

```
import java.util.Scanner;
import java.util.regex.Pattern;

public class RegularExpressions {
    public static void main(String[] args) {
        System.out.println(isValidEmail("mohamed.el-babili@fms-ea.com")); //true
        System.out.println(isValidEmail("mohamed.@fms.fr")); //true
        System.out.println(isValidEmail("@.@fms.c")); //true

        System.out.println(isReallyValidEmail("@.@fms.c")); //false

        TestonScan();
    }

    public static boolean isValidEmail( String email ) {
        String regExp = "^.+@.+\\..+$"; //expressions régulières
        //^ : doit commencer par le contenu de la chaîne, rien avant
        //. : tous les caractères sont permis avant @ sauf \n
        //+ : répétition du . au moins une fois -> mohamed.el-babili
        //@ : @ -> @
        //.+ : partie gauche nom de domaine ici -> fms-ea
        //\\. : permet d'activer le point -> .
        //.+ : partie droite nom de domaine -> com
        //$ : l'ER est fini ici, rien après
        return email.matches( regExp );
    }

    public static boolean isReallyValidEmail(String email) {
        String regExp = "^([A-Za-z0-9._-]+@[A-Za-z0-9._-]+\\.[a-z][a-z]+$";
        //^
        //[liste de tout ce qui est admis [lettres min/maj, les chiffres, _, -, .]+ -> mohamed.el-babili
        //@ -> @
        //[lettres min/maj, les chiffres, _, -, .]+ -> fms-ea
        //\\. -> .
        //[a-z][a-z]+ : min de 2 lettres -> fr
        //$

        return email.matches(regExp);
    }
}
```

Expressions Régulières

```
private static void TestonScan() {
    Scanner scan;
    String str = "- + 10 coucou 15 moi non plus 8";
    scan = new Scanner(str);
    scan.useDelimiter("[^\\d]+");
    while (scan.hasNext()) {
        String token = scan.next();
        System.out.println(token);
    }
    scan.close();    //fermeture du flux

    scan = new Scanner("Neo Anderson/Luke/Vous");
    scan.useDelimiter(Pattern.compile("/"));    //initialisation du pattern de delimitation
    System.out.println(" le délimiteur utilisé ici est : " + scan.delimiter());

    while(scan.hasNext()){ //affiche de tous les tokens ou occurrences séparées par le délimiteur
        System.out.println(scan.next());
    }
    scan.close();    //fermeture du flux
}
```

Stream : Flux

```
4 public class TestStream {
5
6     public static void main(String[] args) {
7         List<String> airbus = Arrays.asList("A320-", "A350-", "A400m-", "a380-", "B330-");
8
9         airbus.stream().sorted();
10        airbus.forEach(System.out::print); //A320 A350 A400m A380 A330
11        System.out.println();
12
13        airbus.stream() // (1) retourne un flux clone de la source
14            .filter(s -> s.toUpperCase().startsWith("A")) // (2) retourne un clone de (1) contenant que des chaines commençant par A (après mise en maj)
15            .map(String::toUpperCase) // (3) retourne un clone de (2) contenant que des majuscules
16            .sorted() // (4) retourne un clone de (3) par ordre croissant
17            .forEach(System.out::print); // (5) affiche chaque élément du stream
18        System.out.println();
19
20        airbus.forEach(System.out::print);
21    }
22 }
23
```

Problems @ Javadoc Declaration Console ×

<terminated> TestStream [Java Application] C:\Users\EI-BabiliM\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v20211012-1059\jre\bin\javaw.exe (4 avr. 2022, 15:32:16 – 15:32:16)

A320-A350-A400m-a380-B330-

A320-A350-A380-A400M-

A320-A350-A400m-a380-B330-

Predicate & Stream & lambda

@FunctionalInterface
public interface **Predicate**<T>
{ **boolean test**(T t); }

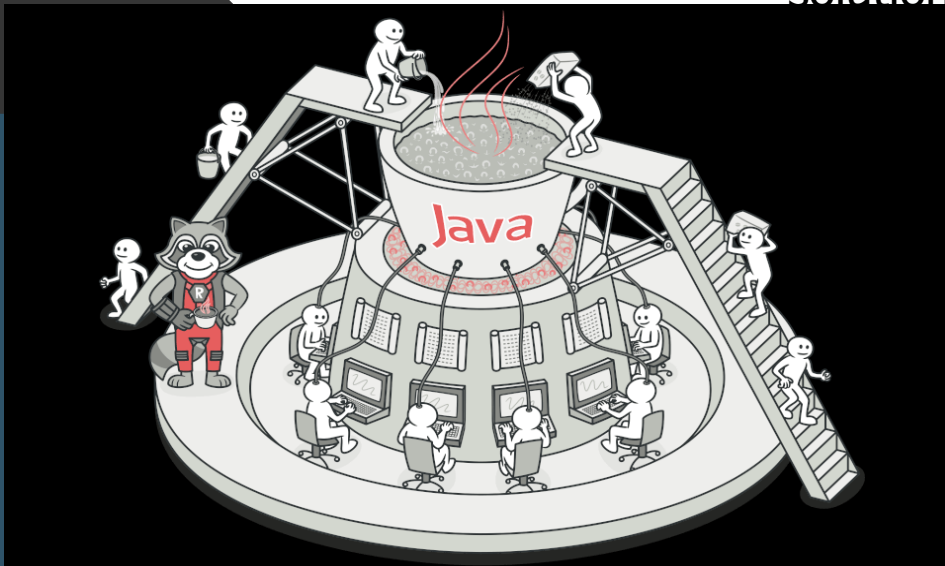
Parmi les nouveautés de java 8, nous avons également la notion de *Predicate* qui est une functional interface, représentant un opérateur qui accepte un paramètre d'entrée et renvoie une valeur boolean.

```
1 public class Employee {  
2     private String name;  
3     private double salary;  
4     private String gender; // "M", "F"  
5  
6     public Employee(String name, double salary, String gender)  
7     {  
8         this.name = name;  
9         this.salary = salary;  
10        this.gender = gender;  
11    }  
12  
13    public String getName() {  
14        return name;  
15    }  
16  
17    public double getSalary() {  
18        return salary;  
19    }  
20  
21    public String getGender() {  
22        return gender;  
23    }  
24  
25    public boolean isFemale() {  
26        return "F".equals(this.getGender());  
27    }
```

```
1 import java.util.Arrays;  
2 import java.util.List;  
3 import java.util.function.Predicate;  
4  
5 public class TestEmployee {  
6     public static void main(String[] args) {  
7         Employee gargamel = new Employee("Gargamel", 200, "M");  
8         Employee leia = new Employee("Leia", 2000, "F");  
9         Employee tintin = new Employee("Tintin", 3700, "M");  
10        Employee minnie = new Employee("Minnie", 5000, "F");  
11        Employee cleopatre = new Employee("Cleopatre", 7000, "F");  
12  
13        List<Employee> employees = Arrays.asList(gargamel, leia, tintin, minnie, cleopatre);  
14  
15        Predicate<Employee> predicate = e -> e.isFemale() && e.getSalary() > 2500;  
16  
17        employees  
18            .stream()  
19            .filter(predicate)  
20            .forEach(e -> System.out.println(e.getName() + " : " + e.getSalary()));  
21    }  
22 }  
23 }
```

Les designs pattern

Def : c'est un modèle de conception qui permet de décrire un problème fréquemment rencontré et l'architecture de la solution de sorte qu'elle soit réutilisable.



Pourquoi les patterns ?

- Des « recettes d'expert » ayant fait leurs preuves
- Un vocabulaire commun pour les architectes logiciels
- Incontournable dans le monde de la P.O.O

Les Designs pattern exigent un niveau d'abstraction plus élevé à comprendre aussi il faut maîtriser la Poo avant tout

Classification Gof (gang of Four)
(Gamma, Helm, Johnson, Vlisside)

Création : DP montre comment les objets peuvent être créés, initialiser, configurer

Structure : comment les objets vont être connectés pour être indépendants des évolutions futures

Comportement : l'interaction entre les objets pour résoudre un problème donné

		Catégorie		
Portée	Classe	Création	Structure	Comportement
		Factory Method	Adapter	Interpreter
Objet				Template Method
		Abstract Factory	Adapter	Chain of Responsibility
		Builder	Bridge	Command
		Prototype	Composite	Iterator
		Singleton	Decorator	Mediator
			Facade	Memento
			Flyweight	Observer
			Proxy	State
				Strategy
				Visitor

Next

- Java SE 8
- Algo
- Git
- La POO avec Java
- Uml
- Programmation avancé
- **Base de données**