Java (youtube → FormationVideo : https://www.youtube.com/@formation-video/videos)

Table des matières

Index

Sout = system.out.println();

Psvm(public static void main (String[] args))

Introduction- #1 et #2

Class *nom* {

Public static void main (String[] args {

System.out.println(“hello world !);

}

}

- Accolades très importantes
- Point virgules à la fin pour terminer une instruction
- // commentaires → non execute → une ligne
- Plusieurs lignes : /*
  Ffff
  Fffff
  Ff
  */

Type de données - #3

Type primitif	**Type « objet » associé**	**Place en mémoire**	**Fourchette de valeurs (min & max)**
boolean	Boolean	1 octet	True (1) | false (0)
byte	Byte	1 octet	-128 à 127
char	Character	2 octets	Unicode (65536 carac. Dispo.)
short	Short	2 octets	-32768 à 32767
int	Integer	4 octets	- 2 147 483 à 2 147 483 647
long	Long	8 octets	-2^{63} à 2^{63} -1
float	Float	4 octets	$1.4x10^{-45}$ à $3.4x10^{38}$
double	Double	8 octets	$4.9x10^{-324}$ à $1.7x10^{308}$

Base numérique :

1. Décimale (10 → 1 à 9)
2. Binaire (2 → 0 à 1)
3. Octale (8 → 0 à 7)
4. Hexadécimale (16 → 0 à F)

Caractères spéciaux :

\n : nouvelle ligne

\t : tabulation

\r : retour chariot

\b : retour en arrière

\f : nouvelle page

\’ \ ‘’

Les caractères spéciaux doivent être mis dans le texte

Variables et constantes- #4

On déclare une variable

Attention :

- Pas de caractères spéciaux
- Commence par une lettre ou underscore
- Camel case : uneVariableExercice
- Pascal case : UneVariableExercice
- Snake case : une_variable

Pour les variables utiliser camel case

Exemple :

Int ageUtilisateur = 25 ;

Créer une variable constante :

final int NUMBER = 26 ;

en général en maj les constantes

Opérateurs- #5

= affectation

+ concaténation (exemple dans le system.out.println(« tu as » + age + « ans »)

+ somme

- soustraction

* multiplication

/ division

% modulo

() utiliser parenthèses pour changer la priorité exemple : (4+5) * 2

Comparaison

Egal à ==

Égal et de même type ===

Différent de !=

Supérieur à >

Supérieur et égal à >=

Inférieur <

Inférieur et égal à <=

! négation de qqch

&& ET

|| OU

Incrémentation et décrémentation

Ajouter une valeur à une autre ou soustraire

int pieceOr =0 ;

pieceOR += 10 ;

+=

-=

*=

/=

%=

A = A +X → A += X

A++ post incrémention ++A pré incrémentation

A -- Post décrémentation --A pré décrémentation

Conditions - #6

If (conditions)

Else if

Else

Int option = 1

Switch (option)

{

Case 1 : // if (option==1)

break ;

case 2 :

etc

default : // option != “X” && option != “n”

sout(« blabla »

break ;

Boucles - #7

Int i =0;

While (i !=10)

Sout(i) → boucle infini

i++ → il s'arrêtera à 9

break interompre une boucle

continue SAUTER et revenir au début

do while :

Sout(i) → boucle infini

i++ → il s'arrêtera à 9

While (i !=10 && || etc)

For :

for (int i =0; i != 10 *condition d'arret*;i++){

sout (i);

}

Construction d'une classe - #8

Représenter une idée / une entité / un concept

Portée : public ou private (non accessible)

1ère classe

Public class *nom* {

Public static void main (String[] args){

Cat miaou = new Cat ();

}

}

2 ème classe

Si on crée une nouvelle classe :

Public class *nom*{

Public Cat()

{

System.out.println(« je suis un chat : » + this) ;

}

}

Attributs - #9

Attribut = variable d'une classe

1ère classe

Public class *nom* {

Public static void main (String[] args){

Cat miaou = new Cat (« Felix », 1);

}

}

2 ème classe

Si on crée une nouvelle classe :

Public class *nom*{

Public Cat(String nom, int age)

{

this.name = name;

this.age = age;

sout (this.name + " – " + this.age);

}

Private String name;

Private int age;

}

Méthodes - #10

```
Public class nom {
Public static void main (String[] args){
Int age = 0;
System.out.println(age);
Age = 15;
System.out.println(age);

// Nous créons une méthode lorsqu'un code se répète

prog() ;  // = System.out.println(« Bonjour ») ;
          //=  System.out.println(« Comment allez-vous »)


int number = getNumberTwo ;
sout (number) ; // =2

}
Public static void prog()
{
System.out.println(« Bonjour ») ;
System.out.println(« Comment allez-vous »)
}
Public static int getNumberTwo(){
Return 2;
}

}
```

Auto documente le code si un bon titre

On peut mettre plusieurs return MAIS il n'exécute qu'un seul et sort de la méthode exemple avec des if // très important

Surcharge de méthode :

On peut mettre le même nom de la méthode mais avec différentes instructions

Exemple :

1. Public static int sum (int a, int b){
   Return a+b;
   }
2. Public static float sum (float a, float b){
   Return a + b;
   }

Encapsulation données - #11

Propre au langage orienté objet : chaque classe va posséder ces propres attributs

Accesseurs : (méthode déjà crée : private int ExpirationYear ;

Getters : accès à un attribut (lecture)

- Public int getExpirationYear () {return this.ExpirationYear;}

Setters : modification d'un attribut

Public void setExpirationYeat (int expiration){this.Expiration = expiration ;}

Public class *nom* {

Public static void main (String[] args){

softwareRegistration sr = new softwareRegistration (2022);

sr.setExpiration = 2050;

sout(sr.getExpirationYear());

}

}

Tableau - #12

Le système d'exploitation gère le stockage sur la mémoire et non le langage

Le système va choisir l'emplacement mémoire pour stocker contigu (à la suite)

Int [] tab = new int [3] un tableau de trois

Int [] tab = new int[] {1,2,3} un tableau initialisé

Int [] tab = {1,2,3}

1	2	3
Indice 0	Indice 1	Indice 2

Sout (tab) ; → retourne l'adresse du tableau dans la mémoire

Sout (tab[0]) ;

Sout (tab[1]) ;

Sout (tab[2]) ;

Sout (tab[3]) ; → message erreur car nous dépassons les limites du tableau

Parcourir un tableau :

For (int i = 0 ; i <tab.length;i++;

Sout(tab[i]);

Equivalent : → for (int number : tab)

Sout(number)

Tableau 2 dimensions

Int [][] tab = new int [3][4] ;

Int [][] tab = {{1,2,3}, {4,5,6,7}; → 3 lignes et 4 colonnes

For (int i = 0 ; i < tab.length; i++)

For (int j = 0; j <tab[i].length;j++)

Sout(tab[i][j];

For (int [] i : tab)

For (int number : i)

Sout(number);

Il exsite des méthodes utilitaire pour les tableau par exemple

Import java.utill.Arrays ;

Arrays.Fill(tab,15) → compléter un tableau avec que des 15

Les chaines de caractères - #13 et #14

String :

- Représente une chaine de caractère
- Commence avec une maj
- Immuable (contenu non modifiable) à chaque qu'on modifie une chaine de caractère on en crée une autre en réalité
- → concat () plus rapide que « + »
- → length()
- →toUpperCase(), toLowerCase() // mettre tout en maj ou miniscule (Attention on créée même si on modifie)
- → trim() // retirer les espaces
- → replace(« a », « o ») // remplace les caractères exemple tous les « a » par des « o »
- → charAt (<index>)
  - Exemple : String test = bobo
  - Sout(test.charAt(0) ; //= « b »
- → substring(<index>, <nb>) //extraire un bout de chaine de caractère
- → equals (<str>), compareTo (<str>) → comparer chaine de caractère

String *nom* = « Hello » ou String s = new String ("Hello")

```
String s = "bon";
    String s2 = "jour";
    String s4 = " tout le monde";

    String s3 = s.concat(s2).concat(s4);
    System.out.println(s3);
}
```

Sout(s + « » + s2 + « » + s4; //= Bonjour tout le monde ! // concaténé →**EVITER D'UTILISER LE +**

s.length()) ; → nombre de caractère

Vérifier chaine de caractère

/ ! \ → si on compare (s == s2) // ici on compare la donnée où elle pointe

Pour comparer :

String s = "bon";

String s2 ="jour"

s.equals(s2))

→ retour : True / false

s.compareTo(s2) → compare par rapport à la table « Ascii »

exemple : String str = « A » ; // 65

String str2 = « a » // 97

Si on compare les 2 on a : -32 car 65-97

Import java.util.StringTokenizer

String s = « news / titre-de-la-news/14 » ;

StringTokenizer st = new StringTokenizer(s, "/"); → séparer, delimiter

While(st.hasMoreTokens())

Sout(st.nextToken(); →

- News
- Tire-de-la-news
- 14

Classes : String Builder / String Buffer → muable // contenu modifiable

- → length()
- → capacity()
- → append(« mot ») → assembler || insert(<index>, « mot ») → assembler les mots dans ordre qu'on veut
-

StringBuilder sBuilder = new StringBuilder("bonjour tout le monde ");

Sout(sBuilder.length()); → longueur → 21 caractères

Sout(sBuilder.capacity()); → capacité → 37 caractères

sBuilder.append("bonjour");

sout(s.Builder); → renvoie : bonjour

sBuilder.append (" tout le monde » ;

sout(sBuilder) ; → renvoie : bonjour tout le monde

sBuilder.append("tout le monde");

sout(s.Builder); → renvoie : tout le monde

sBuilder.insert (0, « bonjour ») ;

sout(sBuilder) ; → renvoie : bonjour tout le monde

Si on utilise StringBuffer = on a le même résultat

Différence :

Monothreadé : 1 seul thread = exécution dans l'ordre exemple douane (frontière) = StringBuilder(asychrone)

Mutli-threadé : plusieurs threads = plusieurs routes donc plusieurs voitures passent = StringBuffer(synchronisé, thread-safe)

Un thread c'est quoi ?

- *En programmation informatique, un thread, ou fil d'exécution, consiste en des informations substituables associées à une utilisation unique d'un programme capable de gérer plusieurs utilisateurs simultanés.*
- *Un thread en Java est un objet, qui doit étendre la classe Thread . Comme Java ne supporte pas l'héritage multiple, il n'est pas toujours possible d'étendre cette classe. Java nous propose donc aussi un mécanisme d'enveloppe (wrapper),*

Paquets - #15

Un package : un répertoire qui va contenir une ou plusieurs classes

Importer toutes les classes sur java :

Import *nom*.* // en général le nom en tout miniscule

Dans les autres classes on doit mettre : package *nom* ;

Classe main :

Import nom.*

Public classe App

Public static void main……

Player P = new player()

Item it = new Item() ;

Lorsque on excute on a les 2 textes item instancie et player instancie

Classe Player :

Package *nom*

Public class Player

Public player()

Sout(« player instancie ! »)

Classe Item

Package *nom*

Public class Player

Public Item()

Sout(« Item instancie ! »)

Portée :

Public : on peut l'utiliser depuis une autre classe

Rien : public que pour le package

Private : pas possible de l'utiliser depuis d'autres classe

Lecture au clavier - #16

- BufferedReader br ; (=lecture de donnée, thread safe, synchrone, chaine de caractère(String))
    - Tampon de 8192 caractères
- Scanner sc ; (asychrone, lecture de donnée + parsing)
    - Tampon de 1024 caractère (une fois atteint un autre tampon est créée)
- Read () : lire un caractère
- Readline() : lire une chaine
- Skip(N) : ignore N caractère

BufferedReader

Psvm (public static void main) throws Exception (=elle peut lever une exception){

InputStreamReader isr = new InputStreamReader(System.in) ;

bufferedReader br = new BufferedReader(isr) ; //importer = import java.io.BufferedReader
//import java.io.InputStreamReader

Sout(« comment t'appelles-tu ? ») ;

String name = br.readline(name) ;

Sout (name) ; → message erreur pour cela, il suffit d'ajouter à coté de « pvsm » throws Exception

}

Dans quel cas utilisé bufferedReader ? = lire chaine de caractère sans vérifier un type particulier juste de la lecture

Scanner

- **nextLine()**
- **nextChar()**
- **nextByte()**
- **nextFloat()**
- **nextDouble()**
- **nextInt()**
- **nextBollean()**
    - **classe gourmande car à l'utilisation en plus de la lecture elle va verifier le type de donnée, si donnée pas bonne = une erreur**

```
Pvsm {

Scanner scan = new Scanner (System.in) ; //import java.util.Scanner

Sout(« accèder à quel niveau ? ») ;

Int game_level = sc.nextInt() ;

Sout(game_level) ;

}
```

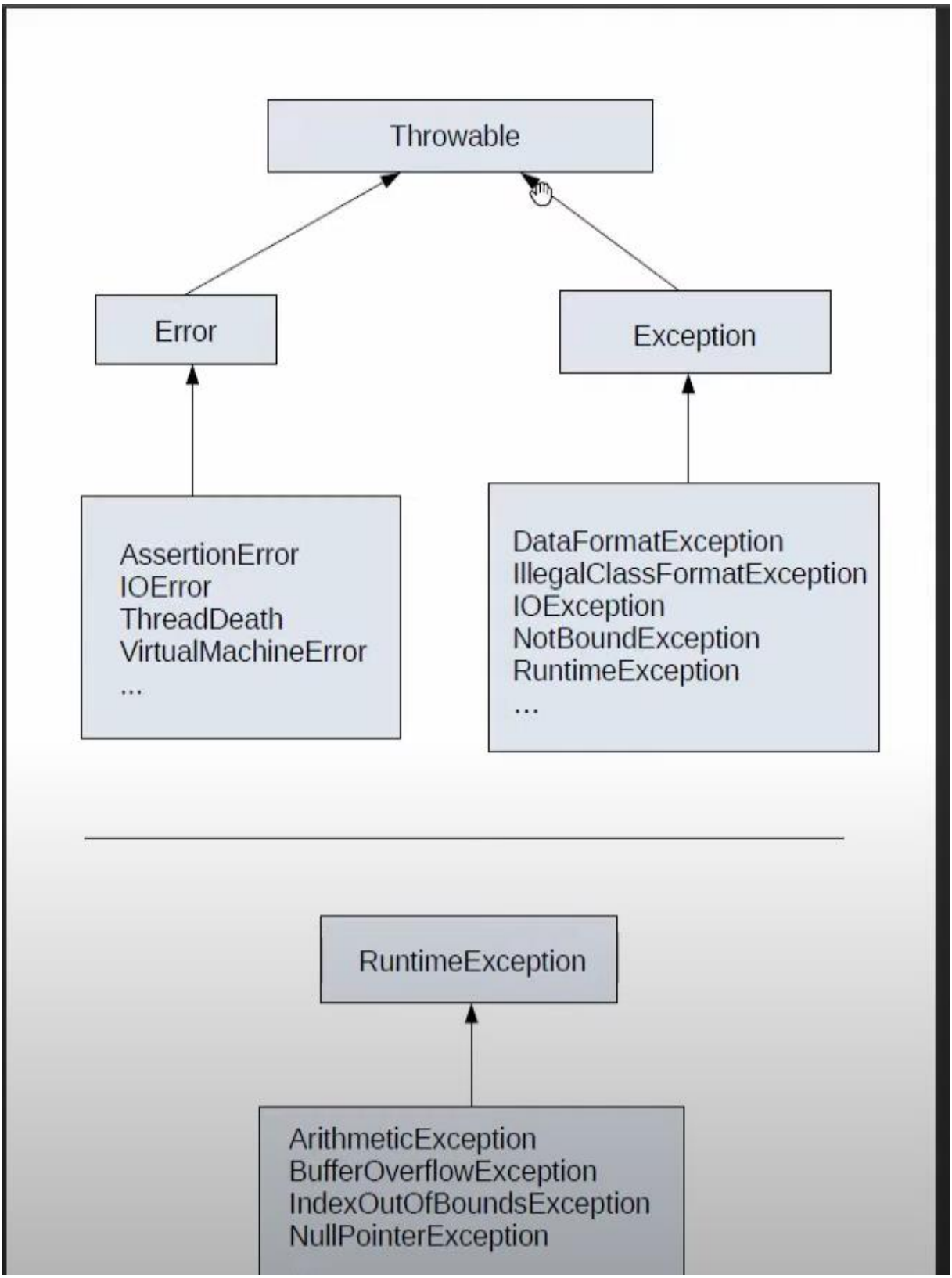
Exceptions - #17

```
Import java.util.*
Pvsm(String[] args) {
Try{
Scanner sc = new Scanner (System.in) ;
Int anneeNaissance = sc.nextInt() ;
System.out.println(anneeNaissance) ;
}
Catch(InputMismatchException nom) // va chercher cette exception est l'affiché
Sout(« Erreur : + e.getMessage() ;
{
Sout(« la date de naissance est incorrete. ») ;

}
Catch(OtherException e)

}
Finally {          //peut importe les exceptions, quoi qu'il arrive, il va lire finally, 1 seul bloc
}
```

- On peut cumuler les catch
- Un seul try
- Un seul finally


Throwable
Error
Exception
AssertionError
IOError
ThreadDeath
VirtualMachineError
...
DataFormatException
IllegalClassFormatException
IOException
NotBoundException
RuntimeException
...
RuntimeException
ArithmeticException
BufferOverflowException
IndexOutOfBoundsException
NullPointerException

Système de fichiers - #28

Lecture de donnée avec BufferedReader :

```
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.nio.file.Files;
import java.io.IOException;

public class Main {
  public static void main(String[] args) {
    Path f = Paths.get("info.txt"); // on part du principe que nous avons une
classe infos.txt dans le même répertoire que la classe actuelle

    try {
      BufferedReader bfr = Files.newBufferedReader(f);
      bfr.skip(5); //ignorer x caractères
      System.out.println(bfr.readLine());
      System.out.println(bfr.readline());

      bfr.close();
    } catch (IOException e) {
      System.out.println("Exception : " + e.getMessage());
    } catch (Exception e) {
      // handle other exceptions
    } finally {
      // code to be executed regardless of whether an exception is thrown or not
    }
  }
}
```

Écriture de donnée avec BufferedWriter:

```
import java.nio.file.Paths;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.nio.file.StandardOpenOption;
import java.nio.file.Files;
import java.io.IOException;
import java.nio.charset.Charset;

public class Main {
  public static void main(String[] args) {
    Path f = Paths.get("info.txt"); // Assuming we have a class info.txt in the
same directory as the current class
    Charset c = Charset.forName("UTF-8");
    StandardOpenOption mode = StandardOpenOption.TRUNCATE_EXISTING;
    // StandardOpenOption mode = StandardOpenOption.CREATE;
    // StandardOpenOption mode = StandardOpenOption.CREATE_NEW; // ERROR IF FILE
ALREADY EXISTS
    // StandardOpenOption mode = StandardOpenOption.APPEND; // Write to the end,
after the existing code without overwriting the rest
    // StandardOpenOption mode = StandardOpenOption.DELETE_ON_CLOSE;

    try {
      BufferedWriter bfw = Files.newBufferedWriter(f, c, mode);
      bfw.write(67); // we can also do: bfw.write("a"); bfw.write("Bonjour");
      bfw.close();

      bfw.write("Hello World", 2, 5); // display from index 2 and write 5
letters
      // same thing
      String s = "Hello World";
      bfw.write(s, 2, 5);
    } catch (IOException e) {
      System.out.println("Exception: " + e.getMessage());
    } catch (Exception e) {
      // handle other exceptions
    }
  }
}
```

/!\ Lorsqu'on met 2x fois ou plus « Bfw.write(abc) ; » et Bfw.write(123) ;

Sur la page infos.txt nous aurons : abc123 //tout collé pour cela il faut ajouter des retour à la ligne → bfw.newLine() ; entre chaque variable et il n y aura pas de ligne vide

Méthodes de manipulations

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.nio.file.StandardOpenOption;
import java.nio.file.Files;
import java.io.IOException;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Main {
  public static void main(String[] args) {
    Path f = Paths.get("info.txt"); // on part du principe que nous avons une classe infos.txt
    System.out.println(Files.exists(f)); // est-ce que le fichier existe, renvoie false ou true

    Path t = Paths.get("test.txt");
    Path directory = Paths.get("repertoire");

    System.out.println(Files.isDirectory(directory)); // est-ce que "nom" est un répertoire, renvoie true ou false
    System.out.println(Files.isRegularFile(directory)); // même chose pour les fichiers, renvoie false ici

    Path fileSource = Paths.get("newFile.truc"); // je veux déplacer ce fichier dans un répertoire (nom fichier source)
    Path machinDir = Paths.get("Machin"); // (repertoire dans lequel on veut le déplacer
    Path fileDestination  = machinDir.resolve("NewFile.newTruc"); // (nom fichier de destination)

    try {
      Files.createFile(t); // crée un nouveau fichier
      Files.createDirectory(directory); // crée un répertoire, pour créer un fichier dans le répertoire on met "nom/" dans le path

      Files.move(fileSource, fileDestination); // déplacer fichier dans répertoire et on ne perd pas le contenu
      Files.copy(fileSource, fileDestination); // copier fichier et renommer dans répertoire et on ne perd pas le contenu

      System.out.println(Files.size(fileSource)); // taille du fichier

      Files.delete(fileSource); //supprimer fichier, mais s'il n'existe pas erreur
      Files.deleteIfExists(fileSource); //supprimer fichier s'il existe

    } catch (IOException e) {
      System.out.println(e.getMessage());
    }
  }
}
```

Bonus :

BigDecimal

Random :

- ThreadLocalRandom.current().nextInt(101); // va jusqu'à 100 donc il fait -1
- double rounded = Math.round(num * 100.0 / 5.0) * 5.0 / 100.0;

Attribut :

Arguments :

Paramètre :