319- Concevoir et implémenter des applications

Rapport personnel

Date de création : 12.09.2024
Version 1 du 12.09.2024

Matias Serrano Grilo

ETAT DE FRIBOURG
STAAT FREIBURG

EMF – Fribourg / Freiburg
Ecole des Métiers / Berufsfachschule
Technique / Technik

Module du 29.08.2024
au 16.01.2025

Table des matières

1 Les bases de Java 5

1.1 La méthode main(), utilité et sa déclaration exacte 5

1.2 Les 8 types de base de Java (liste, limites, … reprendre de « Types de données ») 5

1.3 Déclaration d'une variable et affectation d'une variable 5

1.4 Déclaration d'une constante 6

1.5 Les commentaires (une ligne ou plusieurs lignes) 6

1.6 L'écriture sur la console avec la méthode sout (println et print) 6

1.7 Les opérateurs en Java 6

1.7.1 Les opérateurs de calcul (+ - * / %) 6
1.7.2 Les opérateurs d'assignation (= += -= *= /=) 7
1.7.3 Les opérateurs d'incrémentation et décrémentation (++ --) 7
1.7.4 Les opérateurs de comparaison (== < > <= >= !=) 8
1.7.5 Les opérateurs logiques (|| && ! ^) 8

1.8 Les conditions 9

1.8.1 if / if else /else 9
1.8.2 switch case default break 9

1.9 Nombres aléatoires 10

1.9.1 La méthode Math.random() 10
1.9.2 La génération correcte d'un nombre entier aléatoire entre deux limites 10

1.10 Les boucles 10

1.10.1 for 11
1.10.2 While 11
1.10.3 do-while 11

1.11 Les tableaux 12

1.11.1 Déclaration et création et taille d'un tableau 12
1.11.2 Remplir un tableau avec une valeur 12
1.11.3 Lire et écrire dans un tableau 13

1.12 Les méthodes 13

1.12.1 Déclaration, paramètres et type de retour 13
1.12.2 Retourner une valeur 13
1.12.3 Appel d'une méthode et récupération de sa valeur de retour 13

1.13 Les chaînes de caractères 13

1.13.1 La classe String 13
1.13.2 Tableau des méthodes principales et leur utilité 14

2 Algorithmique de base – méthodes ou codes fréquemment utiles 16

2.1 Code qui teste si le nombre est positif ou négatif 16

2.2 Code qui test si le nombre est pair ou impair 16

2.3 Code qui échange les valeurs de 2 Integer 16

2.4 Remplir un tableau avec une même valeur 16

2.5 Rechercher la position de la première occurrence d'une valeur dans un tableau 17

2.6 Rechercher la position de la dernière occurrence d'une valeur dans un tableau 17

2.7 Remplacer une valeur par une autre dans un tableau.....17

2.8 Compter le nombre d'occurrence d'une valeur dans un tableau 17

2.9 Trouver la plus petite valeur contenue dans un tableau.....18

2.10 Trouver la plus grande valeur contenue dans un tableau 18

2.11 Calculer la moyenne des valeurs contenues dans un tableau 18

2.12 Remplir un tableau avec des valeurs aléatoires.....18

3 Scanner ..... Erreur ! Signet non défini.

4 Conclusions 19

4.1 Ce que j'ai apprécié 19

4.2 Ce que j'ai moins apprécié.....19

4.3 Mon auto-évaluation.....19

4.4 Conclusion.....19

1 Les bases de Java

1.1 La méthode main(), utilité et sa déclaration exacte

La méthode main sert à démarrer l'exécution d'un programme. Sans la méthode main le programme java ne saurait pas quoi exécuter. Quand on code on le fait dedans. Tout ce qui est ce que le programme va exécuter. La méthode main s'écrit comme ça :
public static void main(Strin[] args){}

1.2 Les 8 types de base de Java (liste, limites, … reprendre de « Types de données »)

Les 8 types de base sont des type de donnée que l'on peut mettre dans une variable. Les voici :

Primitive	Signification	Taille	Plage de valeurs acceptée
char	°Caractère Unicode	°16 bits	°Un seul caractère
byte	Chiffre/nombre	8 bits	-128-127
short	Chiffre/nombre	16 bits	-32'768-32'767
int	Chiffre/nombre	32 bits	-2^31-2^31-1
long	Chiffre/nombre	64 bits	-2^63-2^63-1
float	Nombre décimaux	32 bits	7 décimales
double	Nombre décimaux	64 bits	15 décimales
boolean	Vrai/Faux	1 bit	True/False

1.3 Déclaration d'une variable et affectation d'une variable

Pour déclarer une variable on peut utiliser un type de base ou un "String". Si l'on souhaite une variable numérique nous allons privilégier les huit type de base saut le char et le boolean. Char étant pour des caractère unicode comme le symbole "¥". Et le boolean pour dire si c'est vrai ou faux. Si l'on souhaite une variable alphabétique l'on utilise "String".

Pour déclarer la variable on écrit ça dans la méthode main :

On va écrire une variable âge :

```
int age = 16 ;
```

On vient de créer une variable. Pour le String il faut écrire la valeur entre guillemets

Et pout le Char on met des guillemets simples.

Une norme dit que toutes les variables sont écrites en minuscule et que s'il y a plusieurs mots à partir du deuxième mot on met une majuscule au début. Exemple :

```
int ageLegale = 18 ;
```

1.4 Déclaration d'une constante

Une constante est une variable qui ne va jamais changer et qui peut être utiliser dans toutes les méthodes. Donc on écrit une constante en dehors de la méthode main.

Voici un exemple de constante déclarer :

```
public final static int AGE_MAJORITE = 18
```

Une norme dit que toutes les constantes s'écrivent en majuscule.

1.5 Les commentaires (une ligne ou plusieurs lignes)

Dans notre code on peut mettre des commentaires pour expliquer notre code, savoir à quoi il sert, s'organiser,…

Pour écrire un commentaire il faut écrire deux barre oblique comme dans l'exemple ci-dessous.

```
int age = 16 ; //ceci est un comentaire
```

Il est très recommandé d'écrire des commentaires pour ne pas se perdre dans notre code ou pour se rappeler plus tard de comment fonctionne la ligne de code.

Pour écrire des commentaires sur plusieurs lignes on doit faire "/"

1.6 L'écriture sur la console avec la méthode sout (println et print)

Si l'on veut afficher du texte ou bien même des variables nous avons la commande :

```
System.out.println("Bonjour");
```

Si on écrit println il y aura un retour à la ligne à chaque fois si on écrit print il n'y aura pas de retour à la ligne.

Pour ne pas écrire cette commande à chaque fois nous avons la possibilité il y a le raccourci "sout". Pour afficher du texte brut il faut l'écrire dans des guillemets et pour ajouter des variables plus du texte brut on écrit comme ça :

```
int age = 16 ;

System.out.println("J'ai " + age);
```

1.7 Les opérateurs en Java

1.7.1 Les opérateurs de calcul (+ - * / %)

Les opérateurs de calcul en java servent à faire tout ce qui à avoir avec les calculs mathématiques. Les voici est comment les écrire :

Addition :

```
int age = 16 ;

age = age + 1 ; //la variable sera égale à 17
```

Soustraction :

```
int age = 16 ;

age = age - 1 ; //la variable sera égale à 15
```

Multiplication :

```
int age = 16 ;
age = age * 3 ; //la variable sera égale à 48
```

Division :

```
int age = 16 ;
age = age * 3 ; //la variable sera égale à 48
```

Modulo (sert à savoir combien il y aura de reste après une division) :

```
int age = 17 ;
age = age % 3 ; //la variable sera égale à 1
```

1.7.2 Les opérateurs d'assignation (= += -= *= /=)

Les opérateurs d'assignation sont utilisés pour attribuer une valeur à une variable. Il y a plusieurs opérateurs. Les voici :

Le plus "=" assigne une valeur à la variable

int age = 16 ;

Le "+=" sert à ajouter une valeur à une variable

int age = 16;

age += 3; //maintenant la variable age sera 19, c'est comme si on faisait "age = age + 3;"

Le "-=" sert à enlever une valeur à une variable

int age = 16 ;

age -= 3 ;//maintenant la variable age sera 13, c'est comme si on faisait "age = age - 3;"

Le "*=" sert à multiplier la variable par une valeur donnée

int age= 16;

age *=3 ;//maintenant la variable age sera 48, c'est comme si on faisait "age = age * 3;"
Le "/=" sert à diviser la variable par une valeur donnée
int age= 16;

age /=4 ;//maintenant la variable age sera 4, c'est comme si on faisait "age = age / 4;"

1.7.3 Les opérateurs d'incrémentation et décrémentation (++ --)

Les opérateurs d'incrémentation et décrémentation servent simplement à enlever ou ajouter 1 à une variable. Ces opérateurs sont souvent utilisés dans les boucles.

Exemple "++":

```
int age = 16 ;
age++ ;//la variable age sera égal à 17
```

Exemple "--"

```
int age = 16 ;
age-- ;//la variable age sera égal à 15
```

1.7.4 Les opérateurs de comparaison (== < > <= >= !=)

Les opérateurs de comparaison servent à exprimer une condition de continuation dans une boucle ou faire une condition dans un "if". Exemple :

```
int age = 16;
int ageMajeur = 18;
if(age<ageMajeur){
system.out.println("Tu es mineur");
}else{
system.out.println("Tu es majeur")
}
```

Voici un exemple de l'utilisation de "<". On peut le remplacer par les autres opérateurs de comparaison selon ce que l'on souhaite.

Voici la signification des opérateurs de comparaison :

- ==(est égal à)
- <(plus petit à)
- >(plus grand à)
- <=(plus petit ou égal à)
- >=(plus grand ou égal à)
- !=(n'est pas égal à)

1.7.5 Les opérateurs logiques (|| && ! ^)

Les opérateurs logiques sont également utilisés dans les boucles on les "if".

La signification des opérateurs logiques sont celles-ci :

|| : s'il y a au moins une condition qui est vrai

&& : si les deux conditions sont vraies

! : inverse la valeur

^ : s'il y a qu'une seule condition qui est vrai

Example concret :

```
int age = 36
if(age<30 && age>40){
system.out.println("tu n'es pas dans la trentaine")
}else{system.ou.println("tu es dans la trentaine")
}
```

On peut remplacer l'opérateur de comparaison selon notre envie de condition

1.8 Les conditions

Les conditions servent à dire à notre programme ce qu'on veut qu'il fasse selon ce qu'on lui dit comme conditions.

1.8.1 if / if else /else

Quand on veut faire une condition avec deux options va utiliser if/else. En revanche si l'on souhaite faire une condition avec plusieurs options on utilisera if;if else;else.

Exemple de l'utilisation de if/else :

```
int age = 16;
int ageMajeur = 18;
if(age<18){
system.out.println("Tu es mineur");
}else{
system.out.println("Tu es majeur")
}
Exemple de l'utilisation de if/else if/else :
int age = 16;
int ageMajeur = 18;
if(age<17){
system.out.println("Tu es mineur");
}else if(age==17){
system.out.println("Tu es presque majeur")
} else{
system.out.println("Tu es majeur")
}
```

1.8.2 switch case default break

Les conditions avec les switch servent à faire des action selon le numéro de la valeur.

Un exemple expliquera mieux comment ça fonctionne.

Exemple :

int noteExam = 5;

```
switch(noteExam){//nous somme entrain de dire qu'elle valeur nous souhaitons

case 1:system.out.println("très mauvaise note!!!");//si la valeur est 1
afficher le texte

break;

case 2:system.out.println("paf fameux"); //si la valeur est 2 afficher le
texte

break;

case 3:system.out.println("presque"); //si la valeur est 3 afficher le texte

break;

case 4:system.out.println("c'est limite!"); //si la valeur est 4 afficher le
texte

break;

case 5:system.out.println("c'est bien"); //si la valeur est 5 afficher le
texte

break;

case 6:system.out.println("excelent!!!"); //si la valeur est 6 afficher le
texte

break;

default:system.out.println("la note n'est pas valable"); //sinon afficher le
texte

break;

}
```

Complément : Au début nous disons que nous allons nous référer à la valeur"noteExam". Ça veut dire que pour chaque "case" (cas en français) ça va voir en fonction de la valeur choisi. Ça veut dire que dans le "case 1" on dit si la valeur est égale à un faire ça. C'est pareil avec les autres "cases". Le "default veut dire" si aucun cas de correspond faire ça.

1.9 Nombres aléatoires

1.9.1 La méthode Math.random()

La méthode math.random est une méthode utiliser pour créer des nombres aléatoires. Pour ceci il faut définir un maximum et un minimum.

1.9.2 La génération correcte d'un nombre entier aléatoire entre deux limites

Voici comment s'écrit la méthode math.random :

```
int nb = s

//pour le max et le min on choisit généralement de la faire avec des
constantes
```

1.10 Les boucles

Les boucles servent à dire à notre programme de faire quelque chose tant qu'une certaine condition est active. Les boucles sont utiles pour parcourir des **tableau**x, faire des comptes à rebours, répété des actions,... Il y a beaucoup plus d'utilité aux boucles. Chaque "tour" d'une boucle s'appelle itération.

1.10.1for

Les boucle for sont assez pratique à utiliser car la consigne est toute à un même endroit.

Un exemple :

```
for(int i=0;i<6;i++){
system.out.println(i);
}
```

system.out.println("c'est parti !!!");

Compréhension : dans le for la première case est une variable qui sert à initialiser la boucle. Dans la deuxième case c'est la condition pour que la boucle continue. La troisième sert à faire une incrémentation. La variable qui est écrite en premier n'existe que dans la boucle.

1.10.2While

Une boucle while est la même chose qu'une boucle for main écrite différemment.

Exemple plus concluant :

```
Int nb = 1;
while(nb<6){
system.out.println(nb);
nb++,
}
```

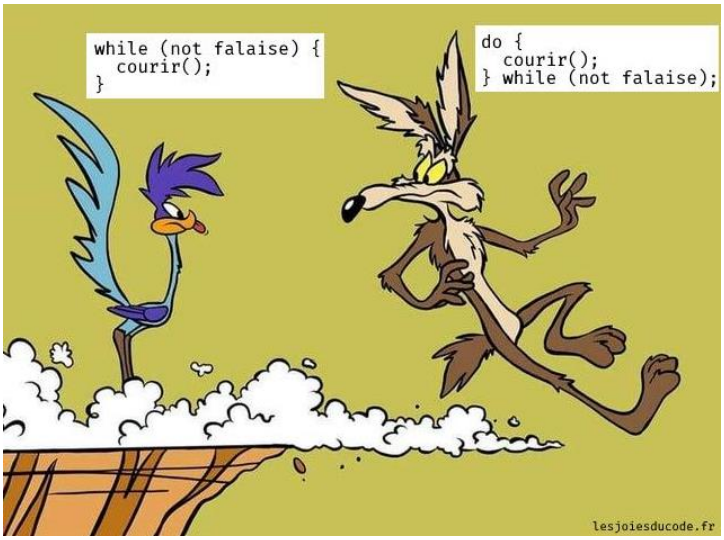
System.out.println("c'est parti!!!")

Complément : c'est la même chose que pour le for sauf qu'ici la déclaration est hors de la boucle et au début, la condition est entre paranthèse; et l'incrémentation est dans la boucle.

1.10.3do-while

Le do-while traduction de fait pendant que. C'est la même chose que le while et le for à une différence près. Il va d'abord exécuter les actions avant de voir s'il doit s'arrêter alors que le while et le for vont d'abord voir s'ils doivent rentrer dans la boucle.

Un exemple concret avec une image :



Les mots-clé break et continue

Les mots-clés break et continue servent a dire des ordre de s'arrêter ou de continue dans une boucle.

Exemple : On veut faire une boucle qui compte de 0 à 5 mais sans compter le 3 donc on dit que si le i est égal à trois de continuer le

```
for (int i = 0; i < 6; i++) {
            if (i == 3) {
                continue;
            }
            System.out.println(i);
        }
```

Le continue sert à aller à la prochaine itération sans exécuter le reste et le break à stopper la boucle.

1.11 Les tableaux

Les tableaux servent à ajouter plusieurs valeurs à une seule variable. En gros on stock des valeurs dans une variable et on va rechercher celle qu'on veut selon l'endroit où elle est stockée.

1.11.1 Déclaration et création et taille d'un tableau

Voici comment déclarer une variable : Pour définir une variable tableau il ne faut pas oublier de mettre des crochets après le type de variable.

```
int[] tab = new int[5] ;
```

Dans ce cas-là nous déclarons le tableau, lui disons que nous souhaitons 5 valeurs à stocker mais on les définie pas encore. (on change le type de variable selon ce que l'on souhaite mettre dans le tableau). Un tableau auquelle on affecte aucune valeur est 0

1.11.2 Remplir un tableau avec une valeur

Pour remplir un tableau on va d'abord comprendre comment le tableau garde les valeurs. Pour cela voici une image visuelle de ce qu'un tableau est :

0	1	2	3	4
x	x	x	x	x

La ligne du haut est l'emplacement dans le tableau. La case une du tableau s'appelle 0. Donc pour définir une valeur on va mettre :

```
int[] tab = new int[5];
tab[0] = 5 ;
```

Pour dire quelle case on souhaite ajouter une valeur on met le numéro dans des crochets et on dit la valeur.

Exemple du tableau après l'exécution du code ci-dessus :

0	1	2	3	4
5	0	0	0	0

1.11.3 Lire et écrire dans un tableau

Pour lire dans un tableau nous allons faire une boucle qui va dans chaque case puis on rajoute un sout :

```
for(int i = 0 ;i < tab.length ; i++){
System.out.println(tab[i]) ;
}
```

Pour écrire dedans nous allons faire une boucle avec un math.random :

```
for (int i = 0; i < tab.length; i++) {
tab[i]= (int) (Math.random() * (10 - 1 + 1) + 1);
}
```

1.12 Les méthodes

1.12.1 Déclaration, paramètres et type de retour

Voici comment déclarer une méthode :

```
public static int[] creerTableau(int taille) {
}
```

Il y a plusieurs choses à prendre en compte ; le paramètre et le type de retour(il faut créer la variable). Le paramètre est une variable que nous souhaitons utiliser dans notre méthode. Le type de retour est ce que nous voulons retourner dans le main(un nombre entier, un tableau, un nombre décimale,…).

1.12.2 Retourner une valeur

Pour retourner une valeur on va écrire à la fin de notre méthode « return… ». Example :

```
public static int devinerTailleTab(int[] tab) {
int tailleTabDeviner = tab.length;

return tailleTabDeviner;
}
```

1.12.3 Appel d'une méthode et récupération de sa valeur de retour

Pour appeler une méthode dans le main nous allons écrire ça :

```
remplirTableauFixe(tab);
```

En écrivant le nom de la méthode nous allons l'appeler et entre parenthèse il y aura les paramètres.

1.13 Les chaînes de caractères

1.13.1 La classe String

La classe String est un type de variable qui sert à mettre du texte. Voici un déclaration de String :

```
String = "bonjour" ;
```

1.13.2Tableau des méthodes principales et leur utilité

Comment voir si un String est vide : il faut utiliser le « .isEmpty »

```
String texte = "Bonjour";
        if (texte.isEmpty()) {
            System.out.println("La chaîne est vide.");
        } else {
            System.out.println("La chaîne n'est pas vide.");
        }
```

Comment changer un String de minuscule à majuscule : il faut utiliser « .toUpperCase »

```
String texte = "bonjour";
        String texteMajuscule = texte.toUpperCase();
        System.out.println(texteMajuscule); // Affiche "BONJOUR"
```

Comment voir si deux String sont identique : il faut utiliser « .equals() »

```
String texte1 = "Bonjour";
        String texte2 = "Bonjour";

        if (texte1.equals(texte2)) {
            System.out.println("Les chaînes sont identiques.");
        } else {
            System.out.println("Les chaînes ne sont pas identiques.");
        }
```

1.14 Scanner

Voici un exercice fait en classe où un scanner à été utilisé. En le lisant tu comprendra comment ça fonctionne.

```
package devoirs.devoir05;

import java.util.Scanner;

public class devoir05 {
    public final static int MAX = 100;
    public final static int MIN = 0;

    public static void main(String[] args) {
        int nbre = (int) (Math.random() * (MAX - MIN + 1)) + MIN;
        Scanner scanner = new Scanner(System.in);
        System.out.print("Entrez une valeur entre 1 et 100 : ");
        int guess = scanner.nextInt();
        while (guess != nbre) {
            if (guess > nbre) {
                System.out.println("Trop grand");
                guess = scanner.nextInt();
```

```
            } else if (guess < nbre) {
                System.out.println("Trop petit");
                guess = scanner.nextInt();
            }
        }
        System.out.println("Bravo, trouvé");
        scanner.close();

    }

}
```

Globalement il y a deux trois trucs à faire :

1. import le scanner
2. créer la variable scanner
3. dire quand l'utilisateur peut l'utiliser
4. fermer le scanner

Peu compréhensible mais moi j'arrive à comprendre comment l'utiliser…

2 Algorithmique de base – méthodes ou codes fréquemment utiles

Bout de code utile :

2.1 Code qui teste si le nombre est positif ou négatif

```
int valeurATester = 2;
        if (valeurATester <= 0) {
            System.out.println("La valeur  est négative");

        } else {
            System.out.println("La valeur est positive");
        }
```

2.2 Code qui test si le nombre est pair ou impair

```
int valeurAtester = 263;
        if (valeurAtester % 2 < 1) {
            System.out.println("La valeur est paire");
        } else {
            System.out.println("La valeur est impaire");

        }
```

2.3 Code qui échange les valeurs de 2 Integer

```
int variable1 = 10;
        int variable2 = 9;
        System.out.println("Le contenu de la variable1 est " + variable1);
        System.out.println("Le contenu de la variable2 est " + variable2);
        System.out.println("...traitement...");
        int variable3 = variable1;
        variable1 = variable2;
        variable2 = variable3;
        System.out.println("Le contenu de la variable1 est " + variable1);
        System.out.println("Le contenu de la variable2 est " + variable2);
```

2.4 Remplir un tableau avec une même valeur

```
for (int i = 0; i < tab.length; i++) {
            tab[i] = 1;
        }
```

2.5 Rechercher la position de la première occurrence d'une valeur dans un tableau

```
for (int i = 0; i < tab.length; i++) {
            if (tab[i] == nbrChercher) {
                System.out.println("La valeur " + nbr + " a été trouvée à la 1ère
position N°" + i);
                break;
            }
}
```

2.6 Rechercher la position de la dernière occurrence d'une valeur dans un tableau

```
for (int i = tab.length-1; i!=-1; i--) {

            if (tab[i] == nbrChercher) {
                System.out.println("La valeur " + nbr + " a été trouvée à la der-
nière position N°" + i);
                break;
            }
}
```

2.7 Remplacer une valeur par une autre dans un tableau

```
for (int i = 0; i < tab.length; i++) {
            if (tab[i] == valeurRecherche) {
                tab[i] = valeurRemplacer;
            }
        }
```

2.8 Compter le nombre d'occurrence d'une valeur dans un tableau

```
int[] tab=new int[]{2,6,6,2,1,1,1,1};
        int nbr = 1;
        int occurence=0;
        for (int i = 0; i < tab.length; i++) {
            if (tab[i] == nbr) {
                occurence++;
            }

        }
        System.out.println("le nombre "+nbr+" à été trouvée "+occurence+" fois");
```

2.9 Trouver la plus petite valeur contenue dans un tableau

```
public static int rechercheValeurMin(int[] tab) {
        int min = tab[0];
        for (int i = 1; i < tab.length; i++) {
            min = Math.min(min, tab[i]);
        }
        return min;
    }
```

2.10 Trouver la plus grande valeur contenue dans un tableau

```
 public static int rechercheValeurMax(int[] tab) {
        int max = tab[0];
        for (int i = 1; i < tab.length; i++) {
            max = Math.max(max, tab[i]);
        }
        return max;
    }
```

2.11 Calculer la moyenne des valeurs contenues dans un tableau

```
public static float moyenneTableau(int[] tab, float moyenne) {
        for (int i = 0; i < tab.length; i++) {
            moyenne += tab[i];
        }
        moyenne /= tab.length;
        return moyenne;
    }
```

2.12 Remplir un tableau avec des valeurs aléatoires

```
public static int[] remplirTableauAleatoire(int[] tab) {
        for (int i = 0; i < tab.length; i++) {
            int valeurCelulle = (int) (Math.random() * (MAXCELLULE - MINCELLULE +
1) + MINCELLULE);
            tab[i] = valeurCelulle;
        }

        return tab;
    }
```

3 Conclusions

3.1 Ce que j'ai apprécié

J'ai vraiment tout apprécié car le dev c'est ce que je préfère dans l'informatique. J'ai beaucoup aimé comment était structuré les cours(exercice puis mise en commun).

3.2 Ce que j'ai moins apprécié

C'est étonnant mais j'ai littéralement tout apprécié pendant ce module.

3.3 Mon auto-évaluation

Je trouve que c'est le module où je me débrouille le mieux. Malgré que j'ai eu quelque difficultés à certain moment j'ai toujours réussis à me débrouiller. C'est la matière que je comprends le mieux.

3.4 Conclusion

Le module c'est parfaitement dérouler et je l'ai beaucoup apprécier.