

CAHIER DE CONCEPTION GÉNÉRALE

Projet Java-Math 2017/2018

# 

# 

# 

# 

Version: CCG

Auteurs : Billaud William, Aubois Alexnadre

ISEN Toulon - Yncrea

Maison du Numérique et de l'Innovation

Place Georges Pompidou

Toulon

**Description du document**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Version** | | **Confidentialité** | |
| Cahier de conception générale | 1.0 | | Usage Externe | |
| **Redacteurs** | **Nom** | **Fonction** | **Date** | **Visa** |
| Billaud William | Membre du projet | 19/12/2017 |  |
| Aubois Alexandre | Membre du projet |
| **Vérificateurs** | Billaud William | Membre du projet | 19/12/2017 |  |
| Aubois Alexandre | Membre du projet |  |
| **Approbateurs** | Billaud William | Membre du projet | 19/12/2017 |  |
| Aubois Alexandre | Membre du projet |  |
| **Destinataire** | | **Fonction** | | **Organisme** |
| Public | |  | | ISEN |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Redacteur** | **Modifications** |
| 1.0 | 19/12/2017 | FMC | Mise en forme |
| 1.1 | 19/12/2017 | Alexandre Aubois | Remplissage |
| 1.2 | 19/12/2017 | William Billaud | Ajout des exigences fonctionnelles |
| 1.2.1 | 20/12/2017 | Alexandre Aubois | Correction des formulations |
| 1.3 | 20/12/2017 | Alexandre Aubois | Ajout partie « Moteur d’affichage » |

**SOMMAIRE**

[1 Introduction 5](#_Toc501617244)

[2 Liste d’exigences 5](#_Toc501617245)

[2.1 Exigences fonctionnelles 5](#_Toc501617246)

[2.2 Exigences de programmations 5](#_Toc501617247)

[3 Objets fonctionnelles 6](#_Toc501617248)

[3.1 Architecture des Modules 6](#_Toc501617249)

[3.2 Données utilisées par chaque objet 6](#_Toc501617250)

[3.3 Echange de données entre les objets 6](#_Toc501617251)

[4 Implémentation des modules 7](#_Toc501617252)

[4.1 Moteur de calcul 7](#_Toc501617253)

[4.2 Le moteur d’affichage 8](#_Toc501617254)

[4.3 Le contrôleur 8](#_Toc501617255)

[Le contrôleur permet de faire le liant entre la partie Modèle et la partie Vue de notre modèle MVC (Modèle-Vue-Contrôleur). 8](#_Toc501617256)

[4.4 Le module de lecture des données 8](#_Toc501617257)

[4.5 Le module d’enregistrement des données 9](#_Toc501617258)

[5 Arbre des fonctions et flux de données 9](#_Toc501617259)

[5.1 Arbre d’appel et flux de données 9](#_Toc501617260)

1 Introduction

Vous trouverez ici toutes les informations relatives à l’architecture du logiciel. Ainsi vous trouverez ci-dessous le découpage en modules fonctionnels du programme, ainsi que la définition des différentes données utilisées par le programme. Vous trouverez aussi l’arbre d’appel des fonctions ainsi que les flux de données et une liste des fonctions du programme regroupées par module.

2 Liste d’exigences

2.1 Exigences fonctionnelles

Le programme répondra aux exigences fonctionnelles suivantes :

EF\_001 : Le programme pourra calculer une transformée de fourrier d’une série de nombre réels en virgule flottante depuis un document au format CSV.

EF\_002 : Les données seront écrites au format x, y, …. Avec x et y des réels

EF\_003 : La taille de la série sera une puissance de 2.

EF\_004 : les résultats pourront être écrit dans un nouveau fichier au format CSV.

EF\_005 : Les données seront écrites de la forme : « Re(x) + i Im(x),Re(y) +i Im(y), etc… », avec x et y les coefficients de fourrier.

EF\_006 : Re(x) et Im(x) auront une précision de 3 chiffres après la virgule maximum.

EF\_007 : Le programme pourra effectuer la transforme inverse d’une série de fourrier passé en paramètre grâce à un fichier CSV respectant les normes ci-dessus.

EF\_008 : La taille de la série sera une puissance de 2.

EF\_009 : L’utilisateur pourra choisir le fichier source grâce à un bouton via l’interface graphique.

EF\_010 : L’utilisateur pourra cocher le type de traitement souhaitée sur l’interface graphique.

EF\_011 : L’utilisateur pourra choisir le nom du fichier de sortie.

EF\_012 : L’utilisateur pourra effectuer la transformée de fourrier d’une série de nombre complexe passée en paramètre dans un fichier au format CSV.

EF\_013 : Les données seront au format « Re(x) + i Im(x), Re(y) + i Im(y), … » avec x et y des nombre complexes.

EF\_014 : En cas de non-respect des règles citée ci-dessus la donnée ne seras pas traitée.

EF\_015 : La séparation avec les décimales seras faites avec des ‘.’ Exemple : douze virgule 6 s’écriras « 12.6 ».

EF\_016 : Un fenêtre POP-UP devra se lancer lors d’une erreur.

EF\_017 : La fenêtre POP-UP devra contenir le message d’erreur.

EF\_018 : La fonction des transformer pourra être visible dans un repère x ;y.

EF\_019 : L’utilisateur pourra enregistrer les résultats dans un fichier csv grâce à un bouton via l’interface graphique.

EF\_020 : Le développement utilisera le pattern MVC (Modèle-Vue-Contrôleur).

EF\_021 : Les résultats seront affichés via un graphe.

2.2 Exigences de programmations

Le programmes répondras aux l’exigences non fonctionnelles suivantes :

1 : le programme seras écris en langage Java 9

# 3 Objets fonctionnelles

3.1 Architecture des Modules

Il sera développé quatre modules différents :

* Le moteur de calcul
* Le moteur d’affichage
* Le contrôleur
* Le module d’enregistrement et des lectures des données.

## 3.2 Données utilisées par chaque objet

Pour l’objet « Complexe » il est utilisé :

* Une partie réelle (un *float*)
* Une partie imaginaire (un *float*)
* Un *logger*

Pour l’objet « FFT » :

* Un tableau d’objet « Complexe »
* Un *logger*

## 3.3 Echange de données entre les objets

Les objets ne s’échangent pas de données

# 4 Implémentation des modules

## 4.1 Moteur de calcul

L’objet « Complexe » :

L’objet sera composé de 2 attributs (2 *float*) et d’un *looger*.

L’objet « Complexe » sera composé de plusieurs constructeurs :

* Avec deux arguments passé en paramètre (qui sont le réel et l’imaginaire du nombre à créer.
* Avec un argument passé en paramètre (qui est l’argument du nombre complexe)

Les deux constructeurs initialiseront les attributs que contiennent l’objet.

L’objet aura les méthodes de *getter* et de *setter* pour les 2 attributs.

Il possèdera aussi les méthodes suivantes :

* Add : qui prendra en paramètre un nombre complexe et qui permettra d’additionner deux nombres complexes, et de retourner le résultat.
* Sub : qui prendra en paramètre un nombre complexe et qui permettra de soustraire deux nombres complexes, et de retourner le résultat.
* Multiply : qui prendra en paramètre un nombre complexe et qui permettra de multiplier deux nombres complexes, et de retourner le résultat.
* toString : qui permettra d’afficher la partie entière et la partie imaginaire d’un nombre.
* Equals : qui prendra en paramètre un objet et qui permettra de tester l’égalité entre deux objets, avec en retour un booléen à *true* si les deux objets sont égaux et à *false* si les deux objets sont différents.
* Conjugue : qui permettra de calculer le conjugué d’un nombre complexe.

L’objet « FFT » :

L’objet sera composé d’un attribut (un tableau d’objet « Complexe ») ainsi que d’un *logger*.

L’objet « FFT » sera composé d’un seul constructeur prenant en entrer une taille (entier). Ce constructeur pourra tester si la taille entrer est supérieur à 0, et enverra un cas d’erreur sinon. Passé ce test elle créera ensuite un tableau d’objet complexe de taille (2^t), « t » étant la valeur passé en paramètre.

L’objet sera composé de deux *getter.* Un prendra en paramètre un indice et qui permettra d’accéder à la valeur correspondante dans le tableau. Elle retournera le nombre complexe correspondante. L’autre prendra permettra d’accéder a l’attribut de l’objet, en retournant ce dernier.

Il possèdera aussi les méthodes suivantes :

* CalculeFFTReelle : qui prendra en paramètre un tableau de *float.* Ellepermettra de calculer la transformer de Fourier rapide d’un nombre réel, qui sont les valeurs contenues dans le tableau passer en paramètre.
* CalculeFFTComplexe : qui prendra en paramètre un tableau d’objet « Complexe »*.* Ellepermettra de calculer la transformer de Fourier rapide d’un nombre complexe, qui sont les valeurs contenues dans le tableau passer en paramètre.
* InverseFFT : qui prendra en paramètre un tableau d’objet « Complexe »*.* Ellepermettra de calculer la transformer de Fourier inverse d’un nombre complexe, qui sont les valeurs contenues dans le tableau passer en paramètre.

## 4.2 Le moteur d’affichage

Le moteur d’affichage utilisera swing pour faire afficher un interface homme-machine.

D’un point de vue graphique, l’interface sera composée de trois boutons et une *checkbox* permettant à l’utilisateur de choisir le type de transformer qu’il veut effectuer sur les trois possibles. Le premier bouton permettant d’ouvrir un fichier csv, et seulement csv. Le second bouton permettra de lancer le programme. Le dernier permettra d’enregistrer les résultats dans un nouveau fichier csv. La *checkbox* permettra de choisir entre les trois possibilités suivantes :

* Transformer de Fourier réel rapide
* Transformer de Fourier complexe
* Transformer de Fourier inverse

L’utilisateur pourra voir la fonction de la transformer qu’il aura choisi et s’afficher dans un repère x ;y. Une page pop-up devra se lancer lors des cas d’erreur et afficher le message d’erreur correspondante.

## 4.3 Le contrôleur

## Le contrôleur permet de faire le liant entre la partie Modèle et la partie Vue de notre modèle MVC (Modèle-Vue-Contrôleur).

Ce module sera composé de 2 attributs et d’un *logger*. Le premier attribut sera un objet de type « FFT » et un objet de type « Fenetre ».

Il contiendra le getter et le setter de l’attribut de type « Fenetre », et le setter de l’attribut de type « FFT ». Ce module aura aussi un constructeur. Il contiendra une méthode « notifyAction » permettant d’envoyé au Modèle les paramètres ainsi que les méthodes qu’il doit lancer.

## 4.4 Le module de lecture des données

Ce module permettra de lire les données contenues dans un fichier csv.

Ce module ne contiendra pas d’attribut mis-a-part un logger.

Il sera composé de deux méthodes. Une qui permet de *parser* les données lors de l’envoie de nombres complexes, et une autre qui permettra de *parser* les données lors de l’envoie de nombres réels.

Il n’y a pas de constructeur, ce sera donc une classe statique.

## 4.5 Le module d’enregistrement des données

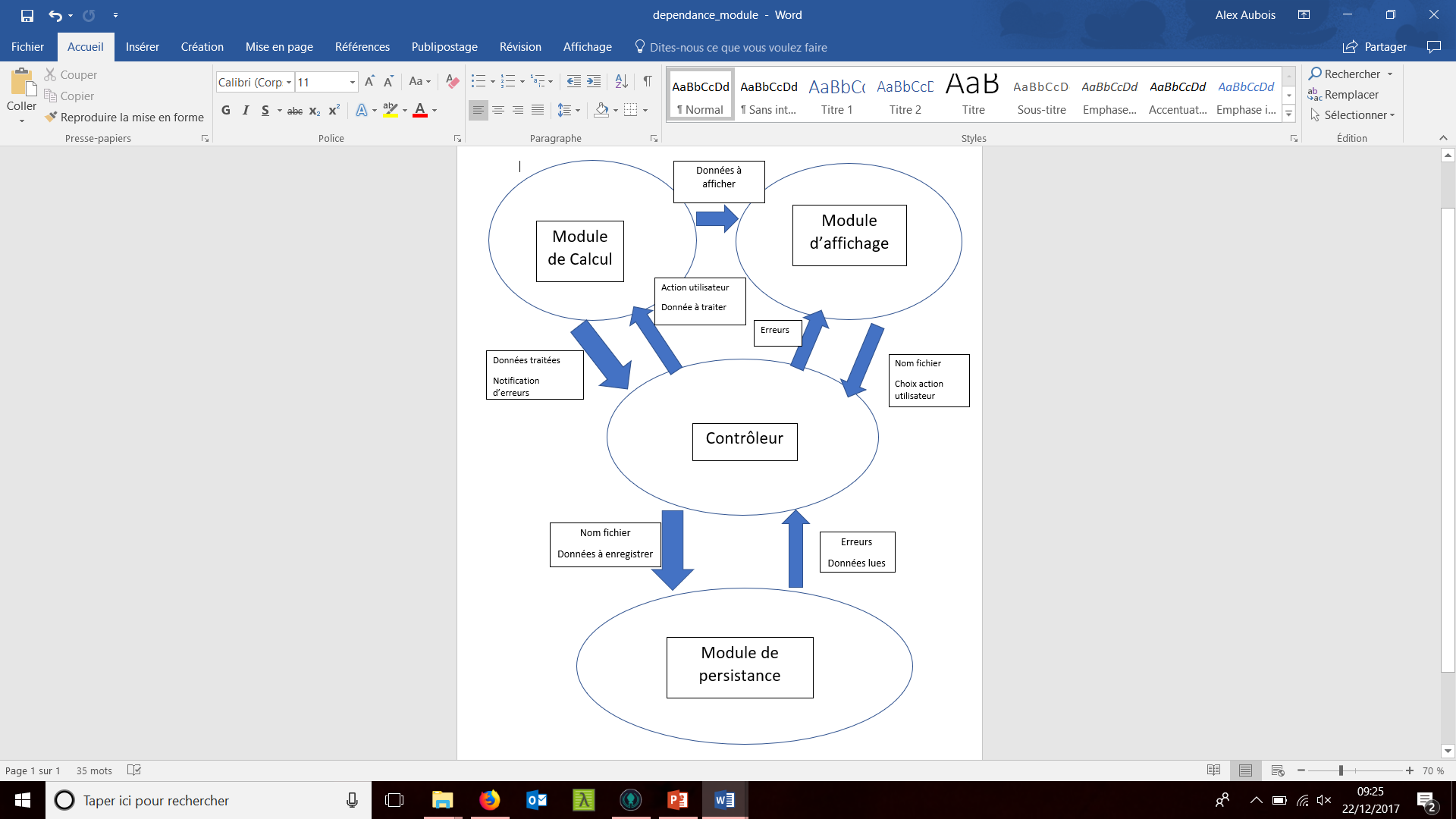
Ce module permettra d’enregistrer les résultats de la transformer effectuer dans un fichier csv.

Ce module ne contiendra pas d’attribut mis-a-part un logger.

Il sera composé d’une méthode. Cette méthode permet d’écrire les données obtenues dans un nouveau fichier csv après la transformer. Ces données seront en complexe, peu importe que le résultat soit complexe ou réel.

Il n’y a pas de constructeur, ce sera donc une classe statique.

# 5 Arbre des fonctions et flux de données

.