

CAHIER DE CONCEPTION GÉNÉRALE

Projet Java-Math 2017/2018

# 

# 

# 

# 

Version: CCG

Auteurs : Guillaume Desrumaux, Félix Herrenschmidt

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 Interne | |
| **Redacteurs** | **Nom** | **Fonction** | **Date** | **Visa** |
| Guillaume Desrumaux | Membre du projet | 19/12/2017 |  |
| Félix Herrenschmidt | Membre du projet |
| **Vérificateurs** | Guillaume Desrumaux | Membre du projet | 19/12/2017 |  |
| Félix Herrenschmidt | Membre du projet |  |
| **Approbateurs** | Guillaume Desrumaux | Membre du projet | 19/12/2017 |  |
| Félix Herrenschmidt | Membre du projet |  |
| **Destinataire** | | **Fonction** | | **Organisme** |
| Privé | |  | | ISEN |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Redacteur** | **Modifications** |
| 1.1 | 19/12/2017 | Félix Herrenschmidt | Ajout des exigences |

**SOMMAIRE**

[1 Introduction 5](#_Toc501525546)

[2 Liste d’exigences 5](#_Toc501525547)

[2.1 Exigences fonctionnelles 5](#_Toc501525548)

[2.2 Exigences de programmations 5](#_Toc501525549)

[3 Objets fonctionnelles 6](#_Toc501525550)

[3.1 Architecture des Modules 6](#_Toc501525551)

[3.2 Données utilisées par chaque objet 6](#_Toc501525552)

[3.3 Echange de données entre les objets 6](#_Toc501525553)

[4 Implémentation des modules 7](#_Toc501525554)

[4.1 Moteur de calcul 7](#_Toc501525555)

[4.2 Le moteur d’affichage 8](#_Toc501525556)

[4.3 Le *controller* 8](#_Toc501525557)

[4.4 Le module d’enregistrement et des lectures des données 8](#_Toc501525558)

[5 Arbre des fonctions et flux de données 9](#_Toc501525559)

[5.1 Arbre d’appel et flux de données 9](#_Toc501525560)

[5.2 Description des fonctions 9](#_Toc501525561)

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 générer et utiliser les nombres complexes : « Re(x) + i Im(x) ».

EF\_002 : Le programme fonctionnera sous le JDK : JAVA 9.

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

EF\_004 : Le programme permettra de calculer une FFT : Réelle.

EF\_005 : Le programme permettra de calculer une FFT : Complexe.

EF\_006 : Le programme permettra de calculer une FFT : Inverse.

EF\_007 : Les différent calcules de FFT devront être stockées dans un objet « FFT ».

EF\_008 : Le programme devra être utilisable sous forme graphique : Choix de la fonction et affichage d’un repère cartésien.

EF\_009 : L’interface graphique suivra un modèle MVC.

EF\_010 : Le programme utilisera un système de « logger », ici « log4j ».

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

## 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’entrée d’objet « Complexe »
* Un tableau d’entrée Réel
* Un tableau de sortie d’objet « Complexe »
* Un *logger*

Pour l’objet « Fenetre » :

* Un *logger*
* Une variable contenant le choix de l’utilisateur
* Un objet contenant le graphique

Pour l’objet « FFTController » :

* Un objet « Fenetre »
* Un objet « FFT »

## 3.3 Echange de données entre les objets

L’objet « FFT » calcul et l’objet « FFTController » se sert de l’objet « Fenetre » pour afficher la vue graphique.

# 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.
* hashCode :
* 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 deux 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. La *checkbox* permettra de choisir entre les trois possibilités suivantes :

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

Par conséquent il existera des méthodes permettant d’écouter le clique en fonction des trois actions possible.

Il existera aussi une méthode permettant de load le fichier envoyer par l’utilisateur.

## 4.3 Le *controller*

## 4.4 Le module d’enregistrement et des lectures des données

# 5 Arbre des fonctions et flux de données

## 5.1 Arbre d’appel et flux de données

.

## 5.2 Description des fonctions