

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

83000 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 | 18/12/2017 |  |
| Félix Herrenschmidt | Membre du projet |
| **Vérificateurs** | Guillaume Desrumaux | Membre du projet | 18/12/2017 |  |
| Félix Herrenschmidt | Membre du projet |  |
| **Approbateurs** | Guillaume Desrumaux | Membre du projet | 18/12/2017 |  |
| Félix Herrenschmidt | Membre du projet |  |
| **Destinataire** | | **Fonction** | | **Organisme** |
| Privé | |  | | ISEN |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Rédacteur** | **Modifications** |
| 1.0 | 18/12/2017 | Félix Herrenschmidt | Ajout des exigences |

**SOMMAIRE**

[1 Introduction 5](#_Toc501697103)

[2 Liste d’exigences 5](#_Toc501697104)

[2.1 Exigences fonctionnelles 5](#_Toc501697105)

[2.2 Exigences de programmations 5](#_Toc501697106)

[3 Objets fonctionnels 5](#_Toc501697107)

[3.1 Architecture des modules 5](#_Toc501697108)

[3.2 Données utilisées par chaque objet 6](#_Toc501697109)

[3.3 Echange de données entre les objets 6](#_Toc501697110)

[4 Implémentation des modules 7](#_Toc501697111)

[4.1 Moteur de calcul 7](#_Toc501697112)

[4.2 Le moteur d’affichage 8](#_Toc501697113)

[4.3 Le contrôleur 8](#_Toc501697114)

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 : La taille de la série sera une puissance de 2.

EF\_003 : Le programme permettra de calculer une transformée de Fourrier : Réelle.

EF\_004 : Le programme permettra de calculer une transformée de Fourrier : Complexe.

EF\_005 : Le programme permettra de calculer une transformée de Fourrier : Inverse.

EF\_006 : Les différents calculs de FFT devront être stockés dans un objet « FFT ».

EF\_007 : Le programme devra être utilisable sous forme graphique.

EF\_008 : L’application suivra un modèle MVC (Modèle-Vue-Contrôleur).

EF\_009 : Le programme utilisera un système de « logger », ici « log4j ».

EF\_010 : L’utilisateur pourra choisir la fonction à tester.

EF\_011 : Le résultat des fonctions s’affichera dans un repère cartésien.

2.2 Exigences de programmations

Le programmes répondra aux exigences non fonctionnelles suivantes :

EP\_001 : Le programme fonctionnera sous le JDK : JAVA 9.

# 3 Objets fonctionnels

3.1 Architecture des modules

Le programme sera développé suivant quatre modules différents :

* Le moteur de calcul
* Le moteur d’affichage
* Le contrôleur d’affichage

## 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 » calculera et l’objet « FFTController » se servira de l’objet « Fenetre » pour afficher la vue graphique.

# 4 Implémentation des modules

## 4.1 Moteur de calcul

* **L’objet « Complexe » :**

L’objet possèdera 3 attributs : (float) reel + (float) imaginaire + le logger.

L’objet « Complexe » sera composé de 2 constructeurs :

* L’objet pourra être construit par un (float) reel et un (float) imaginaire.
* L’objet pourra également être construit par un (float) argument

*Méthodes :*

* Module : Renverra le module du nombre complexe.
* Addition : Renverra un complexe comprenant le résultat de l'addition de ce complexe et du complexe d’entrée.
* Soustraction : Renverra un complexe comprenant le résultat de la soustraction de ce complexe et du complexe d’entrée.
* Conjugue : Renverra le complexe conjugué de ce nombre.
* Multiplication : Renverra un complexe comprenant le résultat de la multiplication de ce nombre et du complexe d’entrée.
* Re : Accesseur sur les réels.
* Im : Accesseur sur les imaginaires.
* **L’objet « FFT » :**

L’objet FFT contiendra 5 attributs : un signal réel (float), un signal complexe (Complexe[]), un signal de sortie (Complexe[]), la taille du tableau (int : puissance de 2).

L’objet « FFT » sera composé de 2 constructeurs :

* L’objet pourra être construit par la taille du tableau et un signal complexe.
* L’objet pourra également être construit par la taille du tableau et un signal réel.

*Méthodes :*

* FFTRéelle : Calculera une FFT à partir de valeurs réelles contenus dans un tableau de flottants.
* FFTComplexe : Calculera une FFT à partir de valeurs complexes contenus dans un tableau de Complexes.
* inverseFFT : Calculera la FFT inverse.
* estPuissance2 : Vérifiera si une valeur est bien une puissance de 2.
* Les différents accesseurs des attributs.

Pour plus d’informations sur les Objets veuillez consulter la « JAVADOC » du programme.

## 4.2 Le moteur d’affichage

L’interface homme machine sera composé d’un graphique permettant d’afficher le spectre des différentes FFT.

Il y aura 3 boutons :

* Le premier, pour calculer et afficher le spectre d’une FFT avec un signal réel simple (sin ou cos).
* Le deuxième pour effectuer la même opération mais avec un signal complexe simple comme exponentielle.
* Et le dernier pourra calculer et afficher le spectre d’une FFT inverse avec un signal complexe.

Il y aura donc des fonctions d’écoute *listener* sur chacun des boutons pour permettre au *controller* de faire lancer le traitement du signal par l’objet « FFT ».

## 4.3 Le contrôleur

Comme introduit ci-dessus, le *controller* permettra de récupérer des données afin de les transmettre d’objets en objets. Ici il permettra de faire le lien entre la fenêtre graphique « Fenetre » et l’objet de calcul « FFT ».

*Mise en situation :*

Je clique sur un bouton de la fenêtre graphique **->** le contrôleur est averti **->** il regarde le choix que la vue lui a indiquée **->** il envoie les données à l’objet « FFT » **->** il récupère les données traitées lorsque l’objet FFT notifie un changement d’état **->** il les envoie à la fenêtre graphique pour l’affichage dans un graphique.

Cela permettra entre autres une meilleure répartition du code à travers le programme.

5 Modules fonctionnels

Données transformées (Complexe[])

Choix de l’utilisateur pour la fonction

Données transformées (Complexe[])

Données de signal (Complexe[] ou Float[])