

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 |

**SOMMAIRE**

[**1 Introduction**](#_tyjcwt) **6**

[**2 Modules fonctionnels**](#_3dy6vkm) **6**

[2.1 Architecture des modules](#_1t3h5sf) 6

[2.2 Données utilisées par chaque module](#_4d34og8) 6

[2.3 Echange de données entre modules](#_2s8eyo1) 6

[**3 Structures de données**](#_17dp8vu) **6**

[3.1 Définition des structures de données](#_3rdcrjn) 6

[3.2 Action portant sur les structures de données](#_26in1rg) 7

[3.3 Visibilité des structures de données](#_lnxbz9) 8

[**4 Arbre des fonctions et flux de données**](#_35nkun2) **8**

[4.1 Arbre d’appel et flux de données](#_1ksv4uv) 8

[4.2 Description des fonctions](#_44sinio) 8

[4.2.1 FONCTIONS GLOBALES](#_2jxsxqh) 8

[4.2.2 MODULE AFFICHAGE](#_z337ya) 8

[4.2.3 MODULE MOTEUR](#_3j2qqm3) 9

[4.2.4 MODULE IA](#_1y810tw) 9

# 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 Objets fonctionnelles

## 2.1 Architecture des objets

Il sera développé deux objets différents :

* Nombre complexe
* Transformé de Fourier rapide et son inverse

## 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*

## 2.3 Echange de données entre les objets

Les objets ne s’échangent pas de données

# 3 Implémentation des objets

## 3.1 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.

## 3.2 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.

## 3.3 Visibilité des structures de données

Pour l’objet « Complexe » :

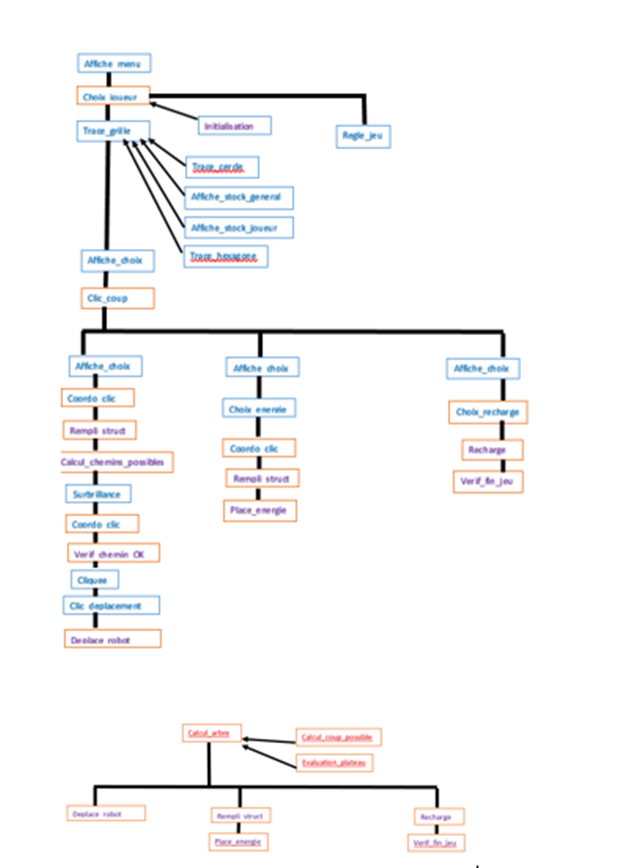
* L’imaginaire sera définie en *private*.
* Le réel sera défini en *private*.
* Le *logger* sera définie en *private.*

Pour l’objet « FFT » :

* Le tableau d’objet « Complexe » sera définie en *private*.
* Le logger sera définie en *private*.

# 4 Arbre des fonctions et flux de données

## 4.1 Arbre d’appel et flux de données



En violet, les fonctions moteur.

En bleu, les fonctions affichage.

En rouge, les fonctions IA.

Entouré en orange, partie clic.

Entouré en bleu, partie affichage.

## 4.2 Description des fonctions

Pour une description plus détaillée, veuillez vous référez au CCD[[1]](#footnote-1).

Listes des fonctions à ajouter.

1. CCD : Cahier de Conception Détaillée [↑](#footnote-ref-1)