

ECOLE HASSANIA DES TRAVAUX PUBLICS



## RAPPORT DE PROJET EN LANGAGE JAVA

Filière : Génie Électrique 1

---

### Traitement du signal

---

Réalisé par :

YACOUBI Mohamed Reda

Encadré par :

Pr.ZIATI El Houssaine

15 Octobre 2020 — 10 Janvier 2021

# Table des matières

Remerciements . . . . .	4
Résumé . . . . .	5
Introduction au traitement du signal . . . . .	6
Chapitre I : Présentation de l'interface graphique . . . . .	7
I.1 Introduction . . . . .	7
I.2 Le développement de l'interfaces graphique avec SWING . . . . .	8
Chapitre II : Traitement du signal . . . . .	10
II.1 L'échantillonnage . . . . .	10
II.2 La numérisation . . . . .	11
II.3 L'ajout du bruit . . . . .	11
Conclusion . . . . .	11

# Table des figures

1	Les classes utilisées . . . . .	7
2	Les composants de Swing . . . . .	8
3	Extrait de code de la classe InterfaceSignal . . . . .	9
4	L'interface de notre application . . . . .	10
5	La fonction permettant l'échantillonnage . . . . .	10
6	Exemple d'un signal sinusoïdal avec un pas d'échantillonnage égale à 0.1 . . . . .	10
7	La fonction permettant la numérisation . . . . .	11
8	Numérisation d'un signal sinusoïdal . . . . .	11
9	Signal sinusoïdal avant l'ajout du bruit . . . . .	11
10	Signal sinusoïdal après l'ajout du bruit . . . . .	11

# REMERCIEMENTS

Avant tout développement sur cette expérience, il apparaît opportun de commencer ce rapport de projet par des remerciements à notre professeur et encadrant Monsieur **Ziyati El Houssaine** pour nous avoir proposé un bon projet dont j'ai beaucoup appris, ainsi pour nous avoir permis d'épanouir dans ce projet par l'autonomie et la liberté dont j'ai pu disposer pour réaliser mes objectifs. Et on le remercie aussi pour son énorme soutien durant son cours d'élément de module Java qui était bien détaillé et intéressant et qui nous a permis de bien réaliser ce projet.

# RÉSUMÉ

La réalisation de ce mini projet a été une bonne occasion pour nous d'une part d'acquérir de nouvelles connaissances, et d'une part, d'assimiler les différents outils acquis durant ce semestre en matière de Java.

La principale mission du projet fut de réaliser la conception d'une application graphique capable de générer des signaux aléatoires et des signaux sinusoïdaux avec un régulateur de fréquence, Amplitude, et de phase. Ainsi que des fonctions permettant d'échantillonner et de numériser le signal généré.

# Introduction au traitement de signal

Le traitement du signal est devenu une science incontournable de nos jours : Toutes applications de mesures, de traitement d'information mettent en œuvre des techniques de traitement sur le signal pour extraire l'information désirée. Généralement on peut distinguer entre deux types de signaux : Un signal déterministe dont on peut connaître à coup sûr la valeur à chaque instant, et un signal aléatoire qui, comme son nom l'indique, varie aléatoirement en fonction du temps, en particulier sa valeur à un instant  $t$  ne peut pas être prédite. On va définir maintenant les différentes notions de base du traitement du signal.

## ↪ Un signal :

- Représentation physique d'une information à transmettre.
- Entité qui sert à véhiculer une information.

## ↪ Un bruit :

- Tout phénomène perturbateur pouvant gêner la perception ou l'interprétation d'un signal.

## ↪ Traitement du signal :

- Ensemble de techniques permettant de créer, d'analyser, de transformer les signaux en vue de leur exploitation.
- Extraction du maximum d'information utile d'un signal perturbé par le bruit.

## ↪ Signaux déterministes :

- Signaux dont l'évolution en fonction du temps  $t$  peut être parfaitement décrite grâce à une description mathématique ou graphique.

## ↪ Signaux aléatoires :

- Signaux dont l'évolution temporelle est imprévisible et dont on ne peut pas prédire la valeur à un temps  $t$ .
- La description est basée sur les propriétés statistiques des signaux (moyenne, variance, loi de probabilité, ...).

## ↪ L'échantillonnage :

- Échantillonner un signal revient à prendre un certain nombre de points régulièrement espacés de ce signal. Cette fonction consiste mathématiquement à multiplier le signal original  $S(t)$  avec un signal d'amplitude 1 à chaque instant pour lequel on prend un échantillon (opération répétée à la période  $T_{ech}$ ) et d'amplitude zéro sinon. Cette fonction est appelée peigne (ou train périodique d'impulsions) de Dirac  $\delta(t)$ .

## ↪ La numérisation :

- La numérisation consiste à transformer un signal analogique continu qui contient une quantité infinie d'amplitudes en un signal discret contenant une quantité finie de valeurs. Le passage de l'analogique au numérique repose sur trois étapes successives : l'échantillonnage, la quantification, et le codage.

# chapitre I : Présentation de l'interface

## I.1 Introduction :

Après une première analyse, on décide de modéliser l'application de la manière suivante :

- une classe **InterfaceSignal** : représentera les éléments graphiques qui serviront à créer notre interface graphique. Elle hérite la classe JFrame et implémente l'interface ActionListener.
- une classe **AleatSignal** : permet la génération du signal aléatoire. Elle hérite la classe JComponent et implémente l'interface Fonction.
- Une interface **Fonction** : contient la fonction compute permettant de calculer l'image d'une variable de type double.
- Une classe **DrawSignal** : est la classe principale et permettra d'afficher la fenêtre de notre application graphique avec la génération d'un signal aléatoire.

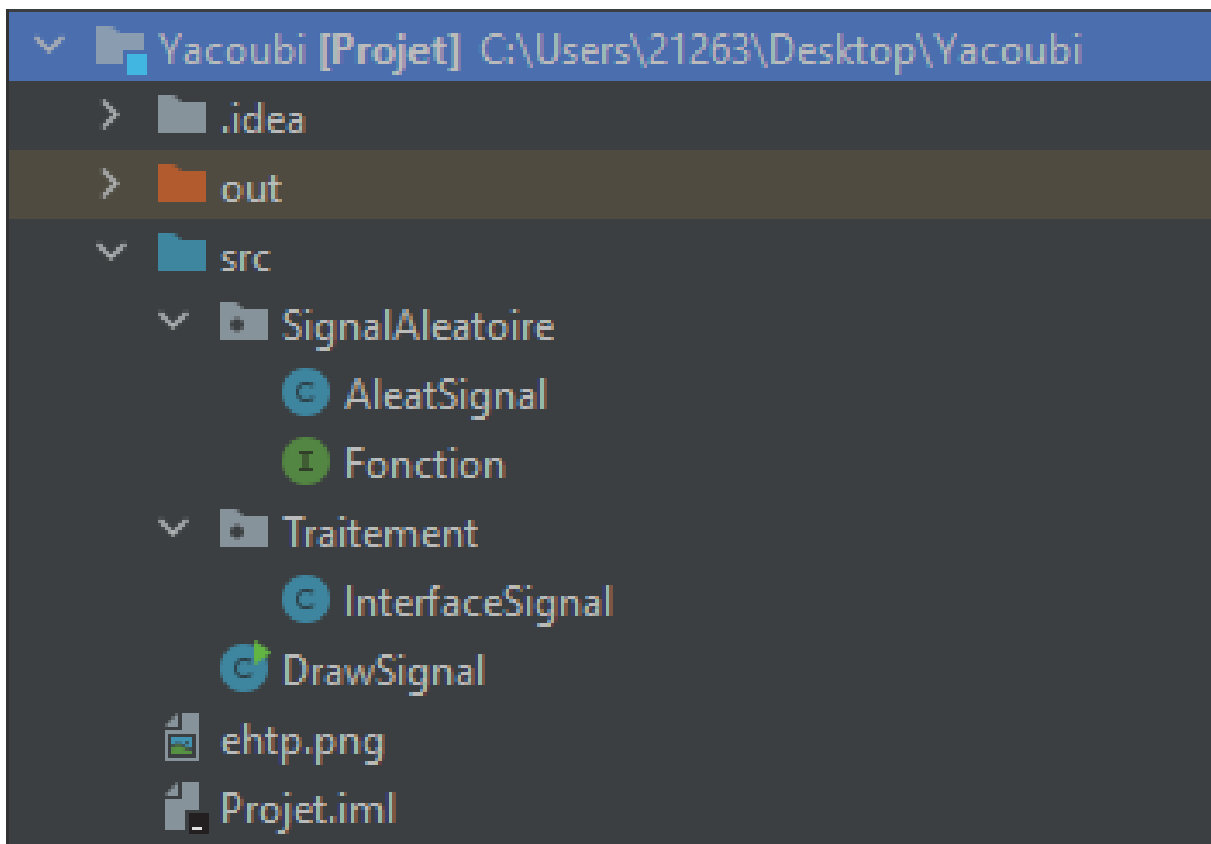


FIGURE 1 – Les classes utilisées

## I.2 L'interface graphique :

Tout au long de ce projet, nous allons utiliser le framework graphique Swing qui fait partie de la bibliothèque Java Foundation Classes (JFC). C'est une API dont le but est similaire à celui de l'API AWT mais dont les modes de fonctionnement et d'utilisation sont complètement différents.

Swing propose de nombreux composants dont certains possèdent des fonctions étendues, une utilisation des mécanismes de gestion d'événements performants et une apparence modifiable à la volée. Les composants Swing forment une nouvelle hiérarchie parallèle à celle de l'AWT. L'ancêtre de cette hiérarchie est le composant JComponent.

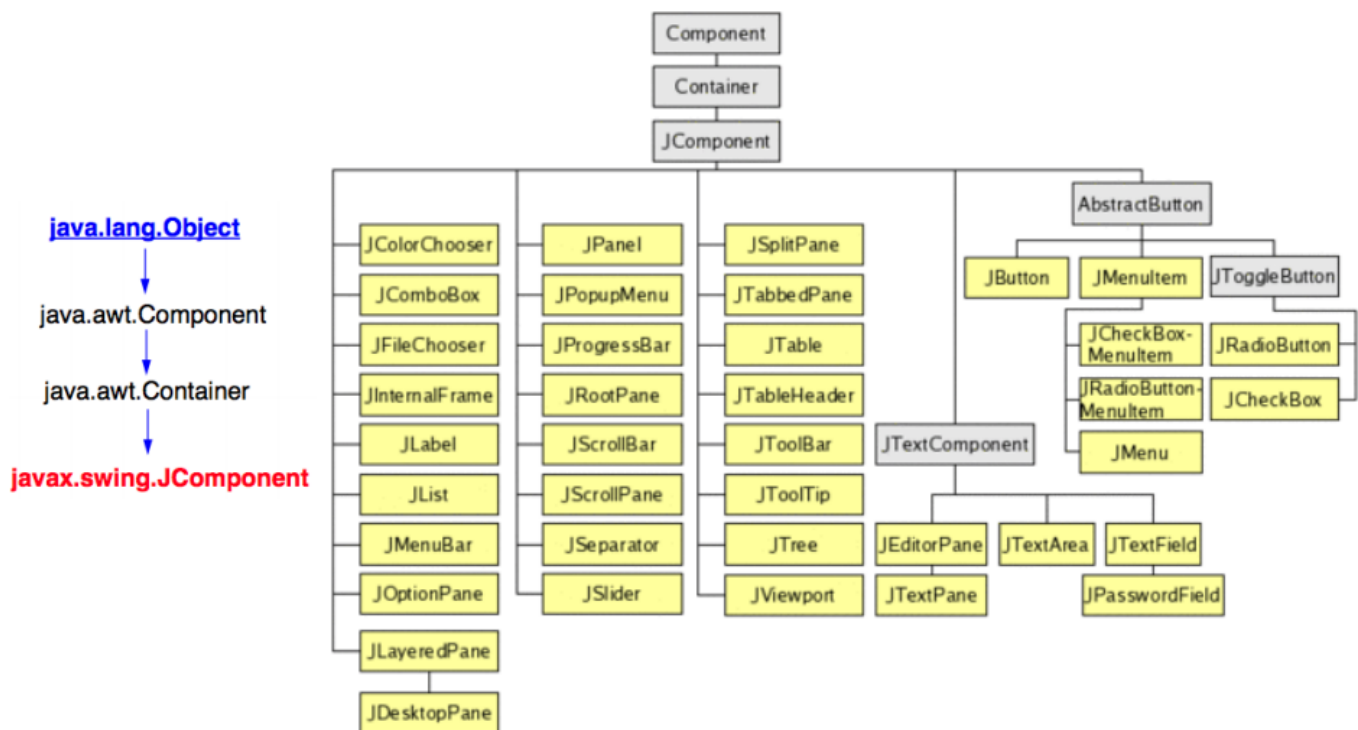
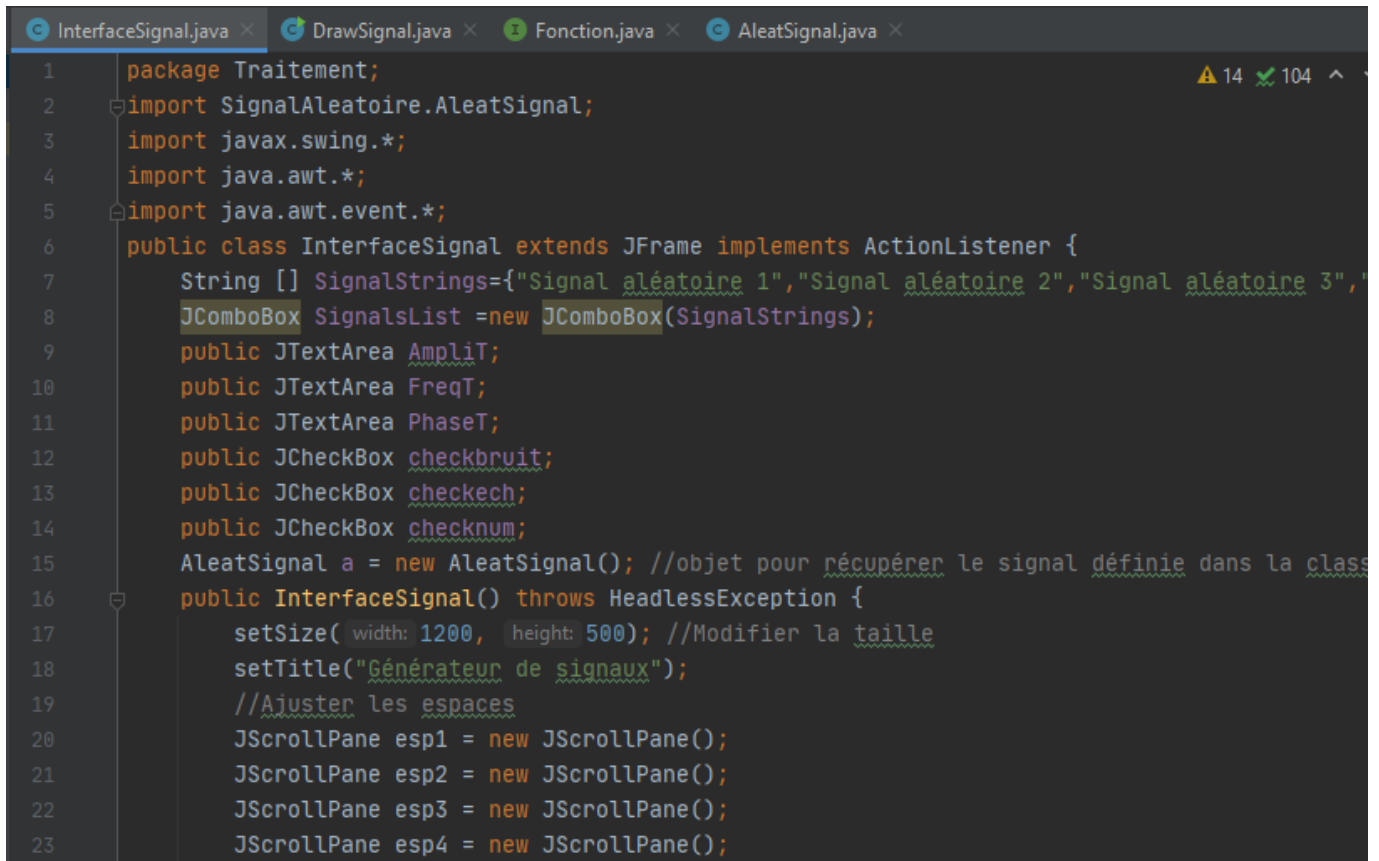


FIGURE 2 – Les composants de Swing



En effet, la classe **InterfaceSignal** est la classe permettant de développer l'interface graphique de notre application.



```
1 package Traitement;
2 import SignalAleatoire.AleatSignal;
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6 public class InterfaceSignal extends JFrame implements ActionListener {
7     String [] SignalStrings={"Signal aléatoire 1","Signal aléatoire 2","Signal aléatoire 3","
8     JComboBox SignalsList =new JComboBox(SignalStrings);
9     public JTextArea AmpliT;
10    public JTextArea FreqT;
11    public JTextArea PhaseT;
12    public JCheckBox checkbruit;
13    public JCheckBox checkech;
14    public JCheckBox checknum;
15    AleatSignal a = new AleatSignal(); //objet pour récupérer le signal définie dans la class
16    public InterfaceSignal() throws HeadlessException {
17        setSize( width: 1200, height: 500); //Modifier la taille
18        setTitle("Générateur de signaux");
19        //Ajuster les espaces
20        JScrollPane esp1 = new JScrollPane();
21        JScrollPane esp2 = new JScrollPane();
22        JScrollPane esp3 = new JScrollPane();
23        JScrollPane esp4 = new JScrollPane();
```

FIGURE 3 – Extrait de code de la classe InterfaceSignal

Au niveau de l'interface que nous devons développer, elle consistera donc en une fenêtre contenant 9 champs :

- Un champ (Textfield) pour saisir l'amplitude.
- Un champ (Textfield) pour saisir la fréquence.
- Un champ (Textfield) pour saisir la phase.
- Un bouton pour effectuer la modification.
- Un bouton pour générer le signal.
- Une liste (JComboBox) pour choisir un signal parmi les cinq signaux (dont 3 sont aléatoires et 2 sinusoïdaux)
- un checkbox pour activer ou désactiver l'échantillonnage.
- un checkbox pour activer ou désactiver la numérisation.
- un checkbox pour activer ou désactiver l'ajout du bruit.

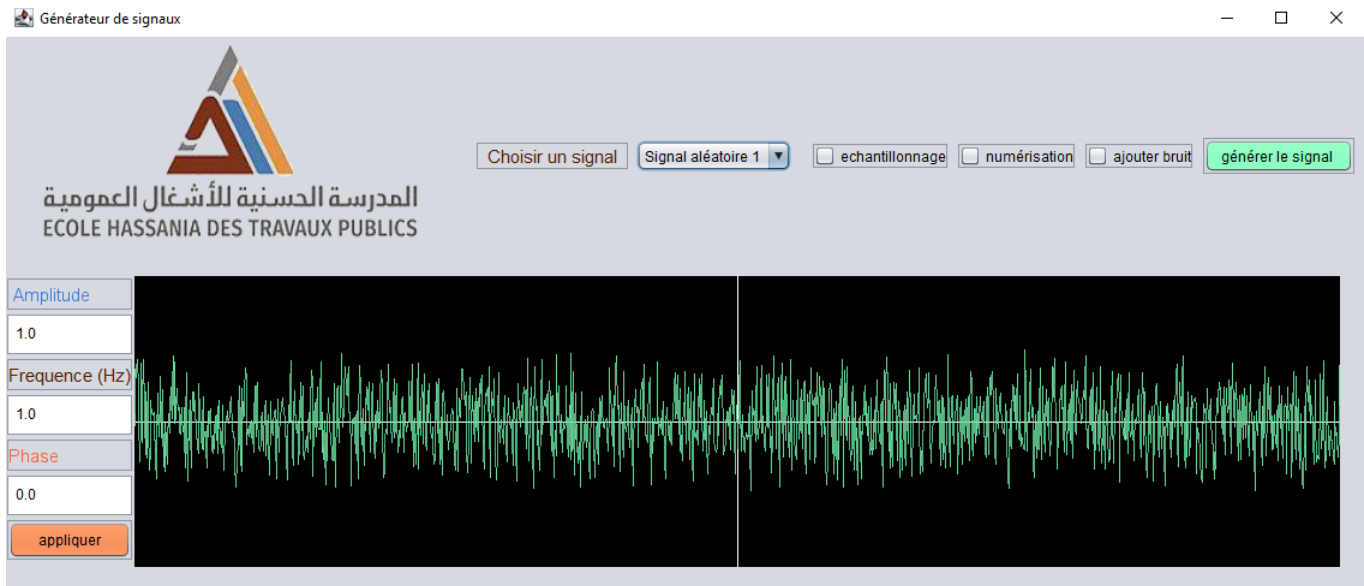


FIGURE 4 – L'interface de notre application

## Chapitre II : Traitement du signal

### II.1 L'échantillonnage :

```
if (isEch()) {
    for (double lx = -200 * getFrequence() * Math.PI + getPasech(); lx <= 200 * getFreque
        int x = xToPixel( x: lx + getPhase());
        int y = yToPixel(fonction.compute(lx) );

        g.drawLine(oldX, y: getHeight() / 2, oldX, oldY);

        oldX = x;
        oldY = y;
    }
}
```

FIGURE 5 – La fonction permettant l'échantillonnage

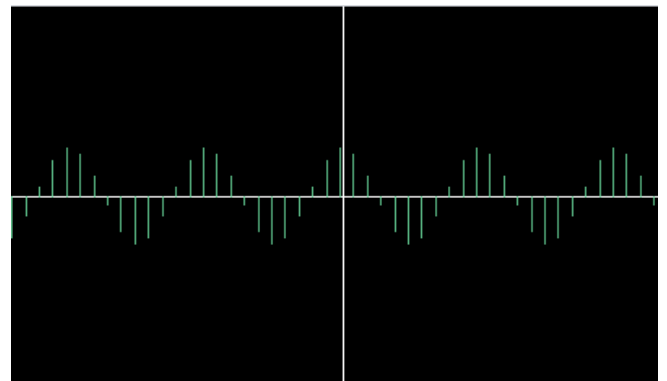


FIGURE 6 – Exemple d'un signal sinusoïdal avec un pas d'échantillonnage égale à 0.1

## II.2 La numérisation :

```
if (isNum()) {  
  
    for (double lx = -200 * getFrequence() * Math.PI + getPasech(); lx <= 200 * getFrequence() * Math.PI + getPasech(); lx += 2 * Math.PI / getFrequence()) {  
        int x = xToPixel(lx + getPhase());  
        int y = yToPixel(fonction.compute(lx));  
        int i = -oldX + x;  
        int j = y - oldY;  
        g.drawLine(x: oldX + i, y: oldY + j, x2: oldX + i, oldY);  
        g.drawLine(oldX, oldY, x2: oldX + i, oldY);  
  
        oldX = x;  
        oldY = y;  
    }  
}
```

FIGURE 7 – La fonction permettant la numérisisation

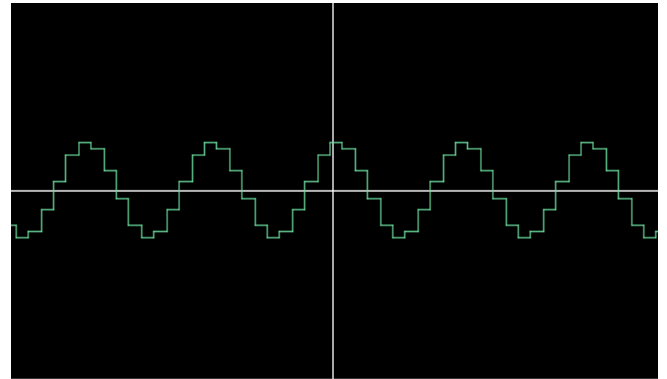


FIGURE 8 – Numérisation d'un signal sinusoïdal

## II.3 L'ajout du bruit :

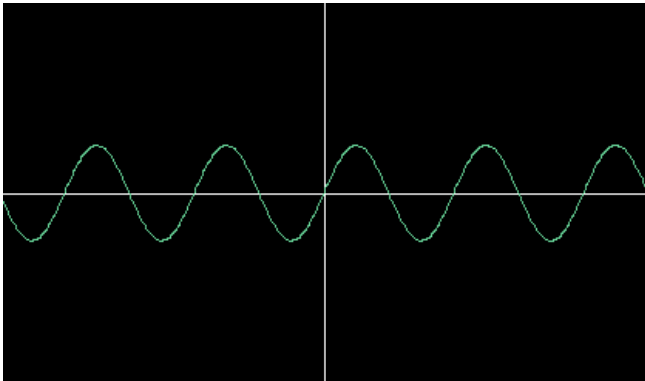


FIGURE 9 – Signal sinusoïdal avant l'ajout du bruit

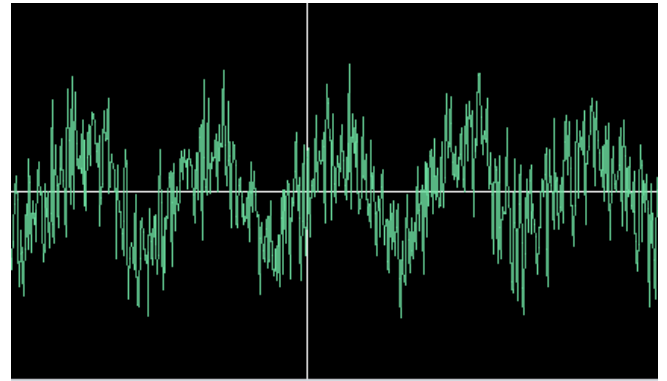


FIGURE 10 – Signal sinusoïdal après l'ajout du bruit

## Conclusion :

Ce projet nous a permis d'acquérir les compétences suivantes :

- Utiliser les techniques paradigmes orientés objet.
- Maîtriser la programmation avec le langage Java.
- Se familiariser avec les interfaces graphiques et le Framework Swing.