ECHODIA / POLYTECH CLERMONT-FERRAND

Implantation d'une nouvelle fonctionnalité dans le logiciel Echosoft.

Charles ONDET

5ème année

GE : 2014

Sommaire

La société ECHODIA est un équipementier clermontois dans le domaine du diagnostic médical ORL et Neurologie. Cette entreprise développe et assemble du matériel électronique de dépistage auditif. ECHODIA commercialise un appareil de dépistage auditif appelé ELIOS. Ce dernier permet de réaliser des mesures de Potentiels Evoqués Auditifs (PEA), signaux témoins de la santé du nerf auditif du patient.

Afin d'exploiter et analyser les signaux PEA, l'entreprise a développé un logiciel appelé ECHOSOFT. Ce logiciel est réalisé sous l'espace de développement NETBEANS utilisant le langage JAVA.

Les développeurs de la société ont conçu le logiciel par un ensemble d'applications structurées répondant chacune à un point précis.

Le sujet de cette note d'application consiste à implanter une nouvelle fonctionnalité dans le logiciel ECHOSOFT.

1. Présentation du développement du logiciel dans l'espace NETBEANS

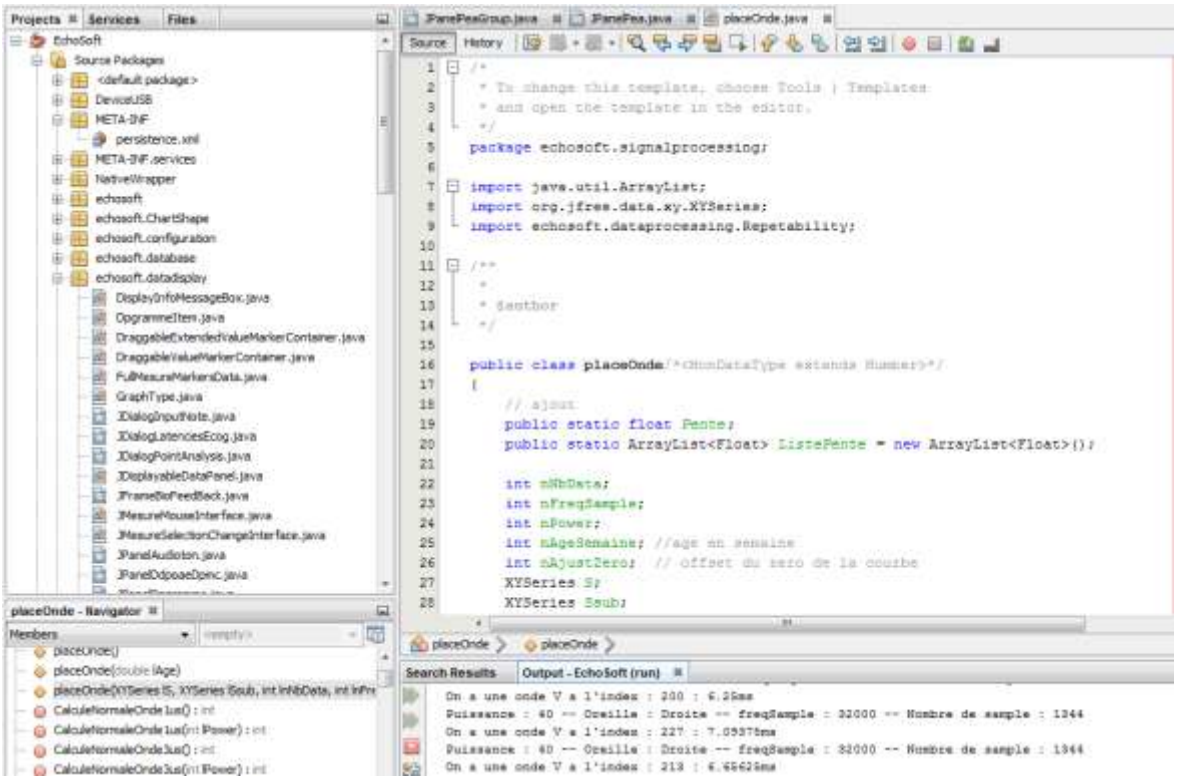

Figure 1 : Espace de développement NetBeans.

NetBeans est l'espace de développement du logiciel ECHOSOFT. Le langage Java est utilisé. La fenêtre, en haut à gauche de la Figure 1 permet de sélectionner les fichiers de code à modifier, ou de créer de nouveaux fichiers. La fenêtre centrale est celle de la programmation. En bas à droite, la fenêtre output permet l'affichage de résultats et donc la mise au point des programmes.

2. Arborescence des applications

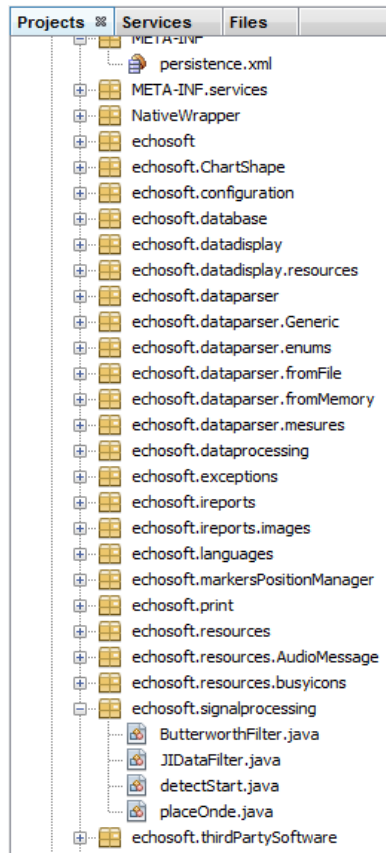

Figure 2 : Structure arborescente des applications.

Le logiciel ECHOSOFT est construit à partir de fichiers spécifiques et ordonnés. Un fichier est nommé de manière claire renvoyant à une spécificité précise du logiciel.

La Figure 2 représente cette arborescence.

3. La programmation en JAVA des fonctionnalités

3.1 Les classes

En langage java, les classes sont des structures permettant le rangement et l'utilisation de données.

```
public class DonneeTriee {
    XYSeries serie;
    String oreille;
    int power;
    int ondeV;
    float PenteV;

    public XYSeries getserie()  {
        return serie;
    }

    public void setserie(XYSeries new_value) {
        serie = new_value;
    }
}
```

Accesseur

Mutateur

Figure 3 : Exemple de classe JAVA.

La Figure 3 est un exemple de définition d'une classe.

Il existe différentes manières d'accéder aux données.

```
// Remplissage du tableau "Tableau" d'objet de type DonneeTriee
DonneeTriee mesure = new DonneeTriee(); // Declaration d'un objet mesure de type DonneeTriee
mesure.setserie(serie); // Copie de la serie de type XYseries dans l'element serie de mesure
mesure.power = power; // Ecriture de power de type int dans l'element power de mesure
mesure.oreille = oreille.toString(); // Ecriture d'oreille de type string dans l'element oreille de mesure
```

Figure 4 : Ecriture d'une classe.

La Figure 4 donne un exemple de déclaration et d'écriture pour une classe. La première ligne de code est une déclaration du nouvel objet, dans l'exemple, une variable mesure de type *DonneeTriee* est déclarée. La deuxième ligne permet l'enregistrement de *serie* de type *DonneeTriee* dans le champ correspondant de la variable mesure. La procédure (fonction) *setserie()* est utilisée dans ce cas, la troisième et quatrième utilisent un accès direct : *mesure.power* et *mesure.oreille* pour transférer respectivement un entier et une chaîne de caractères.

La Figure 5 indique la façon d'accéder aux données. Dans cet exemple, le tableau nommé *Tableau* contient des données de type *DonneeTriee*. La fonction *get()* permet l'accès à un élément de tableau et la fonction *floatValue()* permet de convertir la donnée en type *float*.

```
S1[i] = (Tableau.get(j).serie.getY(i).floatValue());
Res.ondeV1 = Tableau.get(j).ondeV;
```

Figure 5 : Accès aux données d'une classe.

3.2 Les procédures

Les procédures JAVA sont des fonctions prenant des variables de différents types en entrée, et retournant une autre variable en sortie.

```
int nomVariableInt = 5; // Variable de type int initialisée à 5
float nomVariableFloat; // Variable de type float
public static float nomStatFloat; // Variable de type float statique

XYSeries nomSerie; // Déclaration d'une série XY

int nbData = 10;
float tableauFloat[] = new float[nbData]; // Déclaration d'un tableau de float de taille nbData
ArrayList<Float> listeFloat = new ArrayList<Float>(); // Déclaration d'un vecteur de float
```

Figure 6 : Exemples de déclaration de variables.

La Figure 6 indique différentes déclarations de variables pouvant être utilisées dans une procédure. Les types int, float correspondent respectivement à des types entier et flottant. Une XYSeries est un objet tableau à deux dimensions X et Y. Un vecteur est une liste, c'est-à-dire un tableau dont la taille peut évoluer au cours des traitements de données. Les instructions suivantes peuvent être utilisées : listeFloat.add, listeFloat.remove, pour ajouter ou retirer un élément. La taille est retournée par la fonction listeFloat.size.

```
public placeOnde(double lAge){
    nAgeSemaine = (int) (39.0 + (lAge * 52.0)); // + 9mois grossesse
    if (nAgeSemaine < 35)
    {
        nAgeSemaine = 35;
    }
}
```

Figure 7 : Exemple de procédure JAVA.

La Figure 7 est un exemple de procédure qui prend en entrée la variable lAge (âge en années) et affecte à la variable globale nAgeSemaine l'âge correspondant en semaine.

Cette fonction est déclarée en public, c'est-à-dire qu'elle peut être utilisée de manière libre dans la classe.

```
public float PuissanceCarree(float x) {

    return x*x;
}
```

Figure 8 : Autre exemple.

La fonction présente en figure 8 calcule et retourne le carré de la valeur d'entrée.

Les procédures peuvent être déclarées et utilisées dans une classe.

3.3 Lien entre les applications

Il est parfois nécessaire de récupérer des données d'une autre application. Pour cela il est possible d'importer les données calculées dans un fichier dans un autre. Cela peut être utile pour regrouper des fonctions mathématiques et les utiliser dans d'autres fichiers.

Pour lier les fichiers, il faut les importer comme il est indiqué sur les Figures 9 et 10.

```
import org.jfree.data.xy.XYDataItem;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
```

Figure 9 : Exemple d'import de bibliothèques.

```
import echosoft.datadisplay.JPanelPea;
import echosoft.dataprocessing.Repetability;
```

Figure 10 : Exemple d'import de fichiers.