Le programme se repose sur 3 grandes partie : l'interface, l'éventualité de gérer les erreurs ainsi que la classe qui calcule les valeurs tapés par l'utilisateur.

Il existe 3 classes : « **MainPrincipale** » le main Principale pour le lancement du programme, la classe « **Programme** » pour gérer les mises en forme de l'interface graphique et « **Calculer** » pour gérer l'éventualité des erreurs et les calculs à opérer.

INTERFACE :

Premièrement, l'interface est généralement composer d'une **JLabel** afficheur et 20 Boutons (**JButton**).

JLabel : C'est l'afficheur des opérations taper par l'utilisateur(sous forme de **String**) ainsi que le résultat de l'opération (erreur s'il existe).

Le programme se repose sur un Container principale qui est**« contenu »** pour gérer **le layout principale.**
Chaque partie des **JBoutton(Bouton), JLabel** seront superposés à cette conteneur.

« **afficheur.setBounds(10,10,365,50)**» c'est le réglage des dimensions de la fenêtre du Jlabel dont **10** c'est l'espace du début de gauche vers la droite, **10** distance par rapport à la hauteur venant de haut vers bas, **365** de longueur et **50** sa largeur.

«**afficheur.setHorizontalAlignment(SwingConstants.CENTER)**» ce code nous permet de centrer les textes dans le Jlabel.

«**contenu.add(afficheur)**» »Ici on parle d'ajouter au conteneur l'afficheur Jlabel.

Les 20 boutons : Ce sont les touches intégrer dans l'application qui sont : ("1","2","3","4","5","6","7","8","9","+","-","*","(",")","0","/",".","C","S","=").

On a créé des panneaux pour placer ces boutons.

« **JPanel panneau=new JPanel()** » sont les panneau où se places les bouttons.

Dans notre programme, le **JPanel** (**panneau**) ici est de layout « **GridLayout** » permettant ainsi d'inserer les boutons sans se soucier de l'emplacement exacte x,y dans le container «**contenu**» en prenant juste la layout du **JPanel**.

«**panneau.setLayout(new GridLayout(5,4,10,10))**» "**GridLayout**'' aligne automatiquement les contenues dans son Layout sous forme **(x,y,z,h)** dont **x** nombre de ligne, **y** nombre de colonne, **z** espace entre l'horizontale et **h** espace entre la verticalité.

«**contenu.add(panneau)**» c'est dans le container qu'on place les panneaux.

On a mis un écouteur de clic (d'appuiyage) sur les boutons par le code « **boutons[i].addActionListener(this)**» permettant ainsi de récupérer la valeur du boutton cliquer par l'utilisateur.

Ainsi l'écouteur écouteur d'appuie sur le clavier «**boutons[i].addKeyListener(this)**».

« this » pour préciser qu'on fait l'écoute sur la classe actuelle.

Pour finir, on a attribué son rôle à chaque bouton ;

```
}
public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==boutons[19])//Boutons égal.
    {
        valeurStaticCalculeDeJtexField=resultat_Calcul(valeurStaticDansJtextField);
        valeurStaticDansJtextField=valeurStaticCalculeDeJtexField;
        afficheur.setText(valeurStaticCalculeDeJtexField);
    }
    else if(e.getSource()==boutons[17])//Boutton C(Supprimer tout).
    {
        afficheur.setText("");
        valeurStaticDansJtextField="";
        valeurStaticCalculeDeJtexField="";
    }
    else if(e.getSource()==boutons[18])//Boutton S(Supprimer).
    {
        valeurStaticDansJtextField=enleverUneCaractereParDerriere(valeurStaticDansJtextField);
        afficheur.setText(valeurStaticDansJtextField);
    }
    else
    {
            valeurStaticDansJtextField+=e.getActionCommand();
            afficheur.setText(valeurStaticDansJtextField);
    }
}
```

e c'est l'évènement qui appelle tout les boutons appuyer.

Ainsi on a placer des conditions pour attribuer leurs rôles comme le bouton [19] défini comme le bouton égal affichant les résultats, le bouton [17] ou C défini comme la touche supprimer tout vidant le JLabel, et le bouton [18] ou S comme étant le bouton supprimer tout court.

ERREUR :

On a établie chaque condition d'une éventualité d'erreur dans des méthodes comme

siExisteCaractereNonAutoriseDansLentrer, siLaFinEstOperandeNonAutorise, siEntrerEstVide, siLeDebutEstOperandeNonAutorise, siExisteDeuxCaractereSuccessifOpperandeNonAutorise, siExisteDivisionParZero, siExisteParentheseVideInterieure, siExisteParentheseNonFerme, siExisteParentheseIncorrecte, siExisteParentheseMalPlace, siExisteDeuxPointSuccessif

Si l'erreur présente **LaFinEstOperandeNonAutorise ou LeDebutEstOperandeNonAutorise ou ExisteDeuxCaractereSuccessifOpperandeNonAutorise ou siExisteParentheseVideInterieure ou ExisteParentheseNonFerme ou ExisteParentheseIncorrecte ou ExisteParentheseMalPlace ou ExisteDeuxPointSuccessif**, dans ces cas-là le JLabel affiche ''Syntax Error'' et à part ça donnera comme résultat ''Math Error''.

Tout ces méthodes sont globalisés dans la méthode « **retourneErreur** » avec comme **entrer**, la valeur tapé par l'utilisateur qui sont des **String**.

```
private String retourneErreur(String entrer)
{
if(siLaFinEstOperandeNonAutorise(entrer)||
    siLeDebutEstOperandeNonAutorise(entrer)||
    siExisteDeuxCaractereSuccessifOpperandeNonAutorise(entrer)||
    siExisteParentheseVideInterieure(entrer)||
    siExisteParentheseNonFerme(entrer)||
    siExisteParentheseIncorrecte(entrer)||
    siExisteParentheseMalPlace(entrer)||
    siExisteDeuxPointSuccessif(entrer))
return "Syntax Error";
else
return "Math Error";
}
```

OPERATION :

L'opération s'effectue sous l'ordre de priorisation mathématique mais par contre, le programme enlève d'abord toute équivalence de multiplication sur les parenthèses.

Le reste du code est stipulé dans le code source avec des commentaires pour plus de clartés.

Comme

2(3+9) devenait « 2*(3+9) »

(2)(3) devenait « 2*3»

```
private String enleverOperationParenthese(String entrer)
{
    entrer=supprimerUneChiffreAvecParentheseOuverte(entrer);
    entrer=supprimerUneParentheseFermeAvecUneOuverte(entrer);//Attribuer donc l'operation de multiplication.
    String recupereValeurSorti=entrer,recupererValeurEntreParenthese="";
```

Ensuite, on précède à enlever la division, multiplication, l'addition et la soustraction.

```
private String enleverToutOperation(String entrer)
{
    //L'operation doit être d'abbord passer par la suppression des parenthèses(ci-dessous).
    entrer=sortieFinalOperationDividanteEtMultiplicationAvecAdditionEtSoustraction(entrer);
    //La calculatrice suit les règles de priorisations sur les opérations.
    String calculFinalSortie=enleverOperation(entrer,'/');
    calculFinalSortie=enleverOperation(calculFinalSortie,'*');
    calculFinalSortie=enleverOperation(calculFinalSortie,'+');
    calculFinalSortie=enleverOperation(calculFinalSortie,'-');
    return enleverPlusEtMoinsAuDebutNombre(calculFinalSortie);
}
```

<u>Les opérateurs ***, /, +, -**</u> :

Pour la multiplication *

```
if(operateur=='*')
{
        for(int i=0;i<entrer.length();i++)
        {
}
else if(operateur=='/')
{
        for(int i=0;i<entrer.length();i++)
        {
            if(entrer.charAt(i)=='/')
            {
                double c=Double.parseDouble(retourneValeurNombrePourMultiplicationEtDivision(entrer,true,i))/Double.parseDouble(retourneValeurNombrePour
                String finalAfficher=(enleverExponentielleEtForcerSortiPointsurDouble(c));
                apresResultat=prendreResteCaractere(entrer,true,i-retourneValeurNombrePourMultiplicationEtDivision(entrer,true,i).length())+finalAfficher
                return enleverOperation(apresResultat,'/');
            }
```

Pour la division /

```
}
else if(operateur=='+')
{
        if(entrer.charAt(0)=='+'||entrer.charAt(0)=='-')
    debut=1;
    else
    debut=0;
        for(int i=debut;i<entrer.length();i++)
        {
            if(entrer.charAt(i)=='+')
            {
                double c=Double.parseDouble(retourneValeurNombrePourAdditionEtSoustraction(entrer,true,i))+Double.parseDouble(retourneValeurNombrePourAd
                String finalAfficher=(enleverExponentielleEtForcerSortiPointsurDouble(c));
                apresResultat=prendreResteCaractere(entrer,true,i-retourneValeurNombrePourAdditionEtSoustraction(entrer,true,i).length())+forcerAffichag
                return enleverOperation(apresResultat,'+');
            }
        }
```

Pour l’addition +

```
}
else
{
    //L'opérateur est donc '-'.
        if(entrer.charAt(0)=='+'||entrer.charAt(0)=='-')
    debut=1;
    else
    debut=0;
        for(int i=debut;i<entrer.length();i++)
        {
            if(entrer.charAt(i)=='-')
            {
                double c=Double.parseDouble(retourneValeurNombrePourAdditionEtSoustraction(entrer,true,i))-Double.parseDouble(retourneValeurNombrePou
                String finalAfficher=(enleverExponentielleEtForcerSortiPointsurDouble(c));
                apresResultat=prendreResteCaractere(entrer,true,i-retourneValeurNombrePourAdditionEtSoustraction(entrer,true,i).length())+forcerAffic
                return enleverOperation(apresResultat,'-');
            }
        }
}
return apresResultat;
```

Pour la soustraction -