Cycle d’ingénieur

Logiciels & Systèmes Intelligents

Rapport final du projet de fin du demi-module de la programmation mobile concernant la réalisation d’une :

Calculatrice scientifique

Réalisé par : Samir AIT-ABBOU

Nassima EL JAZOULI

Encadré par : Monsieur Hassan ZILI

Remerciements :

C’est avec toute nos profondes gratitudes que nous exprimons nos sincères remerciements à notre cher professeur du demi-module de la programmation mobile Monsieur Hassan ZILI d’avoir d’abord nous donner l’opportunité de découvrir le monde de la programmation mobile avec ses deux types native et hybride et nous permettre de découvrir l’Android le système d’exploitation mobile le plus dominant dans le monde, ce qui va enrichir notre expérience universitaire, ainsi d’avoir nous diriger dans ce projet de fin de demi-module. Sans sa disponibilité, sa patience, les conseils qu’il nous a prodigués tout au long de ce semestre, ce travail n’aurait pas pu, sans doute, être mené à son terme.

Table des matières :

Introduction :

Android est un système d'exploitation mobile fondé sur le noyau Linux et développé par Google. Lancé en juin 2007 à la suite du rachat par Google en 2005 de la startup du même nom, le système avait d'abord été conçu pour les smartphones et tablettes tactiles, puis s'est diversifié dans les objets connectés et ordinateurs comme les télévisions (Android TV), les voitures (Android Auto), les Chromebook (Chrome OS qui utilise les applications Android) et les smartwatch (Wear OS). En 2015, Android est le système d'exploitation mobile le plus utilisé dans le monde, devant iOS d'Apple, avec plus de 80 % de parts de marché dans les smartphones pour l'ensemble de ses versions et adaptations.

L'objectif du projet de ce demi-module c'est la réalisation d'une calculatrice scientifique qui respecte une certaine interface graphique à l'aide de l'environnement de développement Android Studio.

A. L'environnement de développement :

a. Android studio :

Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle. Il peut être téléchargé sous les systèmes d'exploitation Windows, macOS, Chrome OS et Linux.

Avant Android Studio, de 2009 à 2014, Google propose comme environnement de développement officiel une distribution spécifique de l'environnement Eclipse, contenant notamment le SDK d'Android. Android Studio est annoncé le 15 mai 2013 lors du Google I/O et une version Early Access Preview est disponible le jour même. Le 8 décembre 2014, Android Studio passe de version bêta à version stable 1.0. L'environnement devient alors conseillé par Google, et Eclipse est délaissé.

b. L'Emulateur :

Android studio contient un émulateur intégré qui peut simuler dans le cas de développement mobile un téléphone ou une tablette qu'on peut soit conserver l'émulateur déjà installé lors de l'installation ou personnaliser notre propre émulateur selon les capacités de notre device.

L’émulateur avec lequel on travaille est présenté dans la figure suivante :

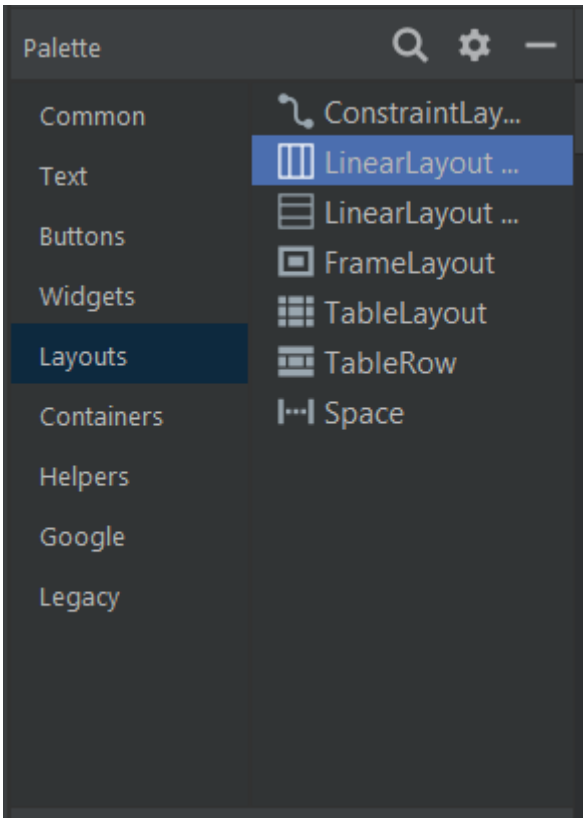
B.L’interface graphique :

a. Manipulation des Layouts :

La création d’un élément visible ou bien comme il est intitulé view demande l’existence d’au moins deux contraintes, mais ça devient très complexe surtout avec un grand nombre de view. Dans notre cas les boutons qui désignent ces views, d’où la nécessité d’un outil plus efficace que la méthode traditionnelle qui est la manipulation des Layouts qui était le sujet de notre deuxième séance dans ce demi module.

Il s’agit de la mise en page qui définit la structure d’une interface utilisateur dans notre application.

Les figures vont mieux expliquer de quoi s'agit-t-il exactement :



On a d'abord choisi un LinerarLayout vertical avec une hauteur wrap content, puis on a met en dedans de ce dernier 5 LinerarLayout horizental puisque on doit construire 20 boutons en total avec 4 boutons dans chacun.

b. Le fichier strings.xml :

Après on a passé à ajuster les noms de chaque bouton à travers le fichier **strings.xml** qui se trouve dans le package **values** qui se trouve lui aussi dans les ressources root **res** qui se trouve dans le java module **app**.

La figure suivante correspond au fichier **strings.xml** :

```xml
<resources>
    <string name="app_name">MyCalculator</string>
    <string name="afficher">Afficher</string>
    <string name="button">Button</string>

    <string name="zero">0</string>
    <string name="one">1</string>
    <string name="two">2</string>
    <string name="three">3</string>
    <string name="four">4</string>
    <string name="five">5</string>
    <string name="six">6</string>
    <string name="seven">7</string>
    <string name="eight">8</string>
    <string name="nine">9</string>

    <string name="addition">+</string>
    <string name="soustraction">-</string>
    <string name="produit">x</string>
    <string name="division">/</string>

    <string name="inverse">1/x</string>
    <string name="cosinus">Cos</string>
    <string name="sinus">Sin</string>
    <string name="tangence">tan</string>

    <string name="egal">=</string>
    <string name="clear">CE</string>

    <string name="ecran">Entrez une valeur</string>
</resources>
```

Après on a passé à remplir l'attribut text de chaque bouton avec **@string/** suivi du nom de l'élément dans le fichier **strings.xml** comme c'est montré dans ce


```xml
<string name="inverse">1/x</string>
```

bouton choisi par hasard :

c. L'écran des opérations :

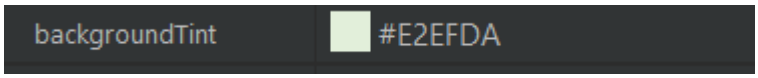
Dans le cadre de respecter l'illustration présentée avec l'énoncé on a passé à ajouter un **TextView** dont on va écrire nos opérations et afficher nos résultats par la suite.

La figure qui suit est une capture de mon écran dont je vais taper les opérations

et obtenir les résultats :

d. La modification des couleurs :

Dans l'énoncé une palette de couleur était demander de la respecter, c'est qu'on a essayé de faire à travers l'extraction du RGB des couleurs utilisées, et les modifier à travers la propriété : **backgroundTint**.



Tous ces différents étapes ont nous permis d'obtenir le résultat suivant :

C. Le fonctionnement de la calculatrice à l'aide de Java :

a. Les events :

On a attribué à chaque bouton le nom d'un event : une fonction qu'on va la définir par la suite dans le fichier **MainActivity.java** comme c'est montré dans la figure suivante :

De la même manière comme pour le bouton **1/X** on a fait pour tous les autres fonctions, et bien sûr on créer en parallèle les fonctions correspondantes dans **MainActivity.java**. Ces fonctions sont jusqu'à maintenant vides.

b. Les chiffres :

L'étape suivante concerne les chiffres, et le fait de les afficher sur l'écran si quelqu'un tape le bouton concerné à travers la fonction **updateText (String strToAdd)** comme c'est montré dans la figure suivante :

```
public void btn0(View view) { updateText( strToAdd: "0"); }
public void btn1(View view) { updateText( strToAdd: "1"); }
public void btn2(View view) { updateText( strToAdd: "2"); }
public void btn3(View view) { updateText( strToAdd: "3"); }
public void btn4(View view) { updateText( strToAdd: "4"); }
public void btn5(View view) { updateText( strToAdd: "5"); }
public void btn6(View view) { updateText( strToAdd: "6"); }
public void btn7(View view) { updateText( strToAdd: "7"); }
public void btn8(View view) { updateText( strToAdd: "8"); }
public void btn9(View view) { updateText( strToAdd: "9"); }
```

c. La fonction updateText (String strToAdd) :

La fonction **updateText (String strToAdd)** permet de modifier ce qui est affiché sur l'écran à chaque fois qu'on tape un bouton dont cette fonction est appelée.

La figure suivante concerne le code de cette fonction :

```java
public void updateText(String strToAdd)
{
    String oldStr = ecran.getText().toString();
    int cursorPos = ecran.getSelectionStart();
    String leftStr = oldStr.substring(0, cursorPos);
    String rightStr = oldStr.substring(cursorPos);
    if("Entrez une valeur".equals(ecran.getText().toString()))
    {
        ecran.setText(strToAdd);
        ecran.setSelection(cursorPos + 1);
    }
    else
    {
        ecran.setText(String.format("%s%s%s", leftStr, strToAdd, rightStr));
        ecran.setSelection(cursorPos + 1);
    }
}
```

d. La fonction principale onCreate (Bundle savedInstanceState) :

Le code qu'on a ajouté dans cette fonction nous a permet d'afficher les valeurs sur notre écran, au plus de cela, la fonction **onClick (View v)** fait l'objet de vider l'écran une fois qu'on met le curseur sur l'écran des opérations et de l'affichage.

La figure suivante concerne le code de cette fonction principale :

```java
public class MainActivity extends AppCompatActivity {
    private EditText ecran;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ecran = findViewById(R.id.Ecran);
        ecran.setShowSoftInputOnFocus(false);
        ecran.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if ("Entrez une valeur".equals(ecran.getText().toString()))
                {
                    ecran.setText("");
                }
            }
        });
    }
```

e. Les fonctions mathématiques :

Avant d'implémenter le code des fonctions mathématiques, il faut d'abord télécharger la librairie mathématique et l'ajouter, ce qui n'est pas intégrée par défaut.

Après l'implémentation du code des chiffres, il nous reste 10 boutons à implémenter leurs codes, à partir :

- Les opérations élémentaires **(+,-,x,/)**.
- Le bouton égal **(=)** qui affiche le résultat.
- Le bouton **CE** qui vide l'écran.
- Les fonctions trigonométriques **(Sin, cos et tan)**.
- La fonction inverse **1/X**.

Commençons d'abord avec le bouton **CE** qu'il suffit de remplacer qu'il que soit ce qui est affiché sur l'écran par une chaine de caractère vide.

```
public void btnclear(View view){
    ecran.setText("");
}
```

Passons maintenant aux fonctions élémentaires **(+,-,x,/)**, qu'il suffit tout simplement de les implémenter par la fonction **updateText (String strToAdd)** et la fonction égal par la suite va continuer le travail sur ces fonctions.

```
public void btnaddition(View view) { updateText( strToAdd: "+"); }
public void btnsoustraction(View view) { updateText( strToAdd: "-"); }
public void btnproduit(View view) { updateText( strToAdd: "*"); }
public void btndivision(View view) { updateText( strToAdd: "/"); }
```

La fonction égal prend l'opération écrite sur l'écran et la calcule et retourne le résultat final :

La figure suivante concerne le code du bouton égal :

```
public void btnegal(View view){
    String userExp = ecran.getText().toString();
    //userExp.replaceAll("x", "*");
    Expression exp = new Expression(userExp);
    String result = String.valueOf(exp.calculate());
    ecran.setText(result);
    ecran.setSelection(result.length());
}
```

Passons maintenant au code des fonctions trigonométriques **(Sin, cos et tan)** et de la fonction inverse **1/X**. La particularité de ces quatre fonctions repose sur le fait que cette calculatrice scientifique ne contient pas des parenthèses, pour cela on est obligé de taper la valeur sur l'écran avant de choisir la fonction qu'on veut, et le résultat s'affiche directement sans avoir besoin de cliquer le bouton (=).

Le code suivant concerne la fonction sinus :

```
public void btnsinus(View view){
    Editable valeur = ecran.getText();
    if (valeur.equals(""))
    {
        ecran.setText("Error");
    }
    else
    {
        double degrees = Double.parseDouble(String.valueOf(ecran.getText()));
        if (degrees<0 || degrees>360)
        {
            ecran.setText("Error");
        }
        else
        {
            double radians = Math.toRadians(degrees);
            double sinValue = Math.sin(radians);
            ecran.setText(Double.toString(sinValue));
        }
    }
}
```

Le code suivant concerne la fonction cosinus :

```
public void btnconsinus(View view){
    Editable valeur = ecran.getText();
    if (valeur.equals(""))
    {
        ecran.setText("Error");
    }
    else
    {
        double degrees = Double.parseDouble(String.valueOf(ecran.getText()));
        if (degrees<0 || degrees>360)
        {
            ecran.setText("Error");
        }
        else
        {
            double radians = Math.toRadians(degrees);
            double cosValue = Math.cos(radians);
            ecran.setText(Double.toString(cosValue));
        }
    }
}
```

Le code suivant concerne la fonction tangente :

```
public void btntangence(View view){
    Editable valeur = ecran.getText();
    if (valeur.equals(""))
    {
        ecran.setText("Error");
    }
    else
    {
        double degrees = Double.parseDouble(String.valueOf(ecran.getText()));
        if (degrees<0 || degrees>360)
        {
            ecran.setText("Error");
        }
        else
        {
            double radians = Math.toRadians(degrees);
            double tanValue = Math.tan(radians);
            ecran.setText(Double.toString(tanValue));
        }
    }
}
```

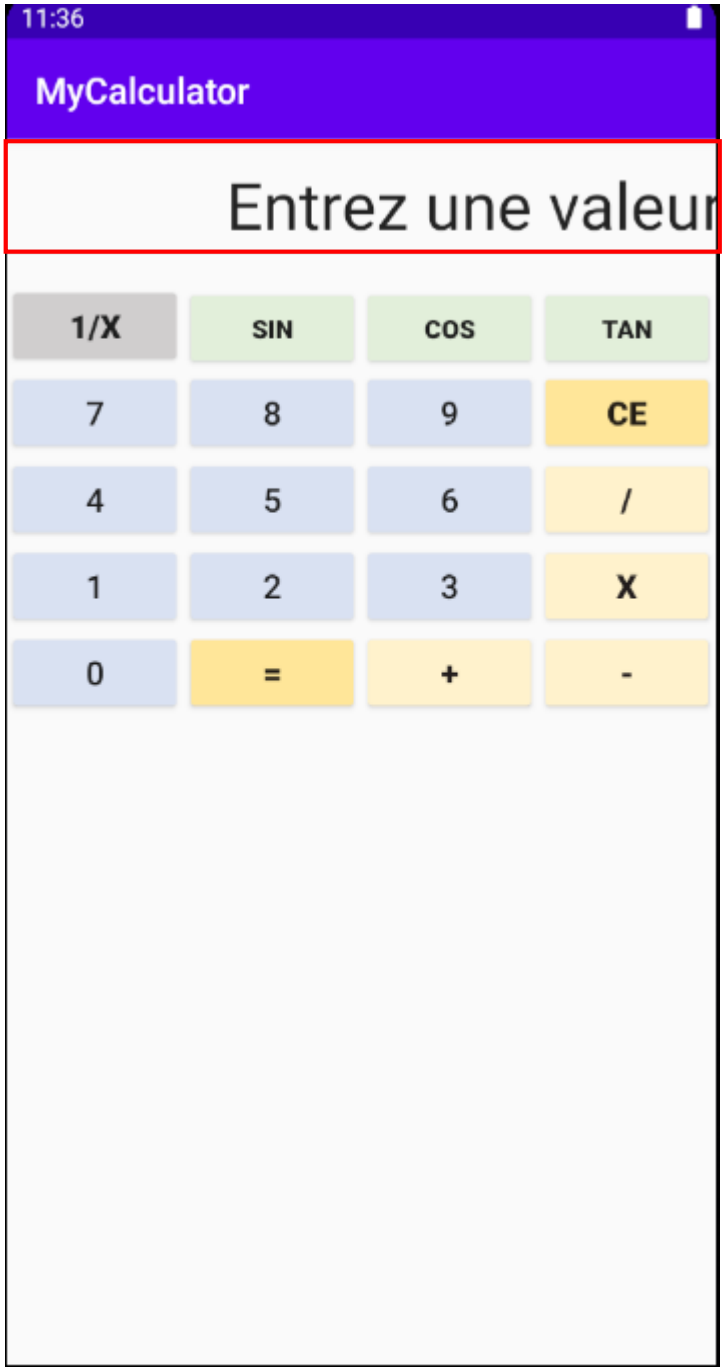
Le code suivant concerne la fonction inverse :

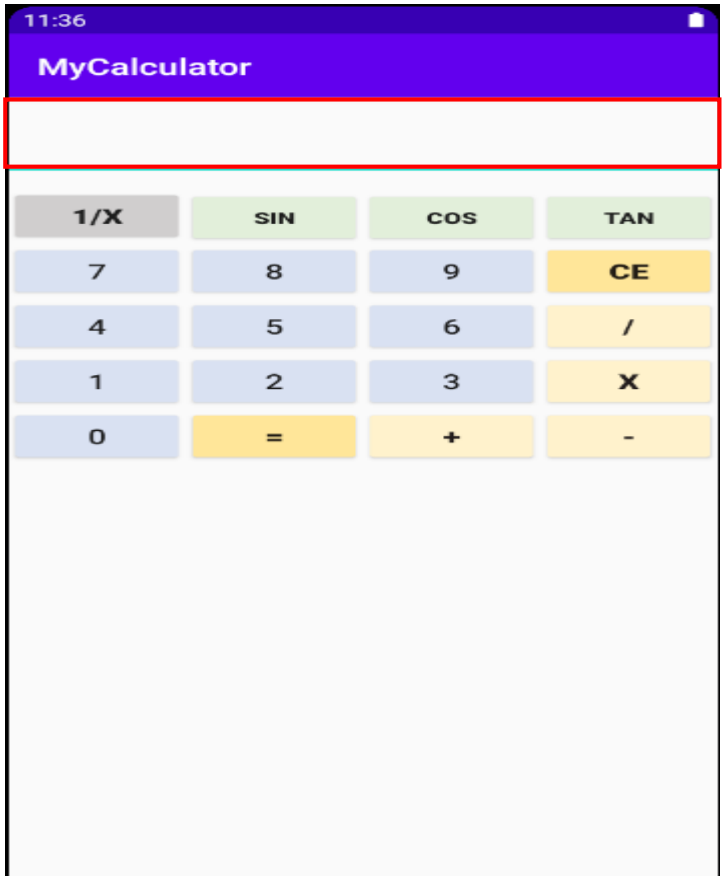
```
public void btninverse(View view){
    Editable valeur = ecran.getText();
    double result = 1/Double.parseDouble(String.valueOf(valeur));
    ecran.setText(Double.toString(result));
}
```

D. L'exécution de l'application :

A l'aide de l'émulateur on va essayer de faire quelques opérations et les exécuter et les accompagner avec des captures d'écran pour chaque opération.

Notre calculatrice a l'interface graphique suivante :



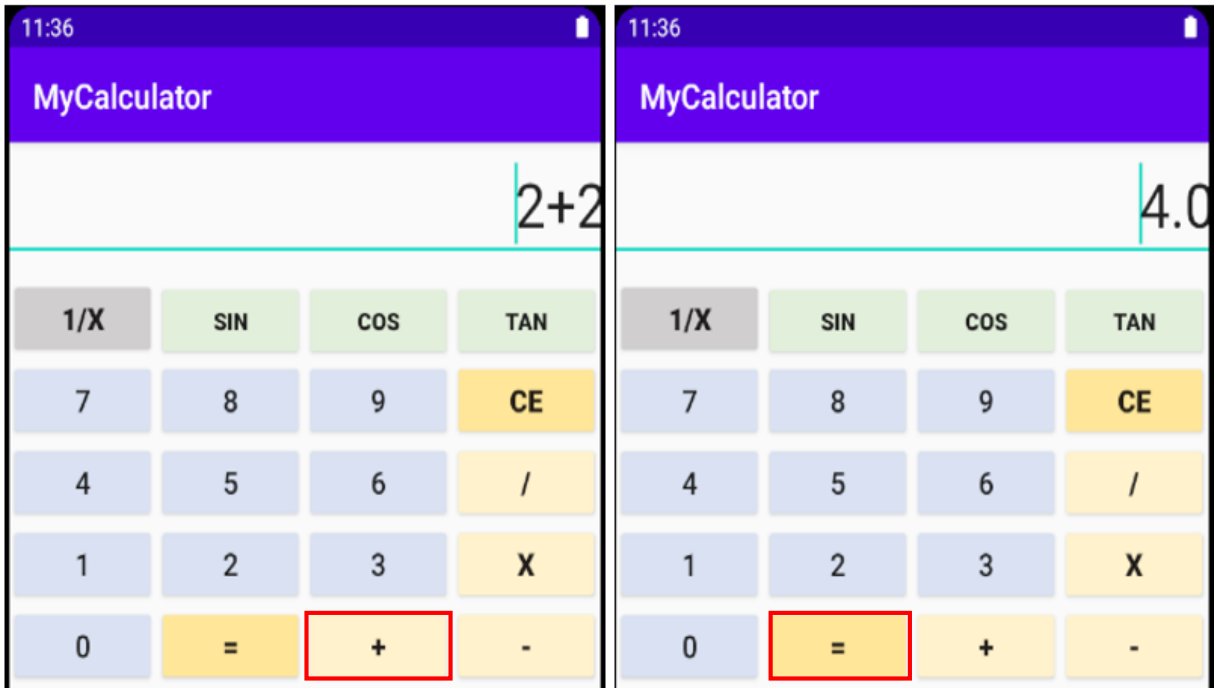


Une fois je tape le curseur sur l'écran un double clique des opérations et d'affichage, l'écran se vide automatiquement pour donner à l'utilisateur l'espace pour entrer son opération.

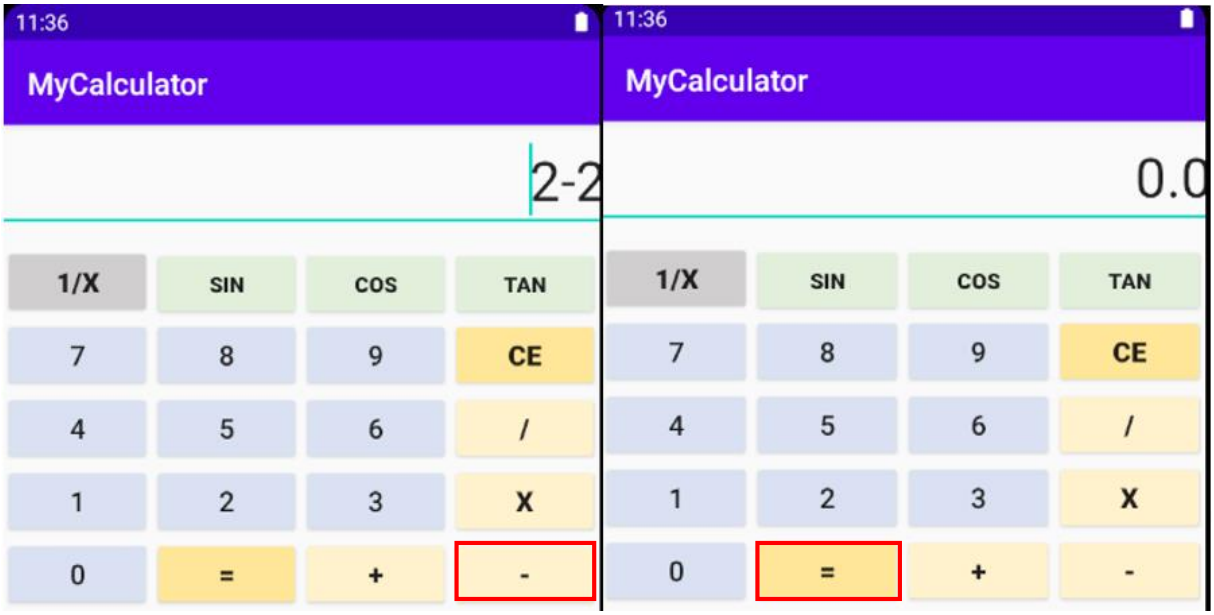
a. Les opérations élémentaires (+,-,x,/) :

On va maintenant entrer 4 opérations différentes et afficher leurs résultats :

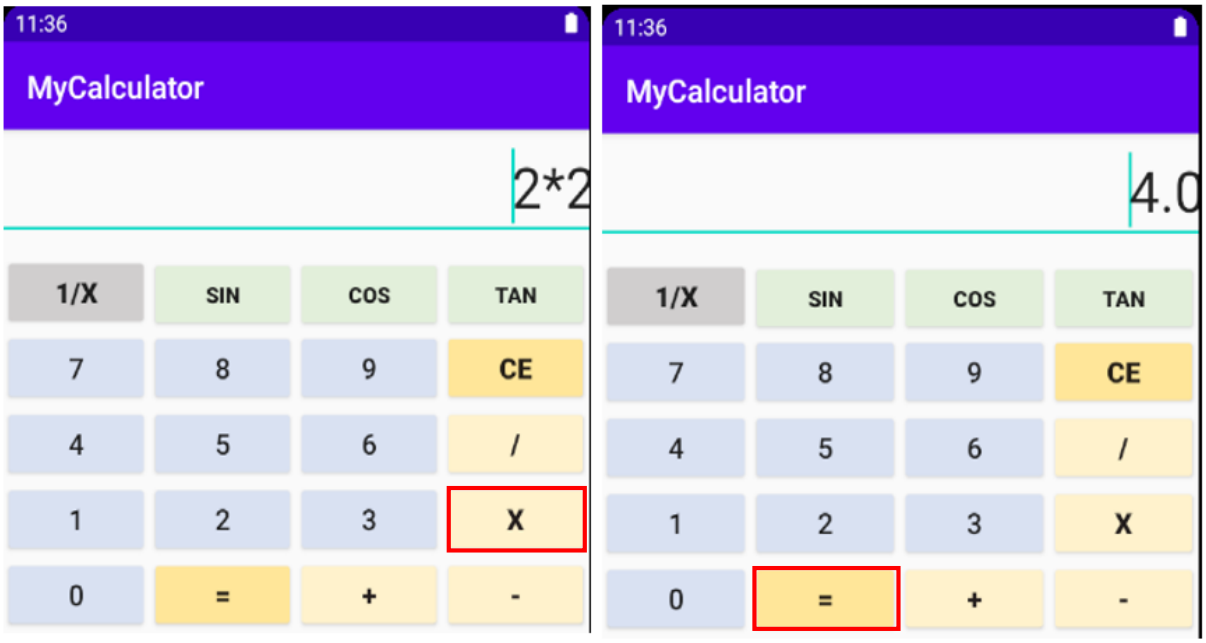
La figure suivante est un exemple d'opération d'addition :



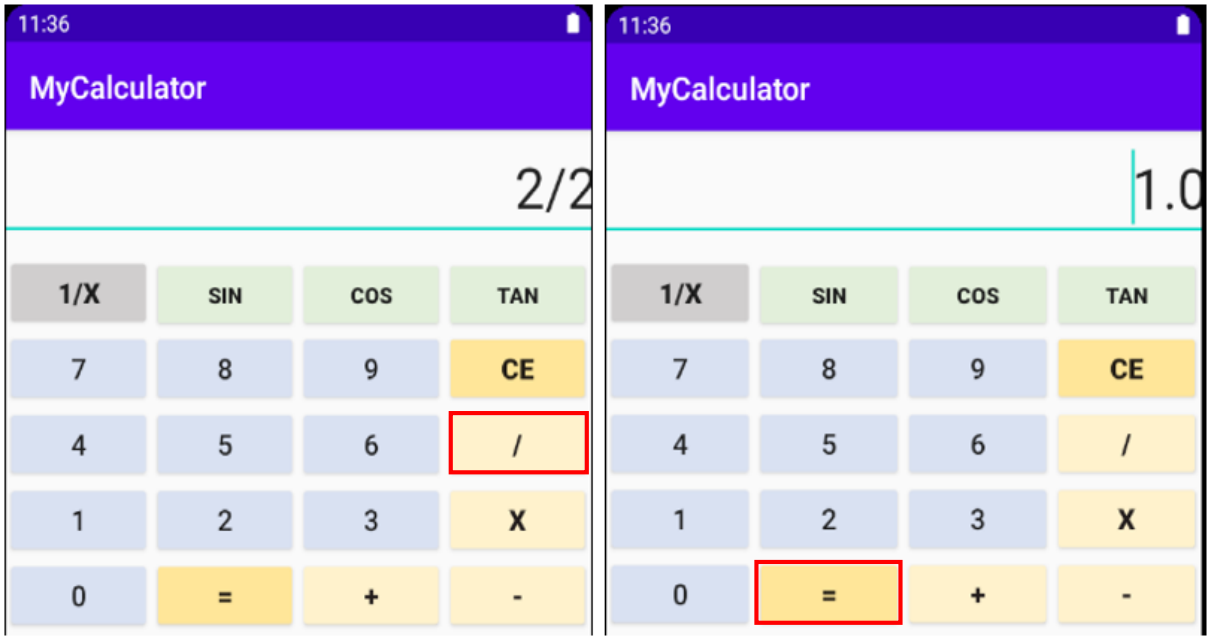
La figure suivante est un exemple d'opération de soustraction :



La figure suivante est un exemple d'opération de multiplication :

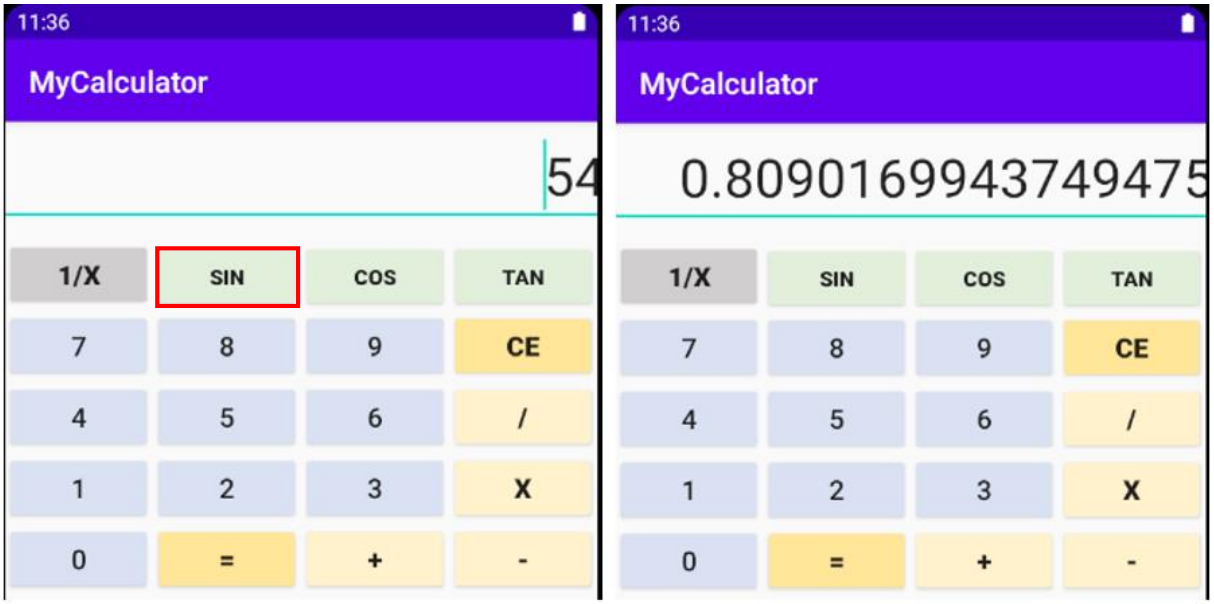


La figure suivante est un exemple d'opération de division :

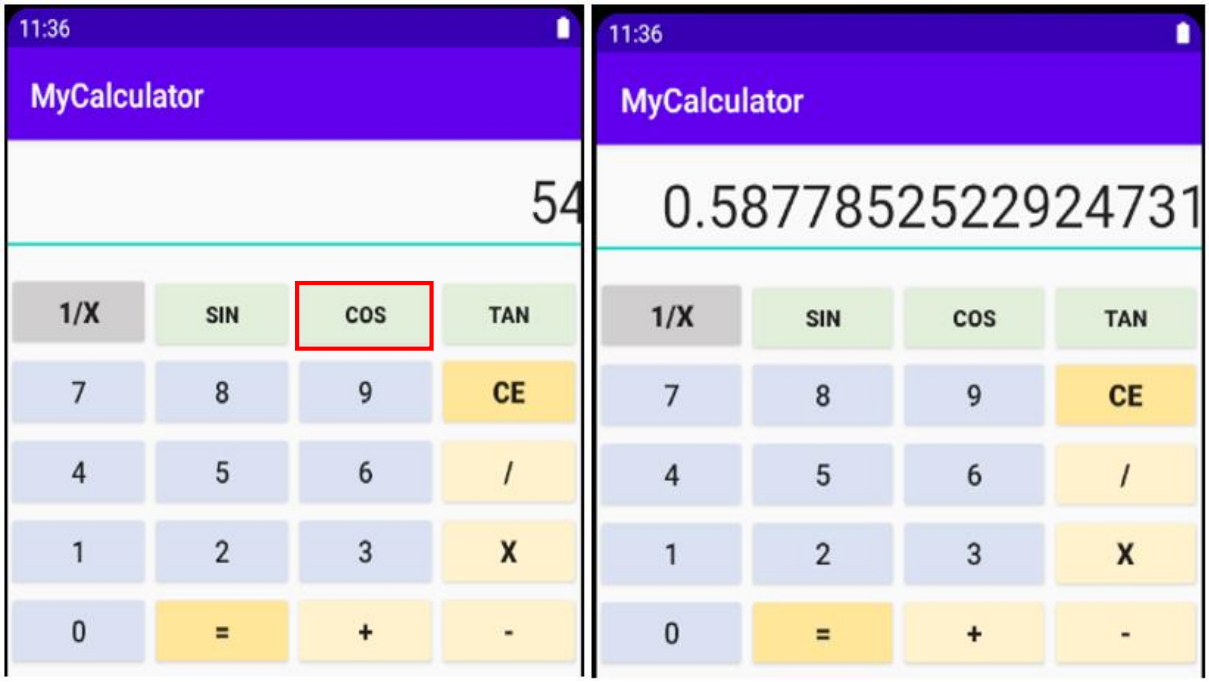


b. Les fonctions trigonométriques (Sin, cos et tan) :

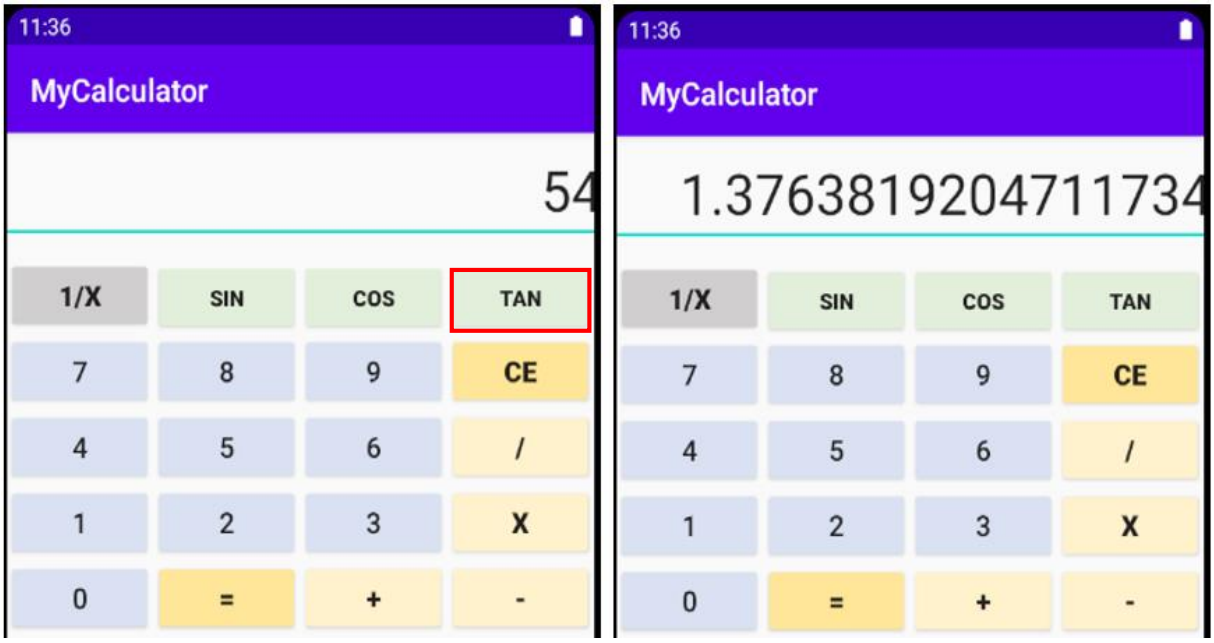
La figure suivante est un exemple dont on a utilisé la fonction sinus :



La figure suivante est un exemple dont on a utilisé la fonction cosinus :

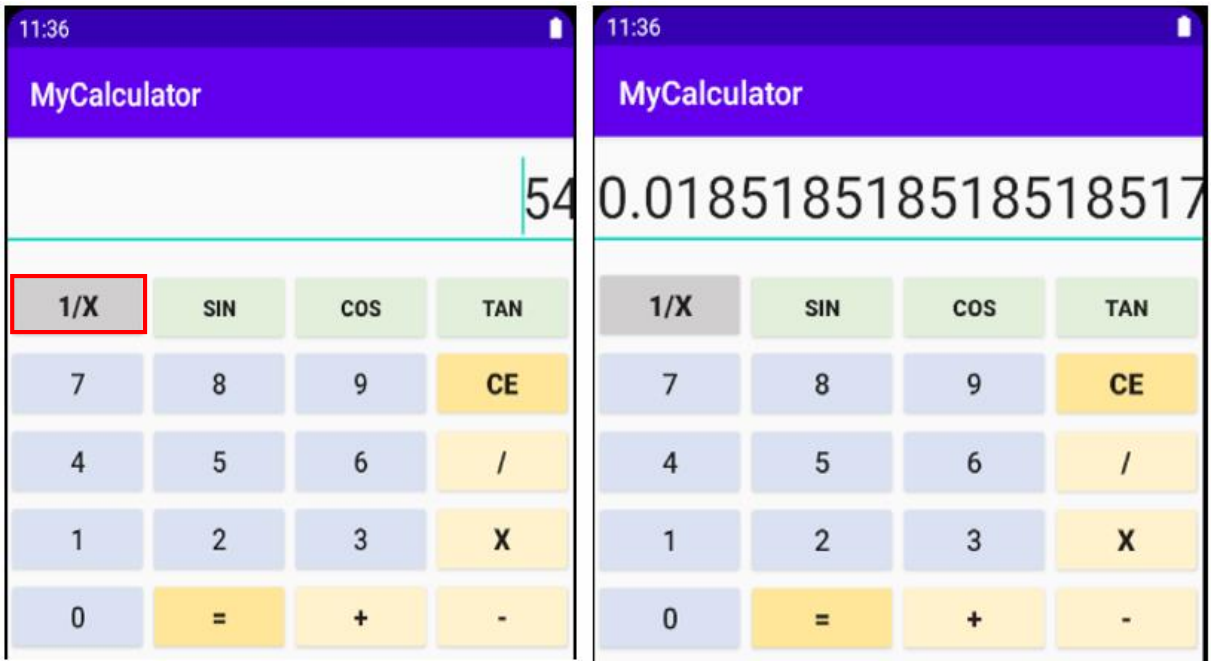


La figure suivante est un exemple dont on a utilisé la fonction tangente :



c. La fonction inverse 1/X :

La figure suivante est un exemple dont on a utilisé la fonction inverse :



Conclusion :

D’après ce mini-projet, on avait l’opportunité de découvrir le monde de la programmation mobile plus précisément la programmation native avec Android. Cette expérience nous a permet de se familiariser d’abord avec l’environnement de développement et par la suite avec l’implémentation du code en travaillant avec le design qui facilite et automatise plusieurs taches et grâce aux connaissances déjà précuises dans Java à travers l’autre demi-module, on a pu construire cette calculatrice scientifique et pouvoir atteindre nos objectifs à la fin de ce demi-module.