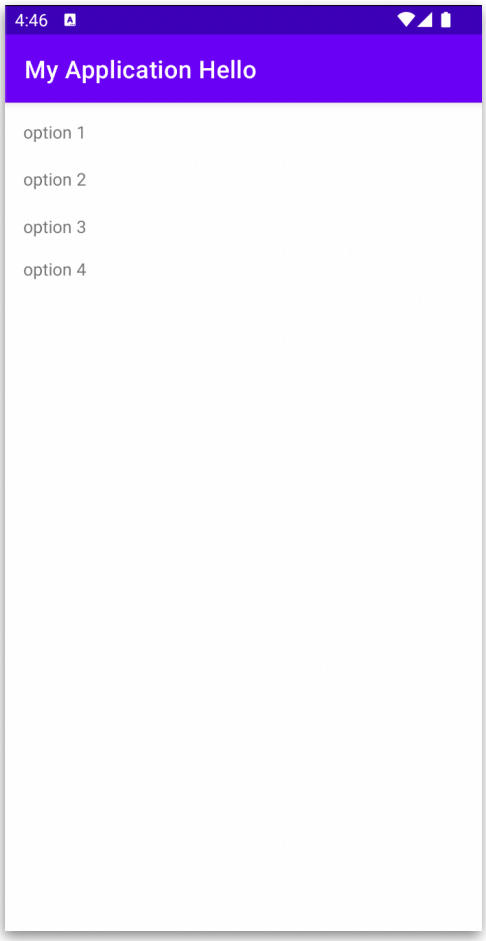
Übung 1 - Protokoll

Einleitung:
Ziel der ersten Übung ist es sich mit der Entwicklungsumgebung „Android Studio“ vertraut zu machen und eine erste einfache App zu programmieren.
Zuerst wurde eine einfache Anwendung erstellt, die nur vier TextViews darstellen soll und ein Feedback über das Log geben soll, falls es einen „Click“ gab.
Erweitert wurde dies durch ein visuelles Feedback (graues Einfärben der TextViews).
Aufgabe 3 ist es eine Anwendung zu programmieren, die die Umrechnung von Celsius und Fahrenheit übernimmt und umgekehrt, mittels zwei Buttons.
In der letzten Aufgabe muss ein vollfunktionstüchtiger Taschenrechner erstellt werden, welche die Rechnung mit den Zahlen 0-9 über die vier Grundrechenarten bewältigt.

Aufgabe 1:



1. App mit vier Optionen

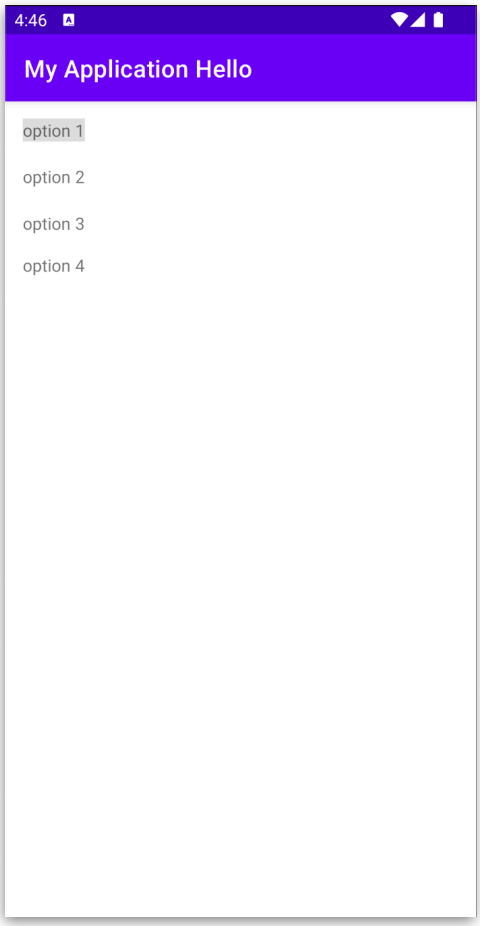
```
D/MainActivity: Textfeld 1 wurde ausgewählt
D/MainActivity: Textfeld 1 wurde ausgewählt
D/MainActivity: Textfeld 1 wurde ausgewählt
D/MainActivity: Textfeld 2 wurde ausgewählt
D/MainActivity: Textfeld 3 wurde ausgewählt
D/MainActivity: Textfeld 4 wurde ausgewählt
```

2. Ausgabe des Logs

Beim ersten Bild sieht man die vier TextViews (option …), sobald man das jeweilige TextView auswählt, wird wie im zweiten Bild das ausgewählte Textfeld über das Log ausgegeben.

Beschreibung der erstellten Klassen mit den dazugehörigen Methoden und Attributen im Kommentarstil erfolgte bereits in der Aufgabenstellung für Aufgabe 1.

Aufgabe 2:



1. App mit vier Optionen und visuellem Feedback

Sobald ein Textfeld vom Benutzer angeklickt wurde, ist der Hintergrund vom TextView grau untermalt.

```
<drawable
name="pressed_color">#DDDDDD</
drawable>

<drawable
name="normal_color">#00FFFFFF</
drawable>
```

Farben für gedrückt und normal (transparent) wurden in colors.xml definiert

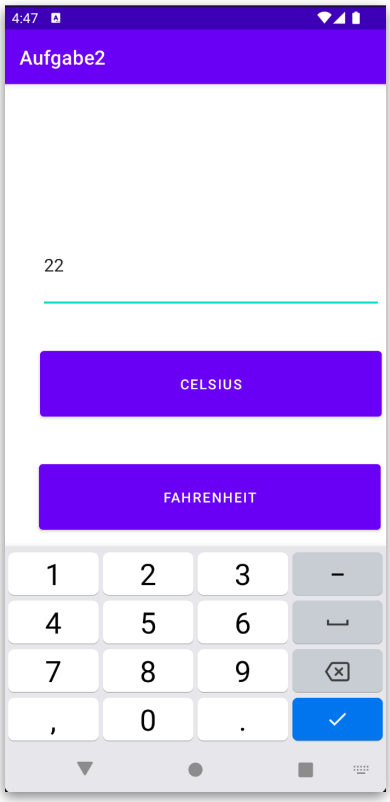
```
<item android:drawable="@drawable/
pressed_color"
android:state_pressed="true" />

<item android:drawable="@drawable/
normal_color" />
```

In res/drawable wurde textview_selector.xml erstellt, wo der Status definiert ist, ob die TextView gedrückt (Android:state_pressed=“true“) ist oder nicht.

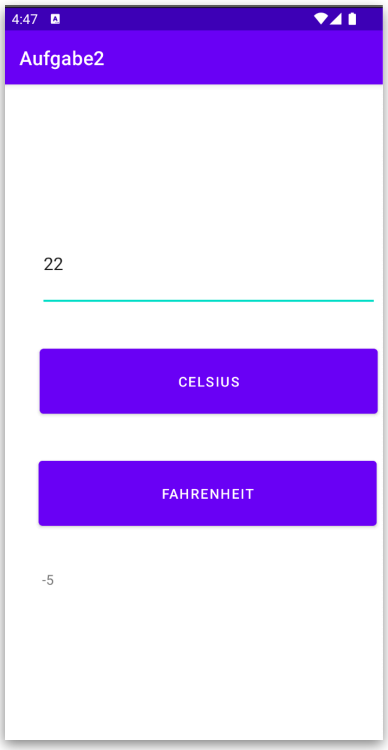
```
android:background="@drawable/textview_selector"
```

In activity_main.xml greifen wir auf den textview_selector zu.

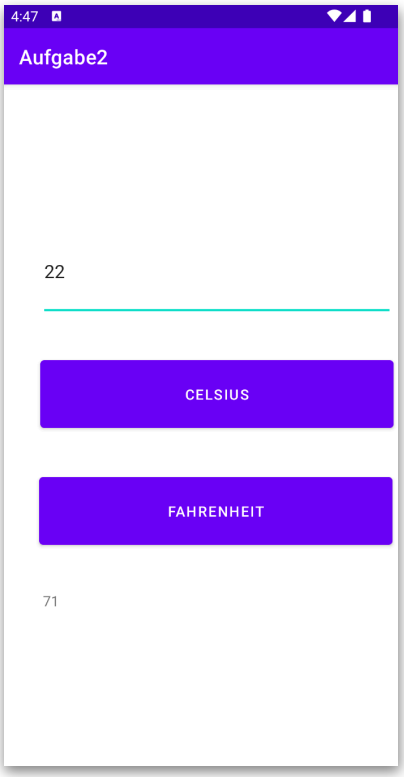
Aufgabe 3:



1. Eingabe der Temperatur



2. Temperatur in Celsius



3. Temperatur in Fahrenheit

Im ersten Bild erfolgt die Eingabe der Temperatur über die Zahlen-Tastatur.
Bei Bild 2 werden 22 ° Fahrenheit in -5 ° Celsius umgerechnet, sobald man den „CELSIUS“ - Button drückt.

```
activity_main.xml:


android:inputType="number"
```

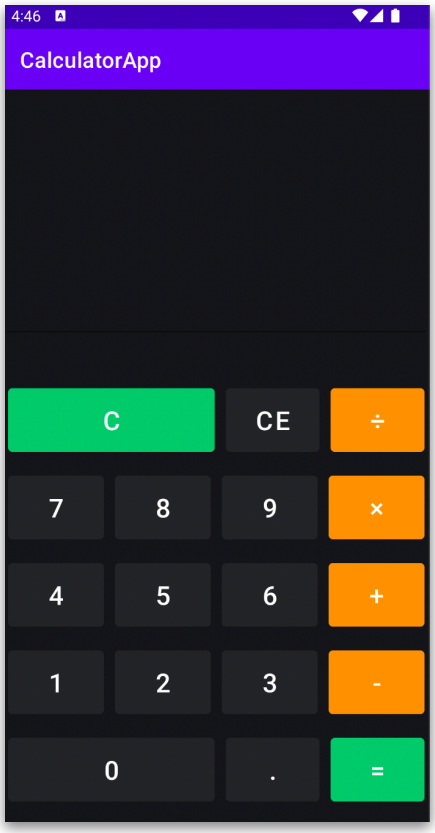
Zuletzt wird 22 ° Celsius in 71 ° Fahrenheit umgerechnet, sobald man den „FAHRENHEIT“ - Button drückt.

Mit android:inputType kann man festlegen, welchen Typ die Tastatur bei der Eingabe hat. Mit “number“ legt man fest, dass nur Zahlen eingegeben werden dürfen.

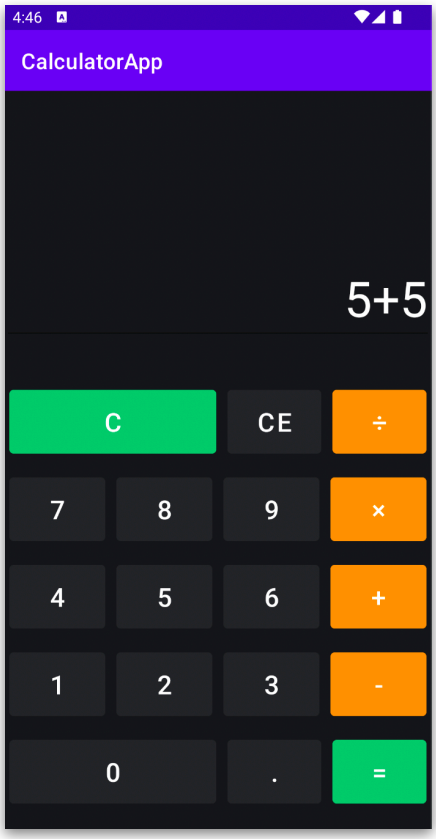
```
MainActivity.java:


buttonC.setOnClickListener(
        new View.OnClickListener() {
            public void onClick(View view) {


                String inputString =
editText.getText().toString();
                int f = Integer.parseInt(inputString);
                int celsius = (f - 32) * 5 / 9;
                textView.setText(" " + celsius);
                System.out.println(celsius);
            }
        });
```

Sobald der „Celsius“ - Button (buttonC) geklickt wurde, wird ein neuer OnClickListener erstellt, der bei onClick, den Text aus dem editText nimmt in einen String umwandelt (editText.getText().toString), anschließend wird dieses String zu einem Integer umgewandelt, um die Umrechnungsformel von Celsius zu Fahrenheit: C = (F - 32) * 5 / 9 anwenden zu können. Die umgewandelte Zahl wird dann im erstellten textView übermittelt und angezeigt. Beim „Fahrenheit“ - Button ist es ähnlich, nur das man da die Formel: F = C * 9 / 5 + 32 zur Umwandlung von Celsius zu Fahrenheit hat.

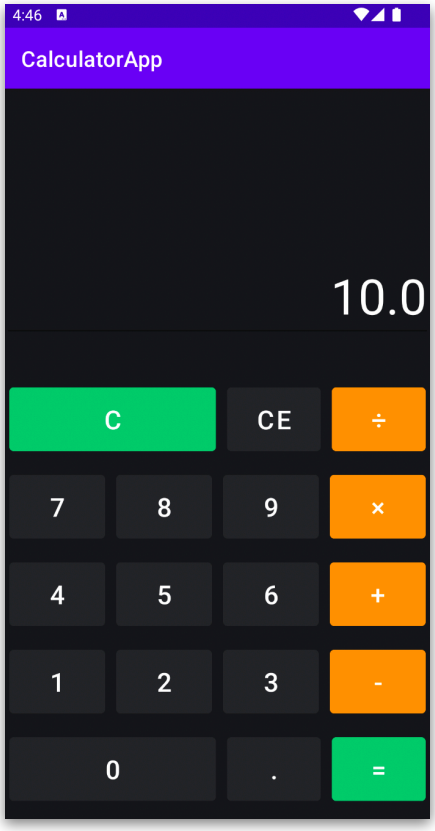
Aufgabe 4:



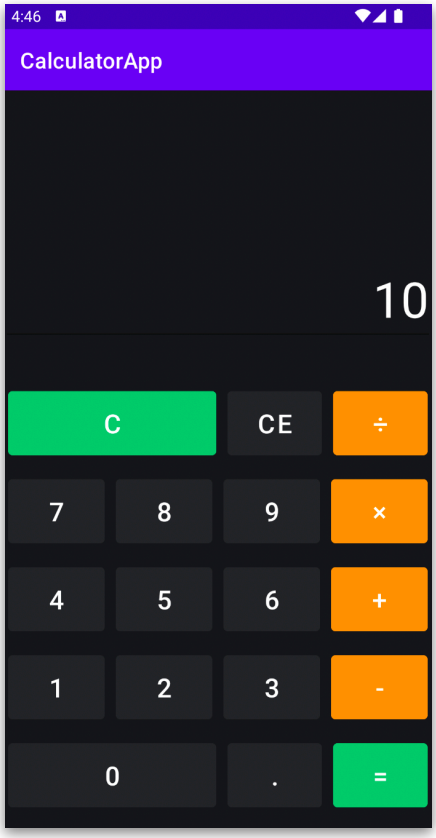
1. Startbildschirm



2. Eingabe der Operation



3. Ergebnis



4. Entfernen von Ziffern

Zuerst wird dem Benutzer die Oberfläche des Taschenrechners mit den geforderten Buttons angezeigt mit einem leeren EditText.
Im 2. Bild sieht man die Eingabe einer Operation (5+5) , die man mithilfe der Buttons tätigen kann.
Sobald man auf den Gleich-Button drückt (Bild 3), wird das Ergebnis auf dem EditText angezeigt (10.0).
Mit dem CE Button kann man einzelne Ziffern entfernen, so wurde bei Bild 4 „.0“ entfernt, sodass im EditText nur noch 10 steht.

```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.95"
    tools:layout_editor_absoluteX="1dp">
   <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <Button
            android:id="@+id/clear"
            style="?android:attr/buttonBarButtonStyle"
            android:layout_width="wrap_content"
            android:layout_height="70dp"
            android:layout_weight="1"
            android:layout_margin="5dp"
            android:backgroundTint="@color/green"
            android:onClick="clearBTN"
            android:text="@string/clear"
            android:textColor="@color/white"
            android:textSize="24sp" />
```

Um den Aufbau des Taschenrechners umzusetzen, habe mich für ein TableLayout entschieden, welches fünf TableRows enthält und dann die jeweiligen Buttons zu den dazugehörigen Reihen. Im Button wurde zuerst die ID festgelegt, anschließend, um die Fehlermeldung, dass der Button borderless sein muss zu entfernen, wurde „style="? android:attr/buttonBarButtonStyle“ von Android Studio hinzugefügt. Des Weiteren wurde die Attribute für das Layout gesetzt. Außerdem wurde eine Verknüpfung zum click-Event in MainActivity.java hergestellt mit „android:onClick="clearBTN"“.
Den Text für den Button kommt von der strings.xml Datei, in welcher festgelegt wurde, dass ein C dargestellt wird (<string name=„clear“>C</string>). Der Aufbau der Buttons in activity_main.xml ist überall gleich.

```
private EditText display;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);


    display = findViewById(R.id.input);
    display.setShowSoftInputOnFocus(false);


    display.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if
(getString(R.string.display).equals(display.getText().toStri
ng())) {
                display.setText("");
            }}});}
```

In der onCreate Methode in MainActivity.java wird dem Attribut display das EditText aus dem View zugeordnet. Mit der Equals Methode kann man den leeren String aus dem EditText entfernen, sodass man mit seinen Rechnungen anfangen kann. Damit die Android-Tastatur nicht erscheint, musste „display.setShowSoftInputOnFocus(false);“ eingesetzt werden.

```
private void updateText(String strToAdd) {
    String oldStr = display.getText().toString();
    int cursorPos = display.getSelectionStart();
    String leftStr = oldStr.substring(0, cursorPos);
    String rightStr = oldStr.substring(cursorPos);

if(getString(R.string.display).equals(display.getText().toSt
ring())) {
        display.setText(strToAdd);
        display.setSelection(cursorPos + 1);
    }
    else {
        display.setText(String.format("%s%s%s", leftStr,
strToAdd, rightStr));
        display.setSelection(cursorPos + 1);
    }
}
```

Mit updateText wird die Eingabe mittels der Buttons auf dem EditText aktualisiert. Dafür wird zuerst der alte Text genommen und mithilfe der Position des Cursors wird ein linkes und rechtes Substring erstellt. Falls das EditText-Feld leer ist wird der String hinzugefügt, der als Eingabe (strToAdd) von der Methode kommt. Ansonsten, falls man etwas zwischen den Ziffern oder Operartionszeichen hinzufügen möchte, wird der hinzuzufügende String zwischen dem linken und dem rechten Substring gepackt, sodass auch das möglich ist. Damit die Eingabe an der richtigen Stelle passiert, wird die Cursor-Position um 1 erweitert.

```
public void zeroBTN(View view) {
    updateText("0");
}
```

Für fast jeden Button wird die updateText-Methode aufgerufen mit jeweils dem Zeichen als Eingabe.

```
public void equalBTN(View view) {
    String userExp = display.getText().toString();

    userExp = userExp.replaceAll("÷", "/");
    userExp = userExp.replaceAll("×", "*");

    Expression exp = new Expression(userExp);
    String result = String.valueOf(exp.calculate());

    display.setText(result);
    display.setSelection(result.length());
}
```

Für den Gleich-Button wird der String aus dem EditText genommen und mithilfe des MxParser (importiert als .jar) wird die calculate Methode der Expression Klasse vom MxParser aufgerufen, um ein Ergebnis aus dem Input zu erstellen, dieses wird dann dem display Attribut weitergegeben. Außerdem werden die Zeichen ÷ , × in die Operationszeichen / , * umgewandelt, damit der MxParser damit arbeiten kann. Damit die Cursor Position immer aktualisiert ist, muss die Selection an die Länge des Ergebnis angepasst werden.

```
  public void clearEntryBTN(View view) {
        int cursorPos = display.getSelectionStart();
        int textLen = display.getText().length();

        if(cursorPos != 0 && textLen != 0) {
            SpannableStringBuilder selection =
(SpannableStringBuilder) display.getText();
            selection.replace(cursorPos - 1, cursorPos, "");
            display.setText(selection);
            display.setSelection(cursorPos - 1);
        }
    }

    public void clearBTN(View view) {
        display.setText("");
    }
}
```

Für den CE-Button müssen wir erstmal überprüfen, ob die aktuelle Position ungleich 0 ist und die Textlänge des EditText-Felds auch ungleich 0 ist, damit kein Fehler entsteht, wenn das Feld leer sein sollte. Mit der SpannableStringBuilder Klasse können wir dann Zeichen in unserem String ersetzen bzw. einfügen. Nachdem das Zeichen eingefügt wurde, muss die Position des Cursors um eins verschoben werden, damit man wieder an der richtigen Stelle ist.
Für den C-Button wird der Text im EditText mit einem leeren String ersetzt.

Tutorial, das für die Erstellung der Taschenrechner-App verwendet wurde:
https://www.youtube.com/watch?v=B5b-7uDtUp4

Zusammenfassung vom gelernten Inhalt und Schwierigkeiten:

Durch die Übung 1 konnte ein erster Eindruck vermittelt werden, wie das Konzept bei Android Apps mit den verschiedenen Klassen, wie zum Beispiel die MainActivity.java Klasse und main_activity.xml, die miteinander verknüpft sind, funktioniert.
Außerdem wurde mir klar, dass man Werte für Farben und Strings in extra dafür vorgesehene Klassen einfügen kann, um die, falls nötig wiederverwenden kann.
Vor allem durch die Taschenrechner App wurden die Lerninhalte aus den Aufgaben 1-3 wichtig, damit man bevor man eine richtige App programmiert, an das Thema herbeigeführt wird. Die Schwierigkeit lag dabei, den Code für die einzelnen Methoden aus der MainActivity.java Klasse vollständig zu verstehen, die notwendig sind, damit der Rechner reibungslos funktioniert.