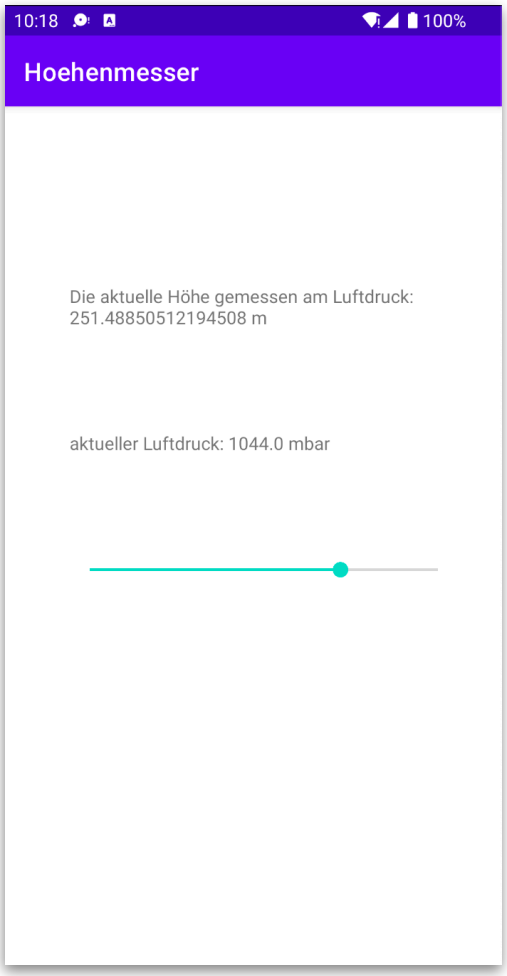
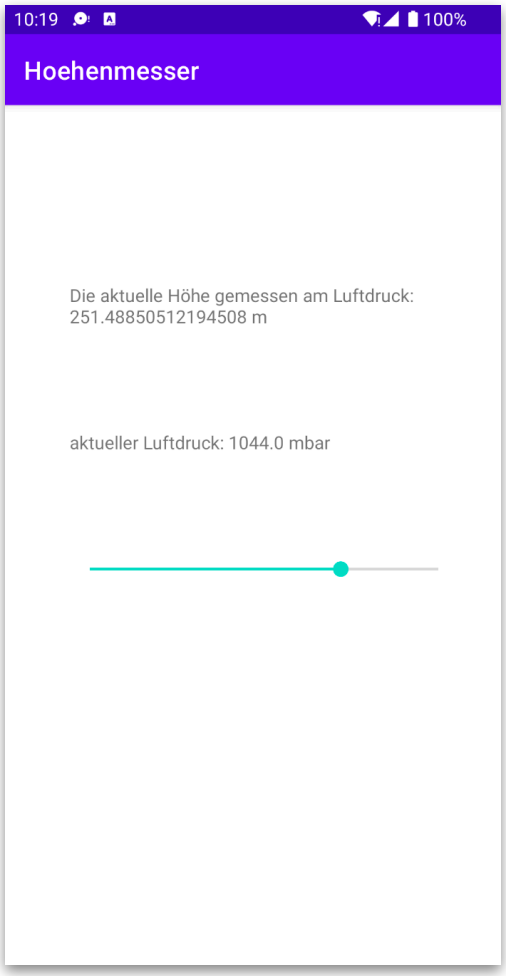
AppleJuice : Albert Kaminski, Sahiram Ravikumar, Dominik Domonell

Übung 3 - Protokoll

Aufgabe 1:



1. Luftdruck nach onCreate



2. Luftdruck nach onDestroy

Einleitung:
Ziel der Aufgabe war es den Höhenmesser aus Übung 2 mit der Option zu erweitern, den geänderten Luftdruckwert zu speichern, sodass dieser Wert nach Schließen der App noch der gleiche ist.

Zusammenfassung vom gelernten Inhalt und Schwierigkeiten:
Durch die Aufgabe hat man gelernt, wie man mit sharedPreferences von der Android Library Daten speichern kann, sodass diese wiederverwendbar sind.

Bearbeitung durch Dominik Domonell

```
private float defaultP = 1013.25F; // Druck bei 0 m Höhe
…
SharedPreferences sharedPref =
getPreferences(Context.MODE_PRIVATE); // sharedPreferences
Objekt zur lokalen Speicherung von Daten


float p =
sharedPref.getFloat(getString(R.string.defaultPressure),
defaultP); // get von dem defaultPressure Wert


seekbar.setProgress((int) p); // nimmt als Eingabe den
sharedPreferences Wert
…
@Override
public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
    SharedPreferences.Editor editor = sharedPref.edit();
    editor.putFloat(getString(R.string.defaultPressure),
progress); // ändert den sharedPreferences nach dem progress
Wert
    editor.apply();
}
```

Aufgabe 2:

1. GPS Tracker mit GPS Provider

2. GPS Tracker mit Network Provider

Einleitung:
Ziel der Aufgabe war es, GPS-Daten vom einem mobilen Endgerät (Uhrzeit, Längengrad, Breitengrad, Höhe) in eine CSV-Datei zu speichern und diese immer wieder zu erweitern, sobald neue Daten hinzukommen. Die Datei „data.csv“wird automatisch beim Öffnen der App erstellt.

data

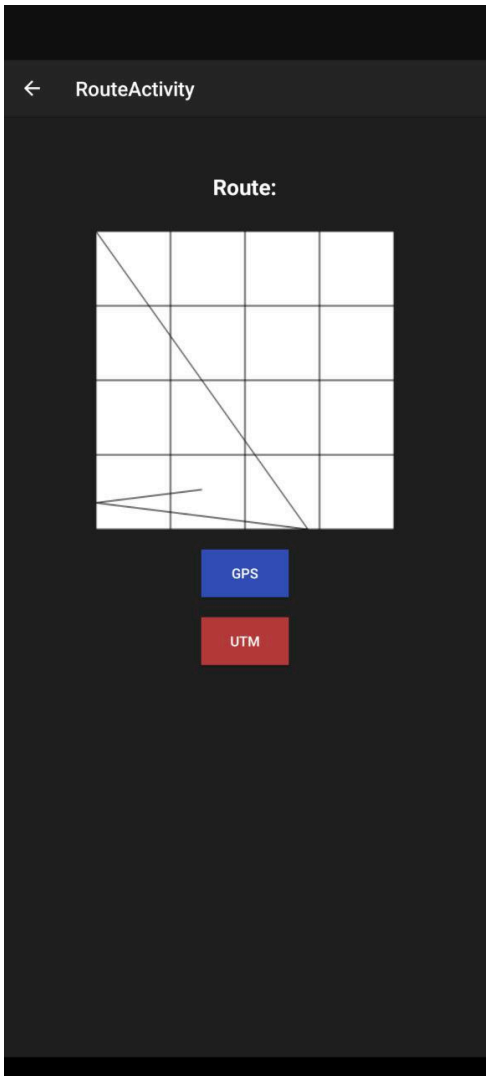
geografische Breite	geografische Länge	geografische Länge	aktuelle Zeit
52.4667156	13.3791687	84.30000305175781	2021-11-21 12:58:05

CSV-Datei mit GPS-Daten

Zusammenfassung vom gelernten Inhalt und Schwierigkeiten:
Durch die Aufgabe hat man gelernt, wie man Daten immer wieder lokal in eine CSV-Datei abspeichern kann und somit eine „Route“ zeichnen kann.

Bearbeitung durch Sahiram Ravikumar

Aufgabe 3:



1. getrackte Daten als Kurve

Einleitung
Ziel war eine App zu entwickeln, die die getrackten Daten als Kurve darstellt. Dafür soll die Kurve auf einem Gitter dargestellt werden, damit die Meridiane in UTM-Koordinaten angezeigt werden können.

Um das Gitter zu realisieren haben wir mit der createCanvas-Methode ein Fenster erstellt auf dem wir in der Methode Draw eine Gitter erzeugt haben. gpsToUMT fügt dann die getrackten Daten in die View ein.

```
void createCanvas() {

    ImageView imageView = findViewById(R.id.imageView_routeTracker);

    bitmap = Bitmap.createBitmap( width: 300,  height: 300, Bitmap.Config.ARGB_8888);
    canvas = new Canvas(bitmap);
    paint = new Paint(Paint.ANTI_ALIAS_FLAG);
    paint_background = new Paint();
    Paint paint_route = new Paint(Paint.ANTI_ALIAS_FLAG);

    paint_background.setStyle(Paint.Style.FILL);
    paint_background.setColor(Color.WHITE);

    paint.setColor(Color.BLACK);
    paint_route.setColor(Color.RED);

    bitmap_width = size = bitmap.getWidth();
    bitmap_height = bitmap.getHeight();

    imageView.setImageBitmap(bitmap);
}

void draw(List<String> lons, List<String> lats) {
    canvas.drawPaint(paint_background);

    canvas.drawLine( startX: 0,  startY: bitmap_height - bitmap_height / 4, bitmap_width,  stopY: bitmap_height - bitmap_height / 4, paint);
    canvas.drawLine( startX: 0,  startY: bitmap_height - 2 * bitmap_height / 4, bitmap_width,  stopY: bitmap_height - 2 * bitmap_height / 4, paint);
    canvas.drawLine( startX: 0,  startY: bitmap_height - 3 * bitmap_height / 4, bitmap_width,  stopY: bitmap_height - 3 * bitmap_height / 4, paint);
    canvas.drawLine( startX: 0,  startY: bitmap_height - 4 * bitmap_height / 4, bitmap_width,  stopY: bitmap_height - 4 * bitmap_height / 4, paint);
    canvas.drawLine( startX: 0, bitmap_height, bitmap_width, bitmap_height, paint);

    canvas.drawLine( startX: bitmap_width - 4 * bitmap_width / 4,  startY: 0,  stopX: bitmap_width - 4 * bitmap_width / 4, bitmap_height, paint);
    canvas.drawLine( startX: bitmap_width - 3 * bitmap_width / 4,  startY: 0,  stopX: bitmap_width - 3 * bitmap_width / 4, bitmap_height, paint);
    canvas.drawLine( startX: bitmap_width - 2 * bitmap_width / 4,  startY: 0,  stopX: bitmap_width - 2 * bitmap_width / 4, bitmap_height, paint);
    canvas.drawLine( startX: bitmap_width - bitmap_width / 4,  startY: 0,  stopX: bitmap_width - bitmap_width / 4, bitmap_height, paint);
    canvas.drawLine(bitmap_width,  startY: 0, bitmap_width, bitmap_height, paint);

    for (int i = 1; i < lons.size(); i++) {
        canvas.drawLine(getX(Float.parseFloat(lons.get(i - 1))), getY(Float.parseFloat(lats.get(i - 1))), getX(Float.parseFloat(lons.get(i))), ge
    }
}
```

```
void gpsToUMT() {
    for (int i = 0; i < longitudes.size(); i++) {
        LatLng ll4 = new LatLng(parseDouble(latitudes.get(i)), parseDouble(longitudes.get(i)));
        UTMRef utm2 = ll4.toUTMRef();
        utm = utm2.toString().split( regex: " ");
        String erdteil = utm[0];
        String utmx = utm[1];
        String utmy = utm[2];
        latitudesUTM.add(utmx);
        longitudesUTM.add(utmy);
    }
}
```

Zusammenfassung vom gelernten Inhalt und Schwierigkeiten:
In dieser Aufgabe haben wir gelernt, wie man GPS-Daten so verarbeiten kann, dass man sie als Kurve in einer App darstellen kann. Schwierigkeiten war hierbei die grafische Anzeige für die Kurve der GPS-Daten zu implementieren.

Bearbeitung durch uns alle

Aufgabe 4:

Einleitung:

Ziel war es, den GPS-Tracker zu erweitern und die Daten in einem GPX-Format zu speichern. Diese wird auch beim Öffnen der App erstellt.

```
public class GPXHandler {
    List<Date> dateList = new ArrayList<>();
    File file_storage_dir;

    void addDate(Date date) { dateList.add(date); }


    public void writePath(List<Location> points, Context context) {

        String header = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\" ?><gpx xmlns=\"http://www.topografix.co
        String name = "<name>" + "TDM" + "</name><trkseg>\n";

        String segments = "";
        DateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ss");

        for (Location loc : points) {
            segments += "<trkpt lat=\"" + loc.getLatitude() + "\" lon=\"" + loc.getLongitude() + "\">" + "<ele>" + loc.getA
        }

        String footer = "</trkseg></trk></gpx>";

        try {
            file_storage_dir = null;
            file_storage_dir = context.getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS);

            String time_stamp = new SimpleDateFormat( pattern: "yyyyMMdd__HHmmss").format(new Date());

            FileWriter writer = new FileWriter(new File( pathname: file_storage_dir.getPath() + File.separator + "data" + tim
            writer.append(header);
            writer.append(name);
            writer.append(segments);
            writer.append(footer);
            writer.flush();
            writer.close();
        } catch (IOException e) {
```

Die GPX Handler-Klasse ist dafür zuständig die GPS Daten in das richtige Format umzuwandeln und zu erweitern , dafür werden header, name, segments sowie dateFormat gesetzt. Wir iterieren dann über alle Locations und für jeden Punkt wird dann das Segment mit den Werten gefüllt damit diese dann im Writer mit der Append Methode zum File hinzugefügt werden.

Zusammenfassung vom gelernten Inhalt und Schwierigkeiten:

Die Schwierigkeit bestand darin die richtige Formatierung zu justieren um dem GPX-Dateiformat zu entsprechen. Durch das mehrfache Anwenden und justieren konnte so das Wissen vertieft werden.

Bearbeitung durch Albert Kaminski