Aufgabe 6

1. Erstellen Sie eine App mit einer Activity, die innerhalb eines TextView eine Frage anzeigt und die Antwort auf diese Frage in einem EditText erwartet. Die Überprüfung der Richtigkeit der Antwort soll durch das Klicken auf einen Button veranlasst und das Ergebnis in einer weiteren TextView angezeigt werden (s. Abbildung 1).
   Das Betätigen eines weiteren Buttons veranlasst die Anzeige einer weiteren Frage.

- Erstellen Sie eine dazu eine Anwendung mit einer Empty-Activity.
- Bearbeiten Sie die Datei activity_main.xml. Das Bearbeiten von Ressourcen ist innerhalb des Android Studios im Designer-Sicht und als Quellcode möglich (Umschalten über die Symbole in der rechten oberen Ecke). Ersetzen Sie das automatisch erstellte TextView durch ein vertikales LinearLayout (s. Vorlesungsscript).
- Erstellen Sie innerhalb des Layouts die benötigten Views mit sinnvollen Eigenschaften: 2 TextViews, 1 EditText (Plain Text) und 2 Buttons.
  Alle Views benötigen IDs (z.B.: **android:id="@+id/buttonTest"**) und
  die Buttons die onClick-Methoden (z.B. **android:onClick="onTestClick"**).
- Speichern Sie die Fragen und Antworten als values-Resource in zwei string-array in der Datei strings.xml. Fügen Sie den Arrays mehrere <item>s für Fragen und Antworten.
- Definieren Sie in MainActivity zwei Array-Variablen, z.B.
  ```
  String[] fragenArray;
  ```

- Definieren Sie weiterhin die Variablen für Views, z.B.
  ```
  TextView fragenView=null;
  ```

  In der onCreate-Methode sind die Array-Variablen mit den Werten aus Ressource zu füllen und die View-Variablen zu initialisieren.
  Der Zugriff auf die Value-Ressourcen ist über die Methode **getResources()** und der Zugriff auf die Views mit der Methode **findViewById(R.id…)** möglich, z.B.:
  ```
  fragenArray=this.getResources.getStringArray(R.array.fragen);
  fragenView= (TextView)findViewById(R.id.textFrage);
  ```

- Definieren Sie die onClick-Methoden mit dem Parameter View und geforderter Funktionalität.

  Hinweise:
- Sehr hilfreich beim Hinzufügen von imports und Erstellen von Methoden ist die Tastenkombination ALT+ENTER
- Wenn die Ressourcen im Quellcode nicht erkannt werden, hilft manchmal das Neukompilieren bzw. Neuöffnen des Projektes.
- Android Studio erlaubt im Logcat-Fenster die Kontrollausgaben mit Hilfe der Klasse Log (s. https://developer.android.com/reference/android/util/Log), z.B. kann mit `Log.v(String tag,String message)` eine VERBOSE-Message ausgegeben werden:
  ```
  Log.v("MainActivity","Meldung");
  ```

  Um die Menge der Ausgaben zu reduzieren können für logcat-Fenster die Einstellungen für Priorität und Inhalt der Ausgaben vorgenommen werden.

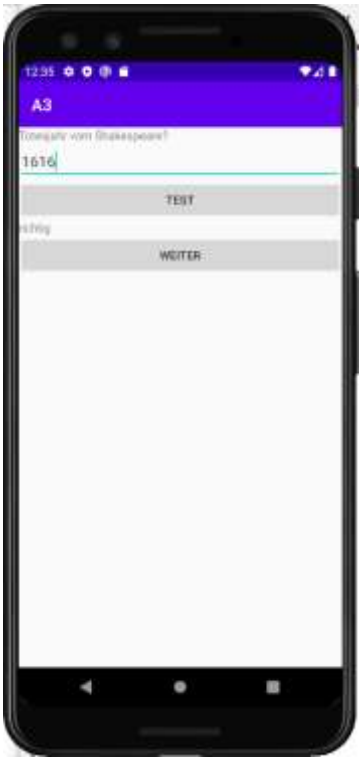
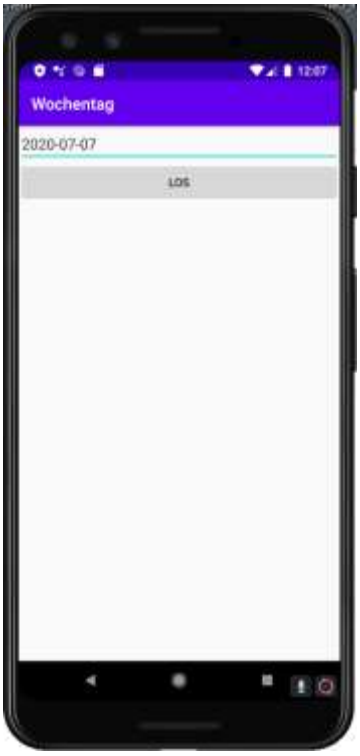


Abbildung 1



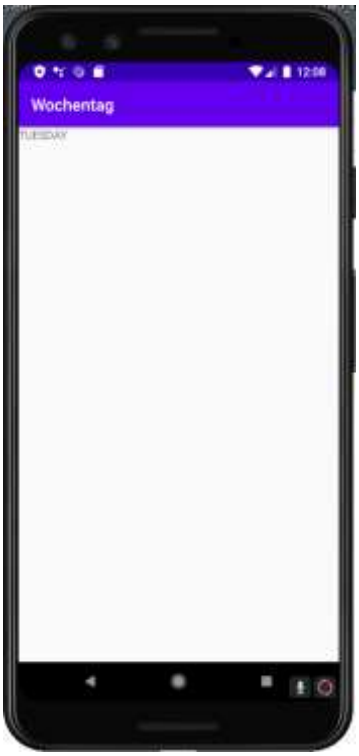


Abbildung 2

2.
Erstellen Sie eine App, die in der MainActivity die Eingabe einer Zeichenkette, die ein Datum repräsentieren soll, ermöglicht. Die Zeichenkette soll an die andere Aktivität übermittelt werden, die infolge des Betätigens eines Buttons gestartet wird und für das gegebene Datum den Wochentag ermittelt und in einem TextView anzeigt (s. Abbildung 2).

Hinweise:

- Er kann davon ausgegangen sein, dass der Nutzer die Zeichenkette im vorgesehenen Format "yyyy-mm-dd" eingibt, die Überprüfung ist nicht erforderlich.
- Zum Ermitteln des Datums bzw. des Wochentags können folgende Methoden verwendet werden:

```
LocalDate date=LocalDate.parse("2020-07-07");
String wochentag=date.getDayOfWeek().toString();
```

3.
Erstellen Sie eine App, die mit Hilfe eines Broadcast-Receivers eine Toast-Meldung über evtl. niedrigen Akkustand meldet:

```
Toast.makeText(context,"Achtung! Der Akku ist bald alle!",
                                    Toast.LENGTH_LONG). show();
```

Erstellen Sie eine von BroadcastReceiver abgeleitete Klasse mit der Methode onReceive(). In der Methode ist der Action-Typ des Intents zu überprüfen und im Falle einer `ACTION_BATTERY_LOW`-Action die Toast-Meldung zu senden.
Das Registrieren (bzw. Zurücksetzen) des Broadcastreceivers soll in den Methoden onResume() bzw. onPause() der MainActivity erfolgen. Der Intent-Filder ist mit `Intent.ACTION_BATTERY_LOW`-Action zu initialisieren.

4.
Erstellen Sie eine App, die die Ergebnisse einer Vogelzählung als UDP-Packet an einen Server sendet. Die App soll über die auf der Abbildung 3 dargestellte Oberfläche verfügen. Erfasst werden die Daten zur Art, dem Ort und der Anzahl der Vögel. Mit dem Betätigen des Buttons „SENDEN“ soll ein Work-Request an Work Manager geschickt werden. Das Versenden des UDP-Pakets erfolgt in der doWork()-Methode des Workers.

Um Workmanager API nutzen zu können, tragen Sie bitte ggf. in build.gradle (:app) folgendes dependency ein:

```
def work_version = "2.7.1"  //on Android 12 API 31
implementation "androidx.work:work-runtime:$work_version"
```

- Zum Verwenden von Sockets werden Internet-Zugriffsrechte benötigt. Tragen Sie diese in die Manifest-Datei ein:
  ```
  <uses-permission android:name="android.permission.INTERNET"/>
  ```

- Verändern Sie die activity_main.xml Datei entsprechend der Aufgabenstellung und erstellen Sie in der MainAktivity drei TextEdit- und ein Button-Objekt.
- Erstellen Sie eine vom androidx.work.Worker abgeleitete Klasse mit einem Konstruktor und der doWork()-Methode.
- Zur Übergabe der Daten von MainActivity an Worker wird in beiden Klassen eine übereinstimmente Key-Definition benötigt, z.B.:
  ```
  public static final String KEY_DATA = "data";
  ```
- In der MainActivity werden Daten- und WorkRequest-Objekte erstellt und WorkRequest an den WorkManager geschickt:
  ```
  Data myData = new Data.Builder()
          .putString(KEY_DATA, aktuelle_daten)
          .build();
  WorkRequest workRequest =
          new OneTimeWorkRequest.Builder(MyWorker.class)
                  .setInputData(myData)
                  .build();
  WorkManager
          .getInstance(getApplicationContext())
          .enqueue(workRequest);
  ```

- In der doWork()-Methode der Worker-Klasse werden die Daten mit Hilfe der Methode getInputData() zur Verfügung gestellt, z.B.:

  ```
  String s= getInputData().getString(KEY_DATA);
  ```

  Verwenden Sie zum Versenden der Daten die DatagramSocket und DatagramPacket. Adressieren Sie das Paket an localhost:

  ```
  InetAddress adr = InetAddress.getByName("10.0.2.2");
  ```

- Erstellen Sie zum Testen Ihrer App einen passenden Desktop-Server, z.B.:

```
public class UDPServer extends Thread{
public static void main(String[] args) {
      new UDPServer().start();
}
```

```
@Override
public void run() {
	while (!Thread.interrupted()){
		try {
			byte[] inhalt=new byte[1024];
			DatagramSocket socket = new DatagramSocket(5555);
			DatagramPacket packet=new DatagramPacket(inhalt,inhalt.length);
			socket.receive(packet);
			String daten= new String(inhalt, 0, packet.getLength() );
			System.out.println(daten);
			socket.close();
		} catch (Exception e) { e.printStackTrace();}
	}
}
}
```

5.

Erstellen Sie eine App, die wiederholt, z.B. täglich zum bestimmten Zeitpunkt, einen Vorschlag für die Lottozahlen als eine Notification anzeigen lässt, verwenden Sie um die wiederholte Ausführung um Hintergrund zu realisieren eine Service-Komponente, die vom AlarmManager gestartet wird. Service erzeugt die Zahlen und teilt diese dem Nutzer per Notification mit. Der AlarmManager wird über die Buttons START/STOP der MainActivity gestartet bzw. beendet (s. Abbildung 4).

- Erstellen Sie ein Service und fügen Sie der Klasse die onStartCommand-Methode hinzu. Beachten Sie, dass alle Komponenten im Manifest eingetragen werden müssen. Überprüfen Sie, ob Ihre Service-Komponente automatisch eingetragen wurde bzw. tragen Sie sie ein.

- Implementieren Sie die onStartCommand -Methode:
  Erzeugen Sie die Lottozahlen mit Hilfe der Random-Klasse und wandeln diese in eine Zeichenketten um.
  Zu erstellen sind außerdem die NotificationChannel und NotificationManager Objekte:

```
NotificationChannel channel=new NotificationChannel(CHANNEL_ID,
"Lottoinformationen", NotificationManager.IMPORTANCE_DEFAULT);
channel.setDescription("eigentlich nur Lottozahlen");
NotificationManager notificationManager =
getApplicationContext().getSystemService(NotificationManager.class);
notificationManager.createNotificationChannel(channel);
NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
CHANNEL_ID)
        .setSmallIcon(R.drawable.ic_stat_name)
        .setContentTitle("Deine Lottozahlen")
        .setContentText("hier zahlen")
        .setAutoCancel(true)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
int notificationId=9999;
notificationManager.notify(notificationId, builder.build());
```

  Notification-Symbol kann über Menü File/New/Image Asset im Android Studio erstellt werden. Wählen Sie als Icon-Typ „Notification Icons“.

- In der MainActivity wird in den Methoden, die auf den Click eines jeweiligen Buttons reagieren der AlarmManager gesetzt bzw. beendet.

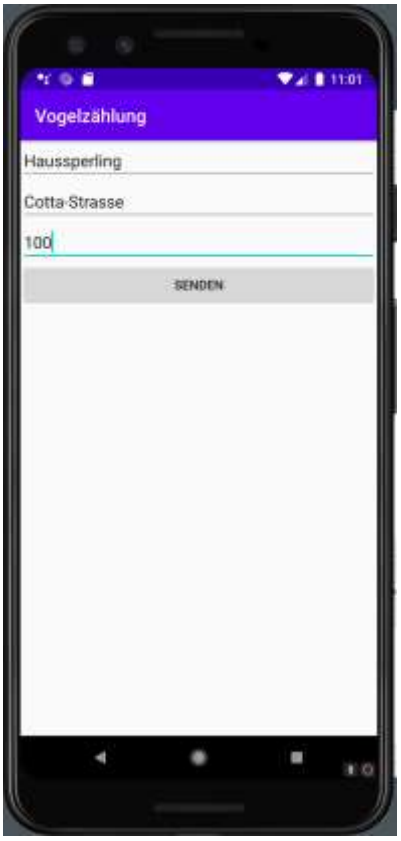


Abbildung 3

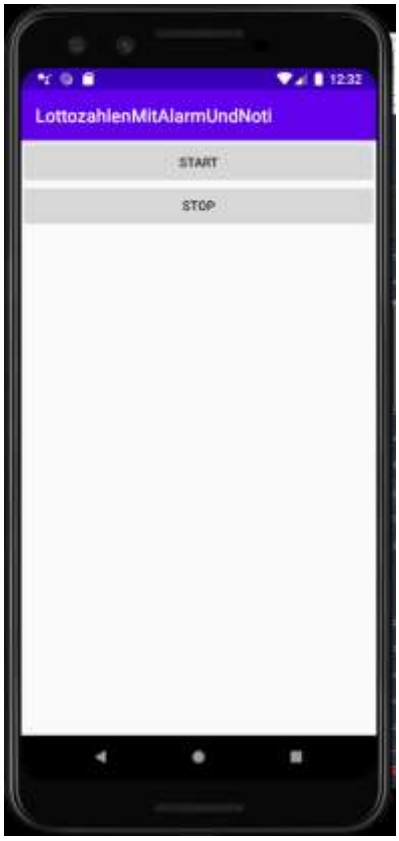


Abbildung 4