Aufgabe 3

Fragen

1. Was ist der Unterschied zwischen einer **View** und einer **ViewGroup**? Wie können diese geschachtelt sein?

- **View**: Ein einzelnes UI-Element (z.B. Button, TextView).
- **ViewGroup**: Ein Container für andere Views (z.B. LinearLayout, ConstraintLayout).
- **Schachtelung**: ViewGroups können andere Views und ViewGroups enthalten, wodurch hierarchische Layouts entstehen.

---

2. Gehört ein Button zu einer **View** oder einer **ViewGroup**? Welcher dieser beiden Elemente ist unsichtbar?

- Ein **Button** gehört zu einer **View**.
- **ViewGroups** sind unsichtbare Container, die lediglich zur Strukturierung dienen.

---

3. Was ist der Unterschied zwischen **LinearLayout** und **ConstraintLayout**?

- **LinearLayout**: Anordnung der Elemente linear (horizontal oder vertikal).
- **ConstraintLayout**: Flexible Anordnung der Elemente basierend auf Constraints zwischen den Komponenten.

---

4. Was versteht man unter einem Constraint beim ConstraintLayout?

- **Constraint**: Eine Regel, die die Position oder Größe einer View relativ zu anderen Views oder dem Container definiert.

---

5. Was versteht man unter dem **ComponentTree**? Dient dieser nur zur Ansicht, oder können auch Änderungen darin vorgenommen werden?

- **ComponentTree**: Hierarchische Darstellung der UI-Komponenten einer Activity.
- **Funktion**: Dient zur Ansicht und ermöglicht Änderungen an den Komponenten.

---

6. Was bedeutet **MatchConstraint**, wenn es für die Breite und Höhe einer View eingesetzt wird?

- Die View nimmt so viel Platz wie möglich ein, abhängig von definierten Constraints.

---

7. Was bedeutet: `android:id="@+id/edit_message"`. Erklären Sie die einzelnen Bestandteile.

- `android:id`: Attribut, das die ID einer View definiert.
- `@+id`: Erstellt eine neue ID.
- `edit_message`: Name der ID.

---

8. Kann mit Java-Code auf IDs zugegriffen werden?

- Ja, mit `findViewById(R.id.<id>)`.

---

9. In welchem File werden neue Ressourcen beim Kompilieren erzeugt?

- In der Datei **`R.java`**.

---

10. Was bedeutet **wrap_content** für die Höhe eines Button-Elements?

- Der Button passt seine Höhe an den Inhalt an.

---

11. Was bedeutet diese Zeile: `android:hint="@string/edit_message"`. Wieso fehlt in dieser Zeile das `+`-Zeichen nach dem `@`? Was bedeutet **hint**?

- **Bedeutung**: `hint` zeigt einen Hinweistext an, der verschwindet, wenn der Benutzer das Textfeld ausfüllt.
- **Fehlendes +**: Es wird eine vorhandene Ressource (`@string/edit_message`) referenziert, nicht erstellt.

---

12. In welchem File sollen **Strings** definiert werden?

- In der Datei **`res/values/strings.xml`**.

---

13. Was versteht man unter der Gewichtung (**weight value**) von Elementen?

- Die Gewichtung gibt an, wie viel Platz ein Element im Vergleich zu anderen erhält (meist im **LinearLayout**).

---

2. Fragen

---

1. Erkläre die Zeile: `setContentView(R.layout.activity_main)`. Wo im Code befindet sich diese Zeile meistens?

- **Erklärung**: Verknüpft das Layout (`activity_main.xml`) mit der Activity.
- **Position**: In der Methode `onCreate()` der Java- oder Kotlin-Klasse.

---

2. Wie muss die Signatur einer **click()**-Methode lauten?

- **Signatur**: `public void click(View view)`.

---

3. Welche Argumente braucht man, um einen **Intent** zu einer anderen Activity zu erzeugen?

- Kontext (`this` oder `getApplicationContext()`) und die Ziel-Activity-Klasse.

---

4. Wie kann man in der Java-Klasse auf den Inhalt eines Textfeldes zugreifen? Gib die benötigte Programmierung an.

```
EditText editText = findViewById(R.id.edit_message);
String message = editText.getText().toString();
```

---

5. Erkläre die Argumente der Zeile: `intent.putExtra(EXTRA_MESSAGE, message);`

- EXTRA_MESSAGE: Schlüssel zur Identifikation der Daten.
- message: Wert, der übertragen wird.

---

6. Wenn mit Android Studio eine neue Activity angelegt wird, welche Dateien werden hinzugefügt bzw. welche Dateien werden geändert?

- **Hinzugefügt**:
    - Neue Java-/Kotlin-Datei für die Activity.
    - Neues Layout-File.
- **Geändert**: - AndroidManifest.xml.

---

7. Erkläre folgende Aktionen vom Android-Designer:

- **Turn On Autoconnect**: Automatische Verknüpfung von Views bei Platzierung.
- **Show Blueprint**: Zeigt eine Drahtgitter-Ansicht des Layouts.
- **Add vertical Guideline**: Fügt eine vertikale Linie zur Ausrichtung der Views hinzu.

---

8. Wie können ankommende Daten von einer Activity ausgelesen und angezeigt werden?

Methode:

```
Intent intent = getIntent();
String message = intent.getStringExtra(EXTRA_MESSAGE);
textView.setText(message);
```

---

9. Welche Änderungen muss man im Manifest durchführen, damit der Back-Button funktioniert?

- Keine spezifischen Änderungen nötig, der Back-Button funktioniert standardmäßig.

---

10. Welche Hilfe bietet die Tastenkombination ALT+Enter?

- Funktion:
    - Automatische Fehlerbehebung
    - Import von Bibliotheken
    - Erstellung von fehlenden Klassen/Methoden.