Aufgabensammlung zu ICT-BBCH Modul M335

Eine Applikation für mobile Geräte nach Vorgabe umsetzen und testen.

Version BIVO 2014

Inhaltsverzeichnis

[1 Unterscheidung Mobile Apps, Desktop Applikationen (--) 2](#_Toc502499088)

[2 Entwicklungsumgebung einrichten (3.1 – 3.6) 3](#_Toc502499089)

[3 App mit Tonausgabe erstellen (4.1) 4](#_Toc502499090)

[4 Bestandteile einer Android-App (4.2) 5](#_Toc502499091)

[5 Einfache Elemente in die Benutzeroberfläche einfügen (4.3) 7](#_Toc502499092)

[6 RelativeLayout und LinearLayout benutzen (4.3) 8](#_Toc502499093)

[7 FrameLayout benutzen (4.3) 10](#_Toc502499094)

[8 Die Eigenschaften Layout\_width und Layout\_height (4.3) 12](#_Toc502499095)

[9 Die Eigenschaft layout\_weight (4.3) 14](#_Toc502499096)

[10 Benutzerfreundliche EditTexts (4.3) 16](#_Toc502499097)

[11 Benutzereingaben verarbeiten (4.4) 18](#_Toc502499098)

[12 Aktivität mit mehreren Buttons (4.4) 19](#_Toc502499099)

[13 Eine App installieren (4.5) 20](#_Toc502499100)

[14 Spiel entwickeln (1) (5.1) 21](#_Toc502499101)

[15 Weitere Aktivitäten starten (5.1) 22](#_Toc502499102)

[16 Grafiken einbinden (5.2) 23](#_Toc502499103)

[17 Grösseneinheiten (5.2) 24](#_Toc502499104)

[18 Game Engine aufbauen (5.3) 25](#_Toc502499105)

[19 Layout für das Mückenspiel einrichten (5.3) 26](#_Toc502499106)

[20 Methode *Bildschirmaktualisieren* (5.3) 29](#_Toc502499107)

[21 Informationen über das eingesetzte Smartphone abrufen (5.3) 30](#_Toc502499108)

[22 Der erste Mückenfang (5.3) 31](#_Toc502499109)

[23 Sound abspielen (6.2) 33](#_Toc502499110)

[24 Lebenszyklus einer Aktivität (6.2) 34](#_Toc502499111)

[25 Problem mit dem Lebenszyklus (6.2) 36](#_Toc502499112)

[26 Daten (Highscores) lokal speichern (7.1) 37](#_Toc502499113)

[27 Aktivität mit mehreren Rückgabewerten (7.1) 38](#_Toc502499114)

[28 Daten (Highscores) online speichern (7.2) 39](#_Toc502499115)

[29 Liste mit Adapter (7.3) 42](#_Toc502499116)

[30 Die Kamera verwenden (8.1) 43](#_Toc502499117)

[31 Sensoren (1) (9.1) 44](#_Toc502499118)

[32 Sensoren (2) (9.1) 45](#_Toc502499119)

[33 Arbeiten mit Geokoordinaten (9.5) 46](#_Toc502499120)

[34 Guidelines und Vorschriften (--) 47](#_Toc502499121)

[35 App testen (--) 49](#_Toc502499122)

# Unterscheidung Mobile Apps, Desktop Applikationen (--)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können aufzeigen, wie sich Mobile Apps von Desktop Applikationen unterscheiden und was dies für die Programmierung von Mobile Apps bedeutet. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Die Programmierung einer App ist ähnlich, wie die Programmierung einer Desktop oder Web Applikation. Die unter Android verwendete Programmiersprache (Java) kennen Sie schon aus anderen Modulen. Es sind aber auch wesentliche Unterschiede, die berücksichtigt werden müssen. Ein typisches Beispiel dafür sind die Bildschirmgrösse und Bildschirmauflösung. Die Bedienung erfolgt nicht mehr mithilfe einer Maus, sondern direkt mit dem Finger. Um Apps benutzerfreundlich zu entwickeln müssen, diese Unterschiede **bewusst** sein. Nur wer das berücksichtigt, wird Apps entwickeln, die von den Benutzern akzeptiert und dauerhaft installiert werden. Und nicht schon nach ein paar Tagen wieder desinstalliert. Ein schönes Design reicht nicht aus. |
| **Aufgabe** | Erstellen Sie eine Liste von Faktoren und Merkmalen, die beide Welten unterscheiden. Unterschiede zwischen Smartphone und Tablett wollen wir an dieser Stelle nicht beachten.  Beschreiben Sie, wie diese Unterschiede jeweils das Programmieren von Apps beeinflusst. |
| **Hinweis Hilfsmittel** | * Nehmen Sie die Tabellen im Anhang als Ausgangspunkt und Beispiel. * Lösen diese Aufgabe in 2er oder 3er-Teams. |
| **Ergebnis** | * Eine Tabelle mit mindestens 5 Unterscheidungskriterien bzw. Merkmalen und deren Einfluss auf die Entwicklung bzw. Programmierung von App. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Faktor** | **Merkmal**  ***(nur wo notwendig)*** | **PC / Notebook** | **Smartphone** | **Einfluss auf die Programmierung (Massnahmen)** |
| Bildschirm | Grösse | ca. 19" bis 27" | 4.3", 4.7", 5", 5.1", 5.2", 5.7" | Sicher stellen, dass der Inhalt nicht auf der rechten Seite abgeschnitten wird oder andere Inhalte überlappt. App unbedingt mit mehreren verschiedenen Smartphone bzw. Auflösungen und screen densities testen. |
| Bildschirm | Auflösung | 1280x1024, 1920x1080, 1920x1200, 2560x1440 | 720x1280, 1080x1920, 1440x2560 |
|  |  |  |  |  |

# Entwicklungsumgebung einrichten (3.1 – 3.6)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können die IDE Android Studio und das SDK installieren. Sie sind in der Lage eine einfache App zu erstellen und auf einem Smartphone oder im Emulator zu testen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Sie haben die Absicht native Applikationen für das Betriebssystem Android zu entwickeln. Sie benötigen dafür eine passende Entwicklungsumgebung. |
| **Aufgabe** | Installieren Sie die neueste Version von Android Studio inkl. SDK.  Erstellen Sie eine kleine App **TestApp** unter Berücksichtigung nachfolgender Anforderungen:   * Company Domain: *Klassenbezeichnung.ch*. Zum Beispiel: **ifzxxx.ch** * Plattform: **Phone and Tablet** * Activity: **Empty Activity**   Testen Sie Ihre App, indem Sie Ihr Smartphone mit dem PC verbinden und die App mit dem grünen Pfeil in der Symbolleiste von Android Studio starten. Alternativ können Sie Ihre App mit einem "Android Virtual Device", zum Beispiel "Nexus 5 API 23 x86", testen. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet, zum Beispiel: <http://developer.android.com/sdk/index.html> |
| **Ergebnis** | * Aktuelle Android Studio und Android SDK installiert und einsatzbereit. * Die fertige App **TestApp**. * Die App ist auf dem Smartphone installiert und gestartet oder wurde im Emulator geladen und gestartet. |

# App mit Tonausgabe erstellen (4.1)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können ein Programm mit Sprachausgabe erstellen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Der Leiter der Abteilung Verkauf & Marketing der Topomedics wünscht sich für seine Mitarbeiter eine App, mit deren Hilfe Standardsätze in verschiedenen Sprachen abgerufen und über den Lautsprecher ausgegeben werden können. Dies soll die Kommunikation bei Verkaufsgesprächen unterstützen. In einer zukünftigen Version sollen die Verkäufer die Möglichkeit haben, eigene Sätze einzugeben, die von der App übersetzt und ausgesprochen werden.  Sie erkunden zuerst die Möglichkeiten von Android, indem Sie eine kleine Beispiel‑App erstellen. |
| **Aufgabe** | Erstellen Sie eine neue App **SagHallo** unter Berücksichtigung nachfolgender Anforderungen:   * Company Domain: *Klassenbezeichnung.ch*. Zum Beispiel: **ifzxxx.ch** * Plattform: **Phone and Tablet** * Activity: **Empty Activity**   Ergänzen Sie das Programm, wie im Buch beschrieben (Kapitel 4.1).  Finden Sie heraus, wie Sie die Lautstärke bestimmen können. Zum Beispiel so laut, wie die von Ihnen eingestellte Lautstärke für Musik, Video und Spiele.  Testen Sie Ihre App |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. |

# Bestandteile einer Android-App (4.2)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie kennen die wichtigsten Dateien eines Android-Projektes und wo diese zu finden sind. |
| **Aufgabentyp** | Einzelaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Das Wissen, wo was gespeichert wird, kann sehr nützlich sein, wenn Fehler gesucht werden oder wenn Sie kleinere Änderungen an Ihrer App auf effiziente Weise vornehmen wollen. |
| **Aufgabe** | Erstellen Sie eine neue App **Aufbau** mit *Backwards Compatibility (AppCompat)* aktiviert.  Aktivieren Sie den Tab "activity\_main.xml" und ziehen Sie "Hello world!" in die obere linke Ecke des Devices.  Lösen Sie die Aufgaben im Anhang, indem Sie das *AndroidManifest* studieren bzw. verändern und ggf. die entsprechende Datei öffnen. Schreiben Sie gewonnene Erkenntnisse auf. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | Korrekte Beantwortung der Fragen. |

|  |  |
| --- | --- |
| **Aufgabe 1** | Finden Sie heraus, welche Zeile im *AndroidManifest* für die Auswahl des Icons für den Launcher zuständig ist. |
|  | Zeile 7: "**android:icon="@mipmap/ic\_launcher**" |
| **Aufgabe 2** | Im Android-View (Explorer) finden Sie einen Ordner "res". In einem seiner Unterordner finden Sie das Icon in mehreren Auflösungen. Finden Sie heraus, wo diese Dateien im Windows abgelegt sind. |
|  | **app/src/main/res/**mipmap-xxxx |
| **Aufgabe 3** | Sie möchten, dass Ihre App nur in der Bildschirmausrichtung *Portrait* angezeigt wird. Wenn Sie das Smartphone drehen, soll sich der Inhalt nicht mitdrehen. Fügen Sie das entsprechende Attribut ein. |
|  | AndroidManifest  android:screenOrientation="portrait" bei <activity> hinzufügen |

|  |  |
| --- | --- |
| **Aufgabe 4** | Finden Sie heraus, wo Sie die Hintergrundfarbe für das Label **Aufbau** ändern können. Wählen Sie eine hellere Farbe aus. Zum Beispiel: 7986CB |
|  | **res/values/colors.xml** |
| **Aufgabe 5** | Finden Sie heraus, wo Sie das Label **Aufbau** ändern können.  Bitte NICHT direkt im *AndroidManifest.xml* anpassen! Ändern Sie es zu **Aufbau Übung**. |
|  | **res/values/strings.xml** |
| **Aufgabe 6** | Wo ist der Text "Hello World!" wohl gespeichert? Finden Sie es heraus und ändern Sie diesen Text zu "Hallo Welt!"  Ihre App soll nun so aussehen: |
|  | **res/layout/activity\_main.xml** |

# Einfache Elemente in die Benutzeroberfläche einfügen (4.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können einfache Elemente (Text, Eingabefelder und Schaltflächen) in einem Layout einfügen und sind in der Lage die Eigenschaften dieser Elemente zu bearbeiten. Sie können mit String-Ressourcen umgehen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Eine Benutzeroberfläche besteht nicht nur aus Beschriftungen, die mittels *TextViews* angezeigt werden. Andere Elemente, wie Schaltflächen (*Buttons*) und Eingabefelder (*EditTexts*) sind auch öfter anzutreffen. |
| **Aufgabe** | Öffnen Sie das Projekt **SagHallo**.   * Fügen Sie, wie im Kapitel 4.3 beschrieben, ein Texteingabefeld und einen *Button* unterhalb des *TextViews* "Hello world!" ein. * Ändern Sie den Text "Hello world!" zu "Was soll ich sagen" unter Verwendung einer String-Ressource, die Sie "beschriftung" nennen. * Ändern Sie die Beschriftung des *Buttons* "New Button" zu "Sag es" unter Verwendung einer String-Ressource, die Sie "sag\_es" nennen. * Öffnen Sie die Datei **strings.xml** und suchen Sie nach den beiden neu definierten Strings. Vielleicht möchten Sie ihren Wert anpassen?   Vorlage (ConstraintLayout): Vorlage (LinearLayout): |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. |

# LinearLayout benutzen (4.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können einfache Elemente (Text, Eingabefelder und Schaltflächen) mithilfe eines LinearLayout platzieren. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 20 Min. |
| **Ausgangslage** | Die Topomedics wünscht eine App für die mobile Verwaltung von Serviceaufträgen. In einem ersten Schritt sollen Sie Vorschläge für das Layout präsentieren. Die App soll noch keine Funktionalität haben.  Sie ergreifen die Gelegenheit, um mit der Platzierung von Elementen auf dem Bildschirm herum zu experimentieren. Die richtige Platzierung der Elemente ist von zentraler Bedeutung (Bedienerfreundlichkeit, Flexibilität, Wartbarkeit, zukünftige Anpassungen. Sie trägt zur Akzeptanz einer App bei den Benutzern bei. |
| **Aufgabe** | Erstellen Sie eine neue App **Serviceauftrag** unter Berücksichtigung nachfolgender Anforderungen:   * Company Domain: *Klassenbezeichnung.ch*. * Plattform: **Phone and Tablet** * Activity: **Empty Activity** * Erstellen Sie ein Layout, wie im Anhang beschrieben |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. |

|  |
| --- |
| **Layout 1** (mit *LinearLayout* lösen) |
|  |

# FrameLayout benutzen (4.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können einfache Elemente (Text, Eingabefelder und Schaltflächen) mithilfe eines FrameLayouts platzieren. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Wie bereits gesehen, ist die richtige Platzierung von Elementen auf dem Bildschirm sehr wichtig. Mit den bis jetzt bekannten Möglichkeiten können Sie aber noch nicht alle Situationen meistern. Sie sollen nun Ihre Kenntnisse über Layouts vertiefen und neue Möglichkeiten entdecken. |
| **Aufgabe** | Erstellen Sie eine neue App **FrameLayout** unter Berücksichtigung nachfolgender Anforderungen:   * Company Domain: *Klassenbezeichnung.ch*. * Plattform: **Phone and Tablet** * Activity: **Empty Activity**   Ersetzen Sie den Inhalt von activity\_main.xml mit dem Inhalt der Datei *FrameLayout.txt*. Ihre Aktivität sollte nun wie folgt aussehen:    Ergänzen Sie das Layout, indem Sie 3 *textViews* in das "Component Tree"-Fenster ziehen und beschriften Sie diese ihrer Position entsprechend.  Finden Sie heraus, welche Properties für die Positionierung, Farbe und Schriftgrösse der Elemente zuständig sind und passen Sie diese entsprechend an.  Ändern Sie die Schriftgrösse von "Mitte oben" zu 60sp und beschreiben Sie was sie feststellen.  Treffen Sie die notwendigen Massnahmen, um das *textView* "Mitte oben" hinter "Links oben" und "Rechts oben" zu platzieren. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet * Die Datei *FrameLayout.txt* |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Beschreibung des neuen Aussehens und eine Erklärung dafür. * Abgeänderte App mit dem verlangten Aussehen. |

# Die Eigenschaften Layout\_width und Layout\_height (4.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können mithilfe der Eigenschaften (*Properties*) *layout\_width* und *layout\_height* die Grösse von Elementen, zum Beispiel *textViews*, beeinflussen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Mithilfe von Layouts (u.a. LinearLayout oder eine Kombination dergleichen) können Elemente auf dem Bildschirm platziert werden. Den Platz auf dem Bildschirm, den ein Element einnimmt, bzw. seine Grösse, ist von anderen Eigenschaften abhängig. Die wichtigsten sind *layout\_width* und *layout\_height*. Mit diesen Eigenschaften experimentieren Sie in dieser Aufgabe. |
| **Aufgabe** | Erstellen Sie eine neue App **LayoutEigenschaften\_1** unter Berücksichtigung nachfolgender Anforderungen:   * Company Domain: *Klassenbezeichnung.ch*. * Plattform: **Phone and Tablet** * Activity: **Empty Activity**   Lösen Sie die im Anhang aufgeführten Aufgaben und beantworten Sie die Fragen. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet * Legen Sie alle Layouts in der gleichen App, unter dem Layout Verzeichnis ab. Für das Testen dergleichen, ändern Sie jeweils MainActivity.java entsprechend. |
| **Ergebnis** | * 2 Layouts, die die gestellten Anforderungen erfüllen. * Korrekte Beantwortung der Fragen und Ihre Schlussfolgerungen. |

|  |  |
| --- | --- |
| **Aufgabe 1** | Ändern Sie das ConstraintLayout zu einem vertikalen *LinearLayout*. Entfernen Sie das *textView* "Hello world" und fügen Sie 3 neue *textViews*, wie unten abgebildet. Formatieren Sie jeweils die Elemente über die Eigenschaft *background* mit verschiedenen Farben. Somit wird die Wirkung der untersuchten Eigenschaften besser ersichtlich.    Mit welchem Wert wurden die Eigenschaften *layout\_width* und *layout\_height* der *textViews* standardmässig belegt?  Wie beeinflusst *wrap\_content* die Grösse eines Elements? |
|  | layout\_width: match\_parent; layout\_height: wrap\_content  wrap\_content = es nimmt nur die Höhe ein, welche es wirklich braucht.  Vergleichbar mit display:inline; im HTML |

|  |  |
| --- | --- |
| **Aufgabe 2** | Ändern Sie die Werte von *layout\_width* und *layout\_height* für die Elemente Ihrer Wahl zu *match\_parent*. Machen Sie mehrere Versuche. Alle Elemente sollten berücksichtigt werden.  Wie beeinflusst *match\_parent* die Grösse eines Elements? |
|  | match\_parent heisst, es nimmt dieselbe Grösse wie die des nächstgrösseren. |
| **Aufgabe 3** | Erstellen Sie nachfolgende Oberfläche, indem Sie schon bekannte *textViews* und *editTexts* (Plain Text) verwenden. Platzieren Sie die Elemente mithilfe von **linearen** Layouts.    Testen Sie die App im Simulator oder mit einem realen Gerät. Notieren Sie allfällige Schönheitsfehler bzw. benutzerunfreundliche Gegebenheiten. |
|  | Es ist sehr ungewohnt, Buttons oben am Bildschirm zu haben.  Das Textfeld wird nur langgezogen und vertikal zentriert - es ist kein richtiges Textfeld, sondern nur ein Einzeiliges Feld. |

# Die Eigenschaft layout\_weight (4.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können die Grösse von Elementen mithilfe der Eigenschaft *layout\_weight* in Kombination mit den schon bekannten *layout\_width* und *layout\_height* bestimmen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Den Platz auf dem Bildschirm, den ein Element einnimmt, bzw. seine Grösse, wird u.a. von den Eigenschaften *layout\_width* und *layout\_height* bestimmt. In dieser Aufgabe lernen Sie eine weitere Eigenschaft kennen, womit Sie bestimmen können, wie viel Prozent des zur Verfügung stehenden Platzes ein Element einnimmt. |
| **Aufgabe** | Erstellen Sie eine neue App **LayoutEigenschaften\_2** unter Berücksichtigung nachfolgender Anforderungen:   * Company Domain: *Klassenbezeichnung.ch*. * Plattform: **Phone and Tablet** * Activity: **Empty Activity**   Lösen Sie die im Anhang aufgeführten Aufgaben und beantworten Sie die Fragen. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet * Legen Sie alle Layouts in der gleichen App, unter dem Layout Verzeichnis ab. Für das Testen dergleichen, ändern Sie jeweils MainActivity.java entsprechend. |
| **Ergebnis** | * 2 Layouts, die die gestellten Anforderungen erfüllen. * Übersicht Ihrer Versuche, inkl. Screenshots, mit Schlussfolgerungen. * Korrekte Beantwortung der Frage. |

|  |  |
| --- | --- |
| **Aufgabe 1** | Fügen Sie 4 *textViews* in einem vertikalen *LinearLayout* ein, wie unten abgebildet. Formatieren Sie jeweils die Elemente über die Eigenschaft *background* mit verschiedenen Farben. Ändern Sie den Wert der Eigenschaft *textSize* zu 18sp. Somit wird die Wirkung der untersuchten Eigenschaften besser ersichtlich.    Achten Sie darauf, dass in *layout\_height* jeweils *wrap\_content* steht. |
| **Aufgabe 2** | Ändern Sie die Werte von *layout\_weight* für alle *textViews* und beobachten Sie dabei, wie viel Platz die verschiedenen Elemente auf dem Bildschirm aufnehmen. Finden Sie heraus nach welcher Regel der Platz auf dem Bildschirm aufgeteilt wird. Dokumentieren Sie Ihre Versuche und Schlussfolgerungen mit ScreenShots.  Mögliche Werte für die verschiedenen *textViews* (als Anstoss für weitere Versuche):  Erster Text Zweiter Text Dritter Text Vierter Text  1 1 1 1  1 2 2 1  2 4 2 4  **Tipp:**  Es lohnt sich, die Experimente selber durchzuführen. Im Internet sind viele falsche Erklärungen bezüglich dieser Eigenschaft!  Was spielt das Gewicht 0 für eine Rolle? |
|  | Gewicht 0 = Es nimmt genau die Grösse ein, die sie effektiv braucht |
| **Aufgabe 3** | Erstellen Sie ein neues Layout, bestehend aus je einem *textView*, *editText* und *Button*. Achten Sie auf die Platzierung der Elemente sowohl in Hoch- wie im Querformat. |

# Benutzerfreundliche EditTexts (4.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können je nach Zweck eines *EditTexts* den entsprechenden Typ auswählen, um dem Benutzer u.a. die passende Tastatur zu aktivieren und einzublenden. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Wir nehmen es kaum wahr, aber die meisten App blenden je nach Eingabeart verschiedene Tastaturen ein. Wir sind einfach daran gewöhnt, dass es "passt".  In einer früheren Übung konnten Sie feststellen, dass die Wahl des *EditTexts* *Plain Text* für die Eingabe eines mehrzeiligen Textes, in jenem Fall "Durchgeführte Arbeiten", nicht optimal ist. Die Position des eingegebenen Textes, in der Mitte des Feldes, ist für einen Benutzer verwirrend und unpraktisch. Wir sind einfach nicht daran gewöhnt.  In dieser Übung werden Sie diese Punkte berücksichtigen und Ihre App benutzerfreundlich einrichten. |
| **Aufgabe** | Erstellen Sie eine neue App **Tastaturlayout** unter Berücksichtigung nachfolgender Anforderungen:   * Company Domain: *Klassenbezeichnung.ch*. * Plattform: **Phone and Tablet** * Activity: **Empty Activity**   Lösen Sie die im Anhang aufgeführten Aufgaben und beantworten Sie die Fragen. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * Ein Layout, das die gestellten Anforderungen erfüllt. * Korrekte Beantwortung der Frage. * Übersicht Ihrer Versuche, inkl. Screenshots, mit Schlussfolgerungen. |

|  |  |
| --- | --- |
| **Aufgabe 1** | Erstellen Sie nachfolgende Oberfläche (email-Formular), indem Sie schon bekannte *TextViews* und *EditTexts* verwenden. Platzieren Sie die Elemente mithilfe eines **linearen** Layouts.    Lösen Sie zuerst das Problem mit dem Nachrichtentext. Der eingetippte Text soll, wie wir es alle erwarten, oben links im Feld erscheinen.  Die benötigte Eigenschaft heisst ***Gravity***.  Wie würden Sie in diesem Kontext Gravity übersetzen? |
|  | Gravity richtet das Objekt an einen bestimmten Ort aus, auch wenn das Feld an sich grösser ist |
| **Aufgabe 2** | Experimentieren Sie mit der Eigenschaft *inputType* und finden Sie heraus, welcher Wert sich jeweils am besten für die Eingabe der Email-Adresse, des Betreffs und der Nachricht eignet.  Notieren Sie welchen *inputType* Sie je *EditText* ausgewählt haben. Begründen Sie Ihre Wahl und untermauern Sie Ihre Aussagen mit Screenshots, welche die Tastatur zeigt, die jeweils eingeblendet wird. |
|  | E-Mail = textEmailAdress ; Betreff = textEmailSubject ; Nachricht = textMultiLine |
| **Aufgabe 3** | Sie möchten, dass die Sätze in dem Nachrichtentext automatisch mit einem Grossbuchstaben anfangen. Innerhalb des Textes sollen neue Zeilen (mit der Enter-Taste) möglich sein. Nehmen Sie die notwendigen Einstellungen vor. |
|  | textCapSentences |

# Benutzereingaben verarbeiten (4.4)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können den *onClickListener* eines *Buttons* implementieren und damit auf Benutzerinteraktion reagieren. Sie sind in der Lage, Benutzereingabe zu verarbeiten. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 20 Min. |
| **Ausgangslage** | Das Layout für die App **SagHallo** steht bereits. Jetzt soll die App aber nicht immer den fest programmierten (hardcoded) Text ausgeben, sondern den, den der Benutzer eingibt. Dafür muss der schon vorhandene *Button* eine entsprechende Methode aufrufen, wenn der Benutzer ihn anklickt. Diese Methode liest den Text aus dem Eingabetext und startet die Sprachausgabe. |
| **Aufgabe** | Öffnen Sie das Projekt **SagHallo**.  Ergänzen Sie das Programm, wie im Buch beschrieben (Kapitel 4.4).  Beantworten Sie die Fragen im Anhang. |
| **Hinweis Hilfsmittel** | * Lehrmittel |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Korrekte Beantwortung der Fragen. |

|  |  |
| --- | --- |
| **Frage 1** | Warum wurde in diesem Beispiel ***this*** als Argument verwendet?  button.setOnClickListener(**this**); |
|  |  |
| **Frage 2** | Wie heisst die Methode, womit Sie den Inhalt eines *EditTexts* auslesen können? |
|  | *editText.getText()* |
| **Frage 3** | Warum wird in diesem Beispiel die Methode *toString()* benötigt? Wir haben doch einen Text in dem Eingabefeld eingetippt.  **tts**.speak(editText.getText().**toString()**, … |
|  | editText gibt kein String zurück, sondern den Typ editText.  toString wandelt den editText also in einen String ein |

# Aktivität mit mehreren Buttons (4.4)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können eine Aktivität programmieren, die unterscheiden kann, welcher Button angetippt wurde. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Wie man eine Methode programmiert, die aufgerufen wird, wenn der entsprechende Button angeklickt wird, wissen Sie bereits. In dieser Übung sollen Sie herausfinden, wie das Ganze funktioniert, wenn 2 (oder mehrere) Buttons in der View sind. |
| **Aufgabe** | Erstellen Sie eine neue App **Ereignisse** nach der bewährten Methode.  Fügen Sie ein *TextView* und 2 *Buttons*, wie unten abgebildet.  Wenn der Benutzer auf den Button "gut" klickt, soll der Text "Das freut uns!" ausgegeben werden. Ein Klick auf den Button "Nicht so gut" soll den Text "Schade. Wir hoffen, es wird sich noch ändern!" ausgeben. |
| **Hinweis Hilfsmittel** | * Die App soll nur demonstrieren, wie man die Ereignisse von mehreren Buttons unterscheidet und verarbeitet. Zur Kontrolle reicht es, einen kurzen Text auf dem Bildschirm einzublenden. Dafür eignen sich so genannte Toasts hervorragend. Beispiel:   Toast.*makeText*(getApplicationContext(), **"Schaltfläche wurde gedrückt!"**, Toast.***LENGTH\_LONG***).show();   * Es sind mehrere verschiedene Lösungswege bzw. Ansätze möglich. * Lehrmittel * Internet |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. |

# Eine App installieren (4.5)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie sind in der Lage eine App weiterzugeben. Zum Beispiel einem Kunde oder Freund. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Wie man eine App auf einem Smartphone mithilfe des USB-Kabels und AS installiert wissen Sie bereits. Wie können Sie aber Ihre Apps weitergeben? Zum Beispiel an die Mitarbeiter der Topomedics, die in den Filialen arbeiten. Dafür sind verschiedene Wege möglich. Den Weg über Google Play können wir noch nicht verwenden. |
| **Aufgabe** | Öffnen Sie ein Projekt Ihrer Wahl.  Verschaffen Sie sich eine Übersicht über die verschiedenen Methoden.  Entscheiden Sie sich für eine Methode, die Sie dann austesten und mit Screenshots dokumentieren. Zum Beispiel Installation über DropBox.  Beantworten Sie die Frage weiter unten. |
| **Hinweis Hilfsmittel** | * Lehrmittel (Kapitel 4.5) * Internet |
| **Ergebnis** | * Korrekte Beantwortung der Frage. * Übersichtstabelle mit mindestens 5 Methoden, eine App zu installieren. * Die App wurde auf einem anderen Smartphone installiert, ohne das AS zu verwenden. * Dokumentation der ausgetesteten Methode. |

|  |  |
| --- | --- |
| **Frage 1** | Warum können Sie diese App so nicht über Google Play veröffentlichen, auch wenn Sie ein Konto haben/hätten? |
|  |  |

# Spiel entwickeln (1) (5.1)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können eine App entwickeln, die aus mehreren Aktivitäten und Layouts besteht. Sie sind in der Lage eine Aktivität programmatisch zu starten. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Für das Spiel **Mueckenfang** benötigen wir ein Layout für die Startseite und ein Layout für das eigentliche Spiel. Diese Layouts werden von 2 verschiedenen Aktivitäten aktiviert. Diese Layouts und Aktivitäten programmieren Sie in dieser Übung. |
| **Aufgabe** | Erstellen Sie eine neue App **Mueckenfang** unter Berücksichtigung nachfolgender Anforderungen:   * Company Domain: *Klassenbezeichnung.ch*. * Plattform: **Phone and Tablet** * Activity: **Empty Activity**   Schreiben Sie das Programm, wie im Buch beschrieben (Kapitel 5.1).  Beachten Sie dabei, dass Sie die Grösse des *TextViews* "Willkommen beim Mückenfang" noch nicht anpassen.  +  Testen Sie die App im Simulator oder auf Ihrem Gerät.  Lösen Sie die Aufgaben im Anhang und beantworten Sie die Fragen. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Korrekte Beantwortung der Fragen. |

|  |  |
| --- | --- |
| **Aufgabe 1** | Entfernen Sie vorübergehend den Eintrag ***<activity android:name=".GameActivity" />*** aus dem Android-Manifest und testen Sie erneut Ihre App.  Wie heisst die Fehlermeldung auf dem Gerätebildschirm, wenn Sie START antippen?  Wie heisst die Fehlermeldung in ***logcat***? |
|  |  |
| **Aufgabe 2** | Fügen Sie den gelöschten Eintrag im Android-Manifest wieder ein. |

# Weitere Aktivitäten starten (5.1)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie wissen genau was passiert, wenn Sie mithilfe eines *Intent*-Objekts eine neue Aktivität starten. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Mit der Erzeugung eines neuen *Intent*-Objekts können Sie eine neue Aktivität starten. Wenn eine neue Aktivität gestartet wird, wird die Ausgangsaktivität aber nicht automatisch beendet. Dies muss jedem bewusst sein, der Apps entwickelt. Sonst ist Ärger mit dem App und Android vorprogrammiert! |
| **Aufgabe** | Erstellen Sie eine neue App **Intents** nach der bewährten Methode.  Die App soll nachfolgende Anforderungen erfüllen:   * Auf der ersten Seite (Hauptseite) ist ein Button "Seite 2". Ein Klick darauf öffnet eine zweite Seite. * Auf der zweiten Seite ist ein Button "Seite 3". Ein Klick darauf öffnet eine dritte Seite.   🡪  🡪    Lösen Sie die Aufgaben im Anhang und beantworten Sie die Frage. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Korrekte Beantwortung der Frage. |

|  |  |
| --- | --- |
| **Frage 1** | Warum wäre es keine gute Idee, die Seiten 2 und 3 mit einem eigenen Button "Zurück 2" bzw. "Zurück 1" zu versehen? Beweisen Sie Ihre Antwort, indem Sie Ihre App entsprechend anpassen. |
|  | Nein, weil der Layer nicht geschlossen werden.  Die Layers werden die ganze Zeit aufeinander gesetzt |

# Grafiken einbinden (5.2)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können grafische Elemente in einem Layout einfügen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 20 Min. |
| **Ausgangslage** | Für den Hauptbildschirm des Mückenspiels benötigen wir das Bild einer Mücke und ein passendes Hintergrundbild. Beide Bilder sind feste Bestandteile des Layouts. Wie Mücken programmatisch und zufällig im Spiellayout erscheinen können, ist Ziel einer weiteren Übung. |
| **Aufgabe** | Öffnen Sie das Projekt **Mueckenfang**.  Ergänzen Sie das Programm, wie im Buch beschrieben (Kapitel 5-2).  Testen Sie Ihre App mit verschiedenen Devices und Ausrichtungen. Verwenden Sie dafür nachfolgende Symbole im Design-Fenster oder arbeiten Sie mit mehreren Emulatoren:    Halten Sie Ihre Beobachtungen fest und machen Sie Vorschläge, wie Sie beobachtete Nachteile beheben würden. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Dateien: *hintergrund.jpg* und *muecke.png* |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * 2-3 gemachte Beobachtungen mit Verbesserungsvorschlägen. |

# Grösseneinheiten (5.2)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie kennen die verschiedenen Grösseneinheiten für die Darstellung von Elementen und können diese gezielt und situationsgerecht verwenden. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Für die Darstellung von Texten oder Bildern, oder für die Platzierung von Elementen auf dem Bildschirm werden verschiedene Grösseneinheiten verwendet. Diese unterschiedlichen Eigenschaften verhalten sich nicht gleich. Es ist daher unabdingbar für einen App-Entwickler diese Eigenschaften zu kennen und gezielt zu verwenden. Ansonsten sind böse Überraschungen vorprogrammiert. |
| **Aufgabe** | Erstellen Sie eine neue App **GroessenEinheiten** nach der bewährten Methode.  Wählen Sie im Design-Fenster ein hdpi-Device (240 dpi) aus. Z.B.: Nexus S.  Fügen Sie 3 *TextViews* ein. Der jeweils angezeigte Text soll dem Wert in der Eigenschaft *textSize* entsprechen:    Wählen Sie nun andere Geräte, mit unterschiedlichen *screen densities*, aus. Beobachten Sie, wie die Grössen der *textViews* sich anpassen und dokumentieren Sie dies mit Screenshots.  Starten Sie nun die App in einem AVD, zum Beispiel Nexus 5X (420dpi).  Wählen Sie verschiedene Benutzer Schriftgrössen (Einstellungen / Display / Schriftgrösse) aus. Beobachten Sie, wie die Grössen der *TextViews* sich verhalten. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Zusammenfassung Ihrer Beobachtungen und 2-3 Regeln, die das Verhalten der verwendeten Einheiten beschreiben. |

# Game Engine aufbauen (5.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können erklären, wie das Spielgerüst mit seinen Methoden funktioniert. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 20 Min. |
| **Ausgangslage** | Die wichtigsten Komponenten einer Game Engine sind:   * Spielzustand (Rundenummer, erzielte Punkte, Restzeit, usw.) * Spiel starten * Runde starten * Bildschirm mit Spielzuständen aktualisieren   In dieser Übung werden Sie einige Werte initialisieren und die erste Version der Hauptmethoden implementieren. |
| **Aufgabe** | Öffnen Sie die App **Mueckenfang**.  Kopieren Sie den Text aus der Datei *Game Engine 1.txt* und integrieren Sie diesen in der Klasse *GameActivity* nach onCreate().  Die Methoden sind korrekt programmiert. Trotzdem lässt sich das Programm leider nicht kompilieren (siehe rot markierte Ausdrücke). Ergänzen Sie die Klasse entsprechend, um diese Fehlermeldungen verschwinden zu lassen und das Programm kompilierbar zu machen. Fehlerhafte Zeilen zu löschen oder auszukommentieren ist selbstverständlich nicht die Lösung.  Fügen Sie an geeigneten Stellen Anweisungen ein, die es erlauben, den Programmablauf zu verfolgen und zu kontrollieren. Zum Beispiel:  *System.out.println(…*  *Toast.makeText(…*  *Log.d(…*  Ergänzen Sie das Programm mit Kommentaren. Zum Beispiel, um den Zweck einer Methode (anhand der bisherigen Anweisungen) und der verschiedenen Attribute zu erläutern. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Die Datei *Game Engine 1.txt* |
| **Ergebnis** | * Funktionstüchtige App. * Kommentare an den im Code markierten Stellen. |

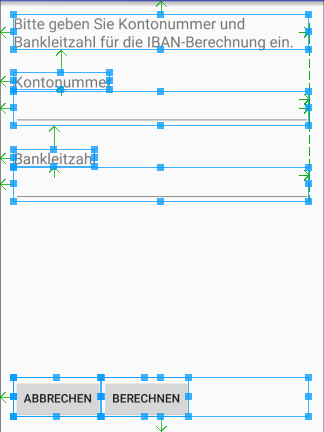
# Layout für das Mückenspiel einrichten (5.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können mithilfe bekannter Elemente und deren Eigenschaften das Layout für ein Spiel legen. Sie sind in der Lage ein Layout zwecks Dokumentation zu skizzieren. |
| **Aufgabentyp** | Lernaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Bevor wir das Spiel programmieren, müssen wir die Bildschirmelemente für die Ausgabe der aktuellen Punktzahl, Nummer der aktuellen Runde, Anzahl gefangener und noch zu fangender Mücken und der verbleibenden Zeit bis zum Ende einer Runde im Layout platzieren und formatieren. Wie das Layout aussieht, soll in einer Skizze dokumentiert werden. Diese Skizze könnte bei zukünftigen Spielerweiterungen bzw. Änderungen wichtig sein und helfen Zeit zu sparen. |
| **Aufgabe** | Öffnen Sie das Projekt **Mueckenfang**.  Ergänzen Sie das Programm, wie im Buch beschrieben (Kapitel 5.3.4).  Die Tabelle im Anhang gibt Ihnen einen Überblick über den Aufbau des Spiellayouts und sollte dabei Ihre Arbeit erleichtern. Beachten Sie auch das *Component Tree* auf Seite 151.  Gewünschtes Aussehen:    Zeichnen Sie zur Dokumentation eine Skizze des angefertigten Layouts.  Siehe auch Beispiel im Anhang (mit einem anderen Layout). |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * Spiellayout mit dem gewünschten Aussehen. * Skizze des Spiellayouts. |

Eigenschaften (im Buch sind nicht alle erwähnt!):

|  |  |  |
| --- | --- | --- |
| **Element** | **Eigenschaft** | **Wert** |
| LinearLayout (vertical) |  |  |
| FrameLayout | layout\_height | wrap\_content |
| TextView | id | points |
|  | layout\_width | wrap\_content |
|  | layout\_gravity | top, left |
|  | textSize | 18sp |
|  | textColor | #64dd17 *(Beispiel)* |
|  | textStyle | bold |
| TextView | id | round |
|  | layout\_width | wrap\_content |
|  | layout\_gravity | top, right |
|  | textSize | 18sp |
|  | textColor | #64dd17 *(Beispiel)* |
|  | textStyle | bold |
| FrameLayout | id | spielbereich |
|  | layout\_weight | 1 |
| LinearLayout | id |  |
|  | layout\_height | wrap\_content |
|  | layout\_gravity | center\_horizontal |
| FrameLayout |  |  |
| FrameLayout | id | bar\_hits |
|  | layout\_width | 50dp |
|  | layout\_height | 5dp |
|  | layout\_gravity | center\_vertical |
|  | background | #d50000 *(Beispiel)* |
| TextView | id | hits |
|  | layout\_width | wrap\_content |
|  | layout\_gravity | right |
|  | textColor | #d50000 *(Beispiel)* |
| FrameLayout |  |  |
| FrameLayout | id | bar\_time |
|  | layout\_width | 50dp |
|  | layout\_height | 5dp |
|  | layout\_gravity | center\_vertical |
|  | background | #18ffff *(Beispiel)* |
| TextView | id | time |
|  | layout\_gravity | right |
|  | textColor | #18ffff *(Beispiel)* |

Beispiel für die Skizze eines Layouts:



LinearLayout (horizontal)

button

Button2

editText2

textView3

editText

textView2

textView

RelativeLayout

# Methode *Bildschirmaktualisieren* (5.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können den Wert eines *TextViews* programmatisch ändern. Sie sind in der Lage Informationen über den verwendeten Bildschirm zu ermitteln. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Die *TextViews*, die Sie in der letzten Übung im Layout eingefügt haben, müssen während des Programmablaufs (Spiel) aktualisiert werden. Zum Beispiel die momentane Punktzahl oder die aktuelle gespielte Runde. Die Breite beider FrameLayouts (bar\_time und bar\_hits) muss laufend proportional zu den anzuzeigenden Werten angepasst werden. Um die richtige Breite zu bestimmen, brauchen Sie zu wissen, wie breit der Bildschirm real ist. |
| **Aufgabe** | Öffnen Sie das Projekt **Mueckenfang**.  Ergänzen Sie das Programm, wie im Buch beschrieben (ab Seite 151).  Testen Sie Ihre App, indem Sie der Variablen gefangeneMuecken den Wert 5 zuweisen. In der ersten Runde des Spiels werden 1 Runde \* 10 = 10 Mücken angezeigt. Somit soll der Balken 50% der maximalen Länge lang sein: |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * App mit der gewünschten Funktionalität |

# Informationen über das eingesetzte Smartphone abrufen (5.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie sind in der Lage Informationen über das vom Benutzer eingesetzte Smartphone zu ermitteln. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | In der Methode *Bildschirmaktualisieren* haben Sie die Methoden *getDisplay.Metrics()* und *getLayoutParams()* kennen gelernt. Damit können Sie Informationen über den Bildschirm und Layout abfragen. In dieser Übung werden Sie noch mehr Informationen über den Bildschirm holen. Sie sollen auch, mithilfe noch unbekannter Methoden, Informationen über das System (Locale, Version SDK und Release) abfragen. |
| **Aufgabe** | Erstellen Sie eine neue App **Eigenschaften** nach der bewährten Methode.  Sie soll nachfolgende Informationen über das Smartphone, in dem sie gerade läuft, sammeln und anzeigen. Zum Beispiel wie folgt: |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * App mit der gewünschten Funktionalität. |

# Der erste Mückenfang (5.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können grafische Elemente (Bilder) programmatisch einfügen und wieder entfernen. Sie sind in der Lage, Methoden zeitgesteuert, in regelmässigen oder zufälligen Abstanden, auszuführen. Sie können Objekte mit *Tags* versehen, um Daten an diese Objekte zu "kleben" und später wieder ablesen zu können. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Das Layout für das Mückenspiel steht schon. Was fehlt ist die Programmlogik. Also das automatische Erscheinen von Mücken an zufälligen Bildschirmstellen. Das Verschwinden dergleichen nach einer bestimmten Zeit. Das Zählen der angetippten (getöteten) Mücken und Aktualisierung der Anzeige (Punktzahl, Nummer der aktuellen Runde, restliche Zeit, …) |
| **Aufgabe** | Öffnen Sie das Projekt **Mueckenfang**.  Ergänzen Sie das Programm, wie im Buch beschrieben (Seiten 154-177). In der Datei *der\_erste\_Mueckenfang.txt* finden Sie die entsprechenden Methoden und einige Einzelzeilen. In dieser Datei haben sich ein paar wenige Fehler eingeschlichen, die Sie zuerst beheben müssen.  Analysieren Sie das Programm und beantworten Sie die Fragen im Anhang.  Lesen Sie die Erklärungen im Buch sorgfältig. Sie müssen in der Lage sein, nach Beendigung der Aufgabe, den Code zu erklären. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Die Datei *der\_erste\_Mueckenfang.txt* |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Korrekte Beantwortung der Fragen. |

|  |  |
| --- | --- |
| **Frage 1** | Was passiert, wenn Sie die Zeile **handler**.postDelayed(**this**,…); in der Methode *zeitHerunterzaehlen()* vergessen bzw. auskommentieren? Erklären Sie warum. |
|  |  |
| **Frage 2** | Erklären Sie wortwörtlich, was diese beiden Zeilen machen: |
|  |  |
| **Frage 3** | Was wird genau in alter gespeichert?  **long** alter = (**new** Date()).getTime() - geburtsdatum.getTime(); |
|  |  |
| **Frage 4** | Mit welcher Methode können Sie ein *View* aus einer *ViewGroup* entfernen? |
|  |  |
| **Frage 5** | Woher kommt die Zahl 50 in:  **int** muecke\_breite = (**int**) Math.*round*(**massstab**\*50);  Wie können Sie dies verifizieren? |
|  |  |
| **Frage 6** | Erklären Sie mit welchem Trick eine Mücke automatisch nach einer Weile wieder vom Bildschirm verschwindet. |
|  |  |

# Sound abspielen (6.2)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können mit Hilfe des *MediaPlayers* eine Tondatei abspielen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 20 Min. |
| **Ausgangslage** | Mit der Mueckenfang App kann man jetzt schon spielen bzw. Mücken fangen. Diese Mücken sollen aber uns nerven, indem Sie wie echte Mücken summen. |
| **Aufgabe** | Öffnen Sie das Projekt **Mueckenfang**.  Ergänzen Sie das Programm, wie im Buch beschrieben (Kap. 6.2). |
| **Hinweis Hilfsmittel** | * Lehrmittel |
| **Ergebnis** | * App mit der gewünschten Funktionalität. |

# Lebenszyklus einer Aktivität (6.2)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie kennen die verschiedenen Zyklen (*cycles*) einer Aktivität und können bei der Programmierung einer App die entsprechenden Methoden sinnvoll verwenden. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | In bisherigen Aufgaben haben Sie mit der bekannten Methode *onCreate()* gearbeitet. In dieser Methode bestimmen Sie, was passieren soll, wenn die Aktivität instanziert wird. Neu haben Sie in der letzten Aufgabe die Methode *onDestroy()* kennengelernt. In dieser bestimmen Sie, was passieren soll bevor die Aktivität aus dem Speicher gelöscht wird. In dieser Aufgabe werden Sie sehen, dass es noch weitere Methoden gibt, die zum so genannten *Lifecycle* einer Aktivität gehören. Es ist sehr wichtig diese Phasen zu kennen. In der nächsten Aufgabe werden Sie gezielt mit 1-2 davon arbeiten, um ein typisches Problem zu lösen. Aber vorerst bitte Geduld. |
| **Aufgabe** | Erstellen Sie eine neue App **Lifecycle** nach der bewährten Methode.  Ergänzen Sie die *onCreate()* Methode mit nachfolgendem Inhalt:  @Override  **protected void** onCreate(Bundle savedInstanceState) {  …  Log.*d*(**"Test: "**, **"onCreate"**);  }  Lösen Sie die Aufgaben im Anhang und beantworten Sie die Fragen. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet * <https://developer.android.com/training/basics/activity-lifecycle/starting.html> * <https://developer.android.com/reference/android/app/Activity.html> |
| **Ergebnis** | * App mit der gewünschten Funktionalität. * Korrekte Beantwortung der Fragen. |

|  |  |
| --- | --- |
| **Aufgabe 1** | Fügen Sie nachfolgende Methoden ein:  *onStart()*, *onRestart()*, *onResume()*, *onPause()*, *onStop()* und *onDestroy()*  Jede Methode, soll analog wie in *onCreate()* einen Eintrag in ***logcat*** erzeugen.  Testen Sie die App, indem Sie nachfolgende Manipulationen durchführen. Notieren Sie jeweils, welche Methoden in welcher Reihenfolge ausgeführt werden. |
|  |  |
| **Aufgabe 2** | **Lifecycle** starten (mehr nicht). |
|  |  |
| **Aufgabe 3** | **Lifecycle** starten und danach beenden (Mit der Zurück Taste). |
|  |  |
| **Aufgabe 4** | **Lifecycle** starten und danach den Bildschirm drehen. |
|  |  |
| **Aufgabe 5** | **Lifecycle** starten und danach eine andere App starten (Mit der Home Taste und danach Anwendungen). |
|  |  |
| **Aufgabe 6** | **Lifecycle** starten und danach eine andere App starten (Mit der Home Taste und danach Anwendungen). Anschliessend die App **Lifecycle** wieder aktivieren. |
|  |  |

# Problem mit dem Lebenszyklus (6.2)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können den Zustand einer Aktivität retten und beim neuen Start wieder herstellen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Sie wissen bereits, dass eine Aktivität während ihrer Existenz mehrere Phasen durchläuft. Als Programmierer können Sie gezielt einen Code schreiben, der innerhalb einer dieser Phasen ausgeführt wird. Typisches Beispiel dafür ist der Aufbau des Bildschirms in der ersten Phase. Der entsprechende Code wird demzufolge in der Methode *onCreate()* platziert. Aufräumarbeiten werden meistens in *onDestroy()* gemacht. Sie wissen auch, dass je nachdem was der Benutzer mit seinem Smartphone macht, die Aktivität automatisch in eine andere Phase wechselt. Bei einem Orientierungswechsel (hoch, quer) wird die Aktivität sogar zerstört und neu instanziert. Was passiert aber mit den Einträgen in Textfeldern und den Variablenwerten? In dieser Übung werden Sie dies untersuchen. |
| **Aufgabe** | Öffnen Sie das Projekt **Lifecycle\_TN**.    Testen Sie nun die App, indem Sie ein paar Mal die Schalfläche KLICK! antippen und Ihren Namen unter Tester schreiben. Drehen Sie dann den Bildschirm. Beschreiben Sie, was Sie sehen bzw. feststellen.  Finden Sie eine Lösung, um das Problem zu beheben. |
| **Hinweis Hilfsmittel** | * Internet (Nach *onSaveInstanceState()* googeln) |
| **Ergebnis** | * Beschreibung des Verhaltens der App * Lösung für das festgestellte Problem |

# Daten (Highscores) lokal speichern (7.1)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können Aktivitäten starten, die einen Rückgabewert liefern. Sie sind in der Lage Daten lokal in den *Shared Preferences* zu speichern und wieder abzurufen. Sie lernen, wie Sie Views unsichtbar und bei Bedarf wieder sichtbar machen können. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Wir wollen nun den bisherigen Spielrekord und den Namen des Rekordhalters dauerhaft speichern und auf der Startseite anzeigen. |
| **Aufgabe** | Öffnen Sie die App **Mueckenfang**.  Fügen Sie, wie im Kapitel 7.1 beschrieben, zwei *TextViews* auf dem Startbildschirm ein:    Ergänzen Sie das Programm, inklusive den Teil "Werte permanent speichern" wie im Buch beschrieben. Lesen Sie die Erklärungen sorgfältig durch. Sie müssen in der Lage sein, nach Beendigung der Übung, den Code zu erklären.  Ergänzen Sie das Programm mit dem Teil "Rekordhalter verewigen":    Testen Sie Ihre App.  Finden Sie heraus, wie Sie die gespeicherten Daten löschen können, um wieder bei Null anzufangen. Dies muss nicht programmatisch erfolgen. |
| **Hinweis Hilfsmittel** | * Lehrmittel |
| **Ergebnis** | * App mit der gewünschten Funktionalität. * Korrekte Beantwortung der Fragen. |

# Aktivität mit mehreren Rückgabewerten (7.1)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können eine Aktivität programmieren, die mehrere Werte an die aufrufende Aktivität zurückgibt. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Sie wissen bereits, wie man eine Aktivität starten kann, die einen Wert zurückgibt. In dieser Übung finden Sie heraus, wie eine Aktivität mehrere Werte zurückgeben kann. Es soll die Punktzahl und die Nummer der Runde zurückgegeben werden. |
| **Aufgabe** | Öffnen Sie die App **Mueckenfang**.  Kommentieren Sie die vorhandenen Methoden onActivityResult() und gameOver() vorübergehend aus. Löschen bzw. bearbeiten Sie sie nicht. Sie werden in den nachfolgenden Übungen weiter gebraucht.  Erstellen Sie je eine neue Version von onActivityResult() und gameOver(). Die Methode gameover() soll neu 2 Werte zurückgeben: Die Anzahl Punkte und die Nummer der Runde. Die Methode onActivityResult() soll lediglich die entsprechenden Daten auf dem Bildschirm (Toast) oder im Logcat zeigen. Es wird nicht verlangt, dass die Daten gespeichert werden. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * App mit der gewünschten Funktionalität. |

# Daten (Highscores) online speichern (7.2)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können mit Datenströmen arbeiten und Daten im Internet speichern. Sie sind in der Lage *Background-Threads* zu starten und *Exceptions* zu behandeln. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 50 Min. |
| **Ausgangslage** | Alleine spielen wird irgendwann langweilig. Wir werden in dieser Übung die eigenen Highscores mit anderen teilen. Eine Liste der besten Spieler wird online, auf einem extra dafür eingerichteten Server gespeichert. |
| **Aufgabe** | Öffnen Sie die App **Mueckenfang**.  Fügen Sie den Inhalt der Datei *Scrollview\_bestenliste.txt* an das Ende von *main.xml*, vor *</LinearLayout>,* ein. Die resultierende Fehlermeldung lässt sich schnell beheben ;-). So soll es im unteren Teil des *Component Tree* Fensters aussehen:    Fügen Sie den Inhalt der Datei *HTTP\_Client.txt* an das Ende von *MueckenfangActivity.java* ein.  Fügen Sie am Anfang der Klasse ein:  **private static final** String ***HIGHSCORE\_SERVER\_BASE\_URL*** = **"http://www.pcd-soft.ch/mueckenpcd/resultate.php"**;  **private** String **highscoresHtml**;  Fügen Sie noch am Schluss von *onResume()* folgende Zeile ein:  internetHighscores(**""**,0);  Und am Schluss von *onClick()* des *Buttons* Speichern:  internetHighscores(leseHighscoreName(), leseHighscore());  Fügen Sie nachfolgende Zeile in die Manifestdatei ein:  <**uses-permission android:name="android.permission.INTERNET"**/>  Testen Sie Ihr Programm und lösen Sie die Aufgaben im Anhang. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet * Die Dateien *Scrollview\_bestenliste.txt* und *HTTP\_Client.txt* |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Korrekte Beantwortung der Fragen |

|  |  |
| --- | --- |
| **Aufgabe 1** | Wenn Sie versuchen Ihre App zu starten wird diese Fehlermeldung eingeblendet:    und Nachfolgendes im *logcat* eingetragen:  04-24 15:52:43.782 4061-4075/? E/AndroidRuntime: FATAL EXCEPTION: Thread-236  Process: ch.ifz424.mueckenfang\_72, PID: 4061  java.lang.SecurityException: Permission denied (missing INTERNET permission?)  Erklären Sie. |
|  |  |
| **Aufgabe 2** | Der grösste Teil der Methode *run()* wurde in einer *try-catch*-Konstruktion eingebaut:  try {  } catch (IOException e) {  }  Ist dies in dieser Situation eine Notwendigkeit oder einfach guter Programmierstil? Begründen Sie Ihre Antwort. |
|  |  |
| **Aufgabe 3** | Für den Zugriff aufs Internet wurde ein neuer Background-Thread gestartet:  **private void** internetHighscores(**final** String name, **final int** points){  (**new** Thread(**new** Runnable()  {  @Override  **public void** run(){  Ist das notwendig? Was könnte passieren, wenn man dies nicht machen würde? Begründen Sie Ihre Antwort. |
|  |  |
| **Aufgabe 4** | Warum wurde im Hintergrund-Thread für den Internetzugriff nochmals ein Thread gestartet?  runOnUiThread((**new** Runnable() {  @Override  **public void** run() {  TextView tv = findViewById(R.id.***highscores***);tv.setText(Html.*fromHtml*(**highscoresHtml**));} }));  Hätten wir nicht einfach den *Textview* *highscores* direkt verändern können? |
|  |  |
| **Aufgabe 5** | Erklären bzw. kommentieren Sie nachfolgenden Codeausschnitt:  InputStreamReader input =  **new** InputStreamReader(conn.getInputStream(), **"UTF-8"**); |
|  |  |
| **Aufgabe 6** | Erklären bzw. kommentieren Sie nachfolgenden Codeausschnitt:  BufferedReader reader = **new** BufferedReader(input, 2000); |
|  |  |
| **Aufgabe 7** | Erklären bzw. kommentieren Sie nachfolgenden Codeausschnitt:  List<String> highscoreList = **new** ArrayList<String>(); |
|  |  |
| **Aufgabe 8** | Erklären bzw. kommentieren Sie nachfolgenden Codeausschnitt:  String line = reader.readLine(); |
|  |  |
| **Aufgabe 9** | Erklären bzw. kommentieren Sie nachfolgenden Codeausschnitt:  **while** (line != **null**)  {  highscoreList.add(line);line = reader.readLine();  } |
|  |  |
| **Aufgabe 10** | Erklären bzw. kommentieren Sie nachfolgenden Codeausschnitt:  **highscoresHtml** = **""**; **for** (String s : highscoreList) {  **highscoresHtml** += **"<b>"** + s.replace(**","**, **"</b> <font**  **color='silver'>"**) + **"</font><br>"**; } |
|  |  |

# Liste mit Adapter (7.3)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können lange Listen mit komplexem Inhalt mithilfe von *ListViews* unter Verwendung eines Adapters professionell darstellen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Die bisherige Liste mit den Highscores sieht nicht gerade einladend aus. Dies wollen wir nun ändern:  Jetzt: Nachher: |
| **Aufgabe** | Öffnen Sie die App **Mueckenfang**.  Ziehen Sie den *TextView* highscores ausserhalb des *ScrollViews* heraus. Löschen Sie den *ScrollView* und fügen Sie an seiner Stelle einen *ListView* ein. Seine Eigenschaften: *layout\_width="match\_parent"* *Layout\_height 0dp* und *Layout\_weight 1* und *cacheColorHint #00000000*  Kopieren Sie die Datei *toplist\_element.xml* in das Layout-Verzeichnis und fügen Sie den Code von *ToplistAdapter.txt* als innere Klasse in *MueckenfangActivity.java* ein.  Weitere Anpassungen (Deklaration von Attributen und Überarbeitung von *internetHighscores()* entnehmen Sie bitte dem Buch.  Testen Sie Ihre App und beantworten Sie die Fragen im Anhang. |
| **Hinweis Hilfsmittel** | * Lehrmittel, Internet * Die Dateien *toplist\_element.xml* und *ToplistAdapter.txt* |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Korrekte Beantwortung der Fragen |

|  |  |
| --- | --- |
| **Frage 1** | Wie entsteht die Zahl für die Anzeige des Rangs? |
|  |  |
| **Frage 2** | Was macht diese Zeile genau?  sss.setString(**highscoreList**.get(position)); |
|  |  |
| **Frage 3** | Mit welcher Befehlszeile wird der Name des Spielers in der aktuellen Zeile (*convertView*) angezeigt und wie "weiss" diese Befehlszeile, dass nur der Name und nicht die Punktzahl angezeigt werden soll? |
|  |  |

# Die Kamera verwenden (8.1)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie wissen wie Sie den Bildschirmhintergrund mit dem Vorschaubild der Kamera ersetzen können und sind in der Lage dies zu erläutern. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Anstatt immer das gleiche Bild im Hintergrund zu sehen, wollen wir das Vorschaubild der Kamera verwenden. |
| **Aufgabe** | Öffnen Sie die App **Mueckenfang81\_TN** (als ZIP-Datei erhältlich).  Studieren Sie die App und ergänzen Sie den Code mit Kommentaren an den vorgegebenen Stellen (siehe **// ...**).  Beachten Sie insbesondere nachfolgende Bestandteile:  AndroidManifest.xml  Game.xml (leicht abgeändert!)  GameActivity.java  CameraView.java  Ändern Sie das Programm so, dass die Vorderkamera verwendet wird. So erscheinen die Mücken auf Ihrem Gesicht!  Präsentieren Sie Ihre Erkenntnisse der Klasse. |
| **Hinweis Hilfsmittel** | * Lehrmittel 7. Auflage  (berücksichtigt leider noch nicht das neue Berechtigungskonzept von Marshmallow). * Internet * Die ZIP-Datei *Mueckenfang81\_TN.zip* |
| **Ergebnis** | * Eine kurze Beschreibung bzw. Erklärung, wie der Zugriff auf die Kamera erfolgt und wie das Vorschaubild als Hintergrund verwendet wird. * Eine kurze Beschreibung bzw. Erklärung des Berechtigungskonzepts von  Marshmallow und neueren Versionen von Android. |

# Sensoren (1) (9.1)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können anhand eines Beispiels erklären, wie man Sensoren in einer App benutzen kann und wie man auf dessen Sensordaten zugreift. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Smartphones sind, je nach Modell, mit mehr oder weniger vielen Sensoren ausgerüstet. Diese liefern Daten über Magnetfeld, Temperatur, Lichtintensität, Luftdruck, Beschleunigung, usw. Wer native Apps entwickelt, kann auf diese Sensoren zugreifen und vielseitige Apps schreiben, die Umweltdaten berücksichtigen. |
| **Aufgabe** | **Teil 1**  Erstellen Sie eine neue App **Sensoren** nach der bewährten Methode.  Fügen Sie den ersten Teil der Datei *Sensoren.txt* in die *MainActivity* ein und passen Sie die Layoutdatei entsprechend an. Testen Sie Ihre App auf einem reellen Gerät. Sie soll eine Liste der vorhandenen Sensoren ausgeben.  **Teil 2**  Erstellen Sie eine neue App **Schrittzaehler** nach der bewährten Methode.  Fügen Sie beide Klassen aus der Datei *Sensoren.txt* ein und passen Sie die Layoutdatei entsprechend an.  **Die beiden Apps:**    Lesen Sie die Erklärungen im Buch sorgfältig. Sie müssen in der Lage sein, nach Beendigung der Aufgabe, den Code zu erklären. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet * Die Datei *Sensoren.txt* |
| **Ergebnis** | * Funktionstüchtige Apps, die die gewünschten Anforderungen erfüllen. * Notizen mithilfe deren, Sie den Code erklären können. |

# Sensoren (2) (9.1)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie sind in der Lage mit weiteren Sensoren zu arbeiten und ihr Vorhandensein auf einem Smartphone zu überprüfen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Um eine gewisse Routine im Umgang mit Sensoren bzw. Sensorendaten zu erhalten, wollen wir weiter üben. Nicht jedes Smartphone besitzt alle Sensoren. Ihr Vorhandensein muss unbedingt geprüft werden, bevor eine App versucht, auf diese zuzugreifen. |
| **Aufgabe** | Erstellen Sie eine neue App **Sensoren\_2** nach der bewährten Methode.  Testen Sie 1-2 verschiedene Sensoren. Zum Beispiel:    Achten Sie darauf, dass Sie jeweils kontrollieren, ob das Smartphone mit dem gewünschten Sensor ausgerüstet ist. |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. |

# Arbeiten mit Geokoordinaten (9.5)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie sind in der Lage die aktuelle Position und die Höhenlage (Meter über Meer) eines Smartphones zu ermitteln. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Wenn Sie Apps erstellen wollen, die den Benutzerstandort ermitteln sollen, gibt es verschiedene Möglichkeiten dies zu tun. Sie können den Standort durch das GPS, WLAN oder mobile Netzwerk bestimmen. Auf welcher Höhenlage sich das Smartphone befindet, kann u.U. auch herausgefunden werden. Wie man es macht, müssen Sie in dieser Übung herausfinden. |
| **Aufgabe** | **Teil 1**  Erstellen Sie eine neue App **Geokoordinate** nach der bewährten Methode.  Ersetzen Sie den Inhalt von *MainActivity.java* mit dem Code aus *geokoordinate.txt*.  Leider sind Teile des Programms (siehe \*\*\*) verschwunden. Ergänzen Sie den Code entsprechend.  Fügen Sie die notwendigen *import*-Zeilen mit Alt+Enter ein und passen Sie die Layoutdatei entsprechend an. Denken Sie daran, das *Manifest* auch anzupassen.  Testen Sie die App und experimentieren Sie damit (zum Beispiel einmal mit dem GPS\_PROVIDER und einmal mit dem NETWORK\_PROVIDER) und notieren Sie allfällige Erkenntnisse.  **Teil 2**  Ändern Sie die App, um mehr Informationen aus dem *LocationManager* zu holen.  **Die App aus Teil 1 Die App aus Teil 2 (Beispiel)** |
| **Hinweis Hilfsmittel** | * Lehrmittel * Internet * Die Datei *geokoordinate.txt* |
| **Ergebnis** | * Funktionstüchtige App, die die gewünschten Anforderungen erfüllt. * Kurze Beschreibung der gewonnenen Erkenntnisse. |

# Guidelines und Vorschriften (--)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie wissen wo die Richtlinien (Guidelines) für Android Entwickler zu finden sind und können ein paar Beispiele aufzählen. Sie kennen die sieben Grundsätze der Dialoggestaltung nach EN ISO 9241-110. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 30 Min. |
| **Ausgangslage** | Um eine App erfolgreich zu verkaufen, braucht es eine gute Idee (Funktionalität). Dies reicht aber nicht. Potentielle Benutzer bzw. Käufer haben Erwartungen bezüglich Design und Verhalten einer App. Wenn Ihre App sich nicht so verhält wie erwartet, soll heissen: wie die meisten Apps, wird sie keinen Erfolg haben. Das Design, zum Beispiel die Farbe der Buttons, spielt ebenfalls eine grosse Rolle bezüglich Akzeptanz. Wenn Sie eine App für eine Firma entwickeln, müssen Sie auch das Corporate Design (CD) respektieren. |
| **Aufgabe** | **Teil 1**  Finden Sie heraus, wie die sieben Grundsätze der Dialoggestaltung nach EN ISO 9241-110 heissen. Geben Sie für jeden der 7 Punkte ein konkretes Beispiel zur Illustration an.  **Teil 2**  Suchen Sie nach den Guidelines für Android Entwickler und beantworten Sie die Fragen im Anhang unter Angabe des genauen Hyperlinks. |
| **Hinweis Hilfsmittel** | * Internet |
| **Ergebnis** | * Die sieben Grundsätze der Dialoggestaltung nach EN ISO 9241-110 mit konkreten Beispielen. * Korrekte Beantwortung der Fragen (inkl. Quellangaben). * Liste mit weiteren informativen Links. |

**Teil 2**

|  |  |
| --- | --- |
| **Frage 1** | Welche Höhe soll nach Google ein Button haben? |
|  |  |
| **Frage 2** | Welche Schriftart wird von Google empfohlen und in Android UI verwendet? |
|  |  |
| **Frage 3** | Welche Eigenschaften soll ein Button noch haben (Schriftgrösse, Beschriftung)? |
|  |  |
| **Frage 4** | Betrachten Sie nachfolgendes Bild:    Es wurden 2 Fehler gemacht. Welche? |
|  |  |

# App testen (--)

|  |  |
| --- | --- |
| **Ziele erwünschtes Ergebnis** | Sie können Testfälle für eine Mobile App erstellen. |
| **Aufgabentyp** | Erarbeitungsaufgabe |
| **Zeitaufwand** | 40 Min. |
| **Ausgangslage** | Bevor Sie Ihre Apps veröffentlichen oder an Betatester weitergeben, sollten Sie diese gründlich testen. Wie man Applikationen testet, wissen Sie bereits aus anderen Modulen. Bei Mobile Apps sind aber ein paar zusätzliche Tests nötig. Wie reagiert zum Beispiel eine App, die auf eine Internetverbindung angewiesen ist, wenn letztere unterbrochen wird? |
| **Aufgabe** | Nehmen Sie das Mückenspiel als Testobjekt und erarbeiten ca. 5 Testfälle.  Führen Sie die Tests durch und protokollieren Sie das Ergebnis.  Sollten Sie die App so programmiert haben, dass ein Wechseln ins Querformat unterbunden wird, machen Sie dies rückgängig (*screenOrientation* in *AndroidManifest.xml*).  Erarbeiten Sie 5 weitere Testfälle, die allgemein verwendet werden können. Ohne sich auf eine bestimmte App zu beziehen. |
| **Hinweis Hilfsmittel** | * Internet * Unten stehende Vorlage |
| Ergebnis | * Ausgefüllte Tabelle mit 5-10 Testfällen. |

**Vorlage:**

|  |  |  |
| --- | --- | --- |
| **Test Nr.** | **Testfall / Beschreibung** | **Erwartetes Ergebnis** |
| 1 | Beim Anhalten der App (zum Beispiel durch das Drehen des Bildschirms) geht die Punktzahl nicht verloren. | Man kann weiter spielen, die Punktzahl wird weiter aktualisiert. |
| **Ergebnis** | **Bewertung** |
| Alle Werte (u.a. die Punktzahl) werden auf 0 zurückgestellt. | Nicht bestanden |
| **Massnahmen** | |
| Programmatisch verhindern, dass das Bild/Spielfeld bei einem Orientierungswechsel gedreht wird. | |
| **Test Nr.** | **Testfall / Beschreibung** | **Erwartetes Ergebnis** |
| 2 |  |  |
| **Ergebnis** | **Bewertung** |
|  |  |
| **Massnahmen** | |
|  | |