Lorcan Mozis
Matrikelnummer: 2234829
Github: lomoz

Einleitung und Projektidee

Die Projektidee ist, eine Android App in Java mit Hilfe von Android Studio zu entwickeln. Das Programm soll sichere Passwörter erzeugen, die man daraufhin bei verschiedenen Web Konten nutzen kann.
Der Nutzer soll dabei spezifizieren können wie die Passwörter aufgebaut werden sollen. Es soll einen Spielraum bezüglich der Wahl der Länge des Passwortes geben, sowie die Auswahl der Nutzung von Groß- & Kleinbuchstaben, Ziffern und Sonderzeichen.
Wenn ein neues Passwort generiert wurde kann der User dieses per Button Klick in der App in die Zwischenablage des Gerätes kopieren. Somit kann das neue Passwort direkt weiterverwendet werden.
Eine Weiterentwicklung des Programms wäre das Speichern der genutzten Passwörter in einem Passwort Manager. Der Nutzer kann verschiedene Konten anlegen, wo das entsprechende Passwort gespeichert wird und dessen Sicherheitsstufe, sowie Alter angezeigt wird. Dies wäre natürlich aufwendiger zu programmieren und ist in der vorgegebenen Zeit für das Projekt nicht umsetzbar.

Definitionen, Hintergrundinformationen, Begründungen

Um dieses Projekt realisieren zu können, muss man wissen wie die Android Programmierung funktioniert. Diese unterscheidet sich etwas von herkömmlichen Java Code oder dem Programmieren von GUIs mit Javafx. Aus diesem Grund schaute ich mir verschiedene Beginner Tutorials für Android Studio an um einen ersten, groben Überblick zu bekommen. Zudem gab es auf der Website „https://developer.android.com/index.html“ weitere Informationen über die App Entwicklung für Android Geräte.
Des weiteren sollte die App für die jeweils generierten Passwörter eine Sicherheitsstufe anzeigen und ausrechnen wie lange es für einen Hacker dauern würde das Passwort zu knacken. Dazu bedarf es der Brute Force Methode. Um diese Methode nutzen zu können benötigt man verschiedene Werte. Man benötigt die Länge des Passworts, sowie die Länge des sogenannten Charsets. Dieses enthält alle möglichen Elemente, die im Passwort enthalten sein können. Um die Anzahl aller möglichen Kombinationen eines

Passworts mit bestimmter Länge berechnen zu können, rechnet man die Anzahl der im Charset enthaltenen Elemente hoch die Länge des Passworts.
Das Ergebnis teilt man durch *keys per second*, also die Anzahl der Passwort Kombinationen, die pro Sekunde ausprobiert werden können. Dieses Ergebnis wiederum ergibt die Dauer um das Passwort zu hacken in Sekunden.
Diese Zahl ist meist sehr hoch, darum sollte sie entsprechend in Jahre, Monate, Tage, Stunden, Minuten und Sekunden umgeformt werden. Optimal ist es natürlich, wenn dieser Wert so hoch wie möglich ist. Daran könnte dann auch ebenfalls eine Einteilung in Sicherheitsstufen vorgenommen werden. Zwei Web-Beispiele dafür wären „http://www.wiesicheristmeinpasswort.de“ und „http://calc.opensecurityresearch.com“.

Projektplan/Projektaufbau

Ein Passwort Generator soll als Android App implementiert werden. Die App soll in drei Teile gegliedert werden. Der Nutzer der App soll ein Passwort erstellen können, wobei er einerseits die Länge zwischen 4 und 20 Zeichen selbst bestimmen kann und zudem wählen kann ob Großbuchstaben, Kleinbuchstaben, Zahlen und Sonderzeichen im Passwort enthalten sein sollen. Einer dieser vier Werte muss natürlich immer ausgewählt sein.
Die zweite Funktion soll dem Nutzer ein Passwort erzeugen, welches er sich einfach merken kann, aber dennoch sicher ist. Dabei soll das Programm aus einer Liste von häufig vorkommenden Nomen, drei zufällig auswählen und diese mit dem vom Nutzer gewählten Trennzeichen aneinander reihen. Diese Art von Passwort ist ebenfalls sehr sicher.
Die dritte Funktion der App ist ein Sicherheitscheck. Der Nutzer kann ein Passwort eingeben, welches er möglicherweise aktuell bei einem Nutzerkonto verwendet und das Programm berechnet, wie lange es mit der Brute Force Methode dauern würde das Passwort zu hacken. Zudem wird das eingegebene Passwort in eine Sicherheitsstufe eingeteilt.
Die Android App soll soll in Java implementiert werden. Dabei wird das Programm „Android Studio“ verwendet. Für das Erzeugen der Passwörter wird die Random-Bibliothek von *java.util.Random* benötigt. Die einzelnen Passwort Teile (Großbuchstaben,

Kleinbuchstaben, Zahlen und Sonderzeichen) werden als String Array gespeichert, da es so am einfachsten ist die entsprechenden Elemente an eine ArrayList weiter zu geben, die je nach gewählten Einstellungen hinzugefügt werden. Dazu wird *java.util.ArrayList* verwendet. Auch die Nomen für das einfach zu merkende Passwort werden in einer ArrayList gespeichert.

Alle Java Dateien findet man über den Pfad: PasswordGenerator/app/source/main/java/com.example.lorcan.passwordgenerator.
Alle XML Dateien über den Pfad: PasswordGenerator/app/source/main/res/layout

Bei der ersten Funktion, dem klassischen Passwort Generator soll der Nutzer die Länge des Passworts auswählen können. Zudem kann er festlegen ob Großbuchstaben, Kleinbuchstaben, Zahlen und Sonderzeichen verwendet werden sollen.
Durch den Klick auf einen Button wird das Passwort mit den entsprechenden Parametern generiert. Die ausgewählten Optionen müssen im Passwort dann auch mindestens einmal im erstellten Passwort enthalten sein, dies ist bei einigen Web-Passwort-Generatoren nicht der Fall.
Das generierte Passwort wird daraufhin angezeigt und es erscheint ein weiterer Button. Der Nutzer hat durch Klick auf diesen die Möglichkeit das aktuelle Passwort in die Zwischenablage zu kopieren. Dies ist sehr hilfreich, wenn der Nutzer das neue Passwort direkt irgendwo verwenden möchte.
Implementiert wird diese Activity durch „MainActivity.java“. In der onCreate Methode dieser Java Datei wird als Layout die xml Datei „activity_main.xml“ aufgerufen. In einem Frame Layout befinden sich ganz oben drei Buttons, welche das Menü darstellen. Darunter befindet sich ein *TextView* mit dem Text „Choose the password length“. Unterhalb davon ist eine *SeekBar,* diese dient dem Nutzer dazu die Länge des Passwortes festzulegen. Unter der *SeekBar* wird jeweils der gewählte Wert angezeigt. Nun folgen vier *CheckBoxen*, welche der User klicken kann um für sein Passwort Großbuchstaben, Kleinbuchstaben, Ziffern und Sonderzeichen hinzuzufügen. Unterhalb der *CheckBoxen* ist ein *Button*, durch dessen Klick das Passwort entsprechend den vom Nutzer gewählten Optionen generiert wird. Unterhalb dieses *Buttons* ist ein weiterer *TextView*, in dem das

erzeugt Passwort angezeigt wird. Ganz unten im Layout gibt es einen weiteren *Button*. Klickt der Nutzer diesen, so wird das aktuelle Passwort in die Zwischenablage des Gerätes kopiert.
In der OnCrate Methode von „MainActivity.java“ werden zwei Methoden aufgerufen. *setButtonWidth* und *setSeekBar*. Die Methode *setButtonWidth* ist dafür zuständig, die Breite der drei *Buttons* für das Menü an das verwendete Gerät anzupassen. Dafür benötigt man zu erst die Breite des verwendeten Geräts in Pixeln. Diese erhält man mit folgendem Aufruf: *Resources.getSystem().getDisplayMetrics().widthPixels;*
Wenn man diesen Integer Wert durch drei Teilt, erhält man somit die passende Breite für die einzelnen *Buttons*. Den errechneten Wert übergibt man an alle drei Menü *Buttons* und somit werden diese auf jedem verwendeten Gerät proportional gleich angezeigt.
Die Methode *setSeekBar* sorgt dafür, dass immer der aktuelle Wert der *SeekBar* in der entsprechenden *TextView* angezeigt wird. Der Nutzer soll zwischen der Länge 4 und 20 frei entscheiden können, wie lang sein Passwort sein soll. Eine *SeekBar* hat aber immer den Minimalwert 0, daher wurde das Maximum auf 16 gesetzt und auf den aktuellen Wert wird immer vier dazu addiert. Wie man den aktuellen Wert einer SeekBar in einer TextView bei jeder Veränderung anzeigt wurde dem YouTube Video "Android Tutorial for Beginners 19 #SeekBar“ entnommen. Link: https://www.youtube.com/watch?v=l5FrTkGoeX8. Dies betrifft die Code Zeilen 153 bis 177, allerdings wurden aufgrund des Problems, das der Wertebereich zwischen 4 und 20 liegen sollte einige Änderungen im Vergleich zum YouTube Tutorial vorgenommen.
Nach der *onCreate* Methode folgen zwei Methoden, welche ein Options Menü erzeugen, diese heißen *onCreateOptionsMenu* und *onOptionsItemSelected*. Wie man der App ein Options Menü hinzufügt wurde dem YouTube Tutorial "Android Studio Tutorials - 32 : Options Menu in Android (Menu’s)" von „rams android“ entnommen. Im Code entspricht dies den Zeilen 64 bis 103. Link: https://www.youtube.com/watch?v=empirRnFzwM.
Daraufhin folgen die Methoden *buttonGoToGeneratePassword*, *buttonGoToEasyPassword* und *buttonGoToSafetyCheck*. Die erste Methode ist allerdings auskommentiert, da diese hier nicht verwendet wird, weil man sich bereits in der entsprechenden Activity befindet.
Diese Methoden werden den entsprechenden Menü *Buttons* der XML Datei als onClick Methode übergeben. Das bedeutet, dass wenn einer dieser Buttons geklickt wird, die

entsprechende Activity gestartet wird und der Nutzer zum entsprechenden Fenster weitergeleitet wird.
Daraufhin folgen die Methoden *uppercaseChecked*, *lowercaseChecked*, *numbersChecked* und *specialsChecked*. Diese werden den entsprechenden *CheckBoxen* als onClick Methoden übergeben. Wenn eine *CheckBox* geklickt wurde, wird ein entsprechender boolean auf true gesetzt, andernfalls ist dieser false.
Es folgt die onClick Methode *buttonGeneratePasswordClicked* für den *Button*, der das Passwort generiert. Zuerst wird geprüft ob mindestens eine *CheckBox* vom Nutzer geklickt wurde. Ist dies nicht der Fall, so bekommt der Nutzer über einen *Toast* die Meldung, er solle mindestens eine *CheckBox* anklicken. Im Feld für das generierte Passwort steht weiterhin „Password“ und der *Button* zum Kopieren des Passworts bleibt weiterhin unsichtbar.
Falls mindestens eine *CheckBox* geklickt wurde, wird aus der Java Klasse „FillArrayList“ die Methode *fillPasswordArrayListChars* aufgerufen. Ebenfalls wird das fertige Passwort in der *TextView* angezeigt, der Copy *Button* wird sichtbar und ein *Toast* erscheint, in dem steht, dass ein neues Passwort generiert wurde.
Die Methode *fillPasswordArrayListChars* prüft jeweils, ob Groß- und Kleinbuchstaben, Ziffern und Sonderzeichen im Passwort enthalten sein sollen. Falls ja, werden die entsprechenden String Arrays der ArrayList hinzugefügt. Am Ende wird wiederum aus der „MainActivity.java“ Datei die Methode *makePassword* aufgerufen. Übersichtlicher und besser wäre es gewesen, alle Methoden, die nur Programmlogik enthalten in separate Klassen zu unterteilen. Dies war allerdings aus Zeitknappheit nicht möglich, da die meisten Methoden viele Parameter übergeben bekommen hätten und dies zu Fehlern beispielsweise bei der Überprüfung der *CheckBoxen* führte.
Die Methode makePassword erstellt ein vorläufiges Passwort, mit der vom Nutzer durch die *SeekBar* gewählten Länge. Dazu werden zufällig stellen aus den vorher durch die *CheckBoxen* ausgewählten Elemente ausgewählt und aneinander gereiht. Wenn ein vorläufiges Passwort erstellt wurde, wird die Methode *checkPassword* aufgerufen. Die Methode checkPassword ruft zuerst die Methoden *checkUppercase*, *checkLowercase*, *checkNumber* und *checkSpecial* auf. Dort wird überprüft, ob minimal ein Großbuchstabe, ein Kleinbuchstabe, eine Zahl und ein Sonderzeichen im vorläufigen Passwort enthalten ist. In *checkPassword* werden dann alle 15 möglichen Kombinationen der *CheckBoxen*

überprüft, ob das vorläufige Passwort auch den vom Nutzer gewählten Parametern entspricht. Ist dies der Fall wird ein boolean für „isValidPassword“auf true gesetzt. Am Ende der Methode wird geprüft ob „isValidPassword“ gleich true ist. Falls ja, wird das vorläufige Passwort an das finale Passwort übergeben. Andernfalls wird erneut *makePassword* aufgerufen. Somit wird garantiert, dass das finale Passwort immer allen Anforderungen entspricht.
Wenn ein neues Passwort generiert wurde und in der *TextView* angezeigt wird, dann wird ebenfalls der *Button* zum kopieren des Passworts sichtbar. Die onClick Methode dazu heißt *buttonCopyPasswordClicked*. Der Code dazu in den Zeilen 302 bis 304 ist aus dem Beitrag „How to Copy Text to Clip Board in Android?“ von Stackoverflow. Link: http://stackoverflow.com/questions/19253786/how-to-copy-text-to-clip-board-in-android.

Als zweite Funktion der App kann der Nutzer sich ein Passwort erzeugen lassen, welches einfach zu merken ist, aber auch sehr sicher ist. Diese können in der Länge zwar nicht festgelegt werden, aber aber Nutzer kann entscheiden welches Trennzeichen zwischen den zufällig ausgewählten Wörtern verwendet werden soll. Auch hier hat der Nutzer wieder die Möglichkeit das fertige Passwort per Button Klick in die Zwischenablage kopieren.
Implementiert wurde diese Activity durch die Java Datei „ActivityEasyRemember.java“ und die XML Datei „activity_easy_remember.xml“. In der *onCreate* Methode der Java Datei wird das Layout durch die XML Datei festgelegt. Auch hier befinden sich ganz oben im FrameLayout die drei Menü *Buttons*. Darunter ein *TextView* mit dem Text „Choose a separator“. Wiederum darunter befindet sich eine *RadioGroup* mit drei *RadioButtons*. Diese wird verwendet, damit der Nutzer nur ein Element auswählen kann und nicht etwa mehrere, wie bei den *CheckBoxen*. Genau wie bei der vorherigen Activity ist darunter ein *Button* zum generieren des Passworts, ein *TextView* für das Passwort und ein *Button* zu kopieren des Passworts.
In der *onCreate* Methode der Java Datei wird ebenfalls wieder die Methode *setButtonWidth* aufgerufen, damit die drei Menü *Buttons* die richtige Breite je nach Gerät übergeben bekommen. Genau wie bei der MainActivity folgen nun zwei Methoden für das Options Menü und die drei *onClick* Methoden für die Menü *Buttons*, um in andere Activities

zu wechseln. In der *onClick* Methode *buttonGenerateEasyPasswordClicked* wird überprüft, ob einer der RadioButtons angeklickt wurde. Falls einer angeklickt wurde, so wird die Methode *getClickedRadioButton* aufgerufen. Andernfalls erscheint ein *Toast*, in dem der Nutzer darauf hingewiesen wird einen *RadioButton* anzuklicken.
Die Methode *getClickedRadioButton* überprüft, welcher der drei *RadioButtons* angeklickt wurde und übergibt entsprechend das gewählte Trennzeichen an den String „separator“ weiter. Dann wird die Methode *addToArrayList* aufgerufen. Diese füllt eine ArrayList mit den 100 häufigsten englischen Nomen, welche sich zuvor in einem String Array befanden. Dies wird gemacht, da es einfacher ist einen Array von Beginn an zu initialisieren, jedoch das zufällige auswählen von Elementen mit einer ArrayList besser funktioniert. Die 100 häufigsten englischen Nomen im String Array „noun“ von Zeile 180 bis 191 im Code entstammen der Website „linguasorb“. Link: http://www.linguasorb.com/english/most-common-nouns/
Wenn die ArrayList mit allen Nomen gefüllt ist, wird die Methode *makeEasyPassword* aufgerufen. Diese wählt zufällig ein Element aus der ArrayList auf und fügt es zu dem String „easyPassword“ hinzu. Daraufhin wird das gewählte Trennzeichen angefügt. Dieser Prozess wird insgesamt drei mal gemacht, sodass am Ende ein sicheres Passwort entsteht. Bevor das fertige Passwort angezeigt werden kann, muss das letze Element entfernt werden. Es soll nämlich kein Trennzeichen am Ende des Passwortes stehen, sondern nur zwischen den zufällig gewählten Nomen. Wurde das letze Element entfernt, so wird die Methode *displayPassword* aufgerufen. Diese Methode ist dafür zuständig das Passwort im entsprechenden *TextView* anzuzeigen. Zudem wird der Copy *Button* sichtbar. Dieser Button hat die onClick Methode *buttonCopyEasyPasswordClicked*. Hier wird das zuvor generierte Passwort wieder in die Zwischenablage kopiert.

Die dritte Funktion ist ein Sicherheitscheck für Passwörter. Der Nutzer kann ein Passwort eingeben und überprüfen lassen, wie lange es dauern würde das Passwort mit der Brute Force Methode zu hacken. Dieser Zeit entsprechend wird das Passwort in eine von 5 Sicherheitsstufen eingeteilt (nachlässig, niedrig, mittel, hoch und sehr hoch).
Implementiert wird diese durch die Java Datei „ActivitySafetyCheck,java“ und die XML Datei „activity_safety_check.xml“. Auch hier wird in der *onCreate* Methode der Java Datei

die XML Datei für das Layout verwendet. Ganz oben sind wieder die drei Menü *Buttons*. Darunter befindet sich ein *EditText* mit dem *hint* „Enter your Password“. Unter diesem ist der *Button* überprüfen des Passworts. Darunter folgen vier *TextViews*. Im ersten steht „Time to hack password:“, darunter soll die benötigte Dauer erscheinen. Wiederum darunter steht „Security Level of your Password:“ und darunter soll die entsprechende Sicherheitsstufe angezeigt werden.
In der *onCreate* Methode wird ebenfalls wieder die Methode *setButtonWidth* aufgerufen, damit die Menü Buttons die richtige Breite haben. Darauf folgen wieder die beiden Methoden für das Options Menü und die drei Methoden um die Activities zu wechseln. In der *onClick* Methode *buttonCheckPasswordSecurity* wird zuerst der in das dafür vorgesehene Feld Text in einen String gespeichert. Wenn dieser die Länge 0 hat, so erscheint ein *Toast*, in dem der Nutzer darauf aufmerksam gemacht wird, dass er nichts eingegeben hat. Andernfalls wird die Methode *calculateTime* aufgerufen. In dieser Methode wird anhand der Länge des Charsets (26 Großbuchstaben + 26 Kleinbuchstaben + 10 Ziffern + 18 Sonderzeichen = 80 Zeichen) und der Länge des Passworts die Dauer in Sekunden berechnet, um das Passwort zu hacken. Diese Zahl wird daraufhin in Jahre, Monate, Tage, Stunden, Minuten und Sekunden umgerechnet. Am Ende wird die Methode *setTime* aufgerufen. Dabei wird die errechnete und formatierte Zeit im entsprechenden *TextView* angezeigt. Daraufhin wird die Methode *setSecurityLevel* aufgerufen. Dort wird anhand der errechneten Zeit die Sicherheitsstufe errechnet und in der passenden *TextView* angezeigt.

Wenn der Nutzer in einer der Hauptactivities auf das Options Menü klickt so hat er die Wahl zwischen den Optionen „Settings“, „Help“, „Feedback“ und „About“. Diese werden jeweils durch eigene Java und XML Dateien dargestellt. Klickt der Nutzer auf eine dieser Optionen erscheint allerdings nur ein leeres Fenster, in dem nur der Name der gewählten Option steht. Dieser Aspekt wurde im Projektplan nicht weiter spezifiziert und aus zeitlichen Gründen wurde dieser Teil der App auch nicht weiter entwickelt.

Anleitung zur Ausführung/Benutzung

1. APK „PasswordGenerator.apk“ auf das Zielgerät downloaden.
2. APK installieren.
3. Eventuell „unbekannte Quellen“ in den Einstellungen aktivieren.
4. App ausführen.

Reflexion

Sowohl der normale Passwort Generator, als auch das generieren der einfach zu merkenden Passwörter funktioniert einwandfrei. Allerdings funktioniert die dritte Activity, der Sicherheitscheck, leider nicht wirklich. Dies ist einerseits dem Ablauf der Zeit für das Projekt geschuldet, da die momentan vorhandenen Probleme sicherlich auf Dauer lösbar wären. Andererseits ist die Anzahl der möglichen Kombinationen ein weiteres Problem. Das Charset hat immer die Größe 80. Somit wird immer 80 hoch die Länge des Passwortes gerechnet. Angenommen man möchte eins der zuvor generierten Passwörter prüfen, beziehungsweise nachdem ein Passwort generiert wurde sollen sowohl Dauer als auch Sicherheitsstufe angezeigt werden, liegt der Wert für die Anzahl der möglichen Kombinationen minimal bei 80^4 = 40.960.000 und maximal 80^20 = 1,15 x 10^38. Diese hohen Werte sind höchstwahrscheinlich nicht korrekt, selbst in einem long, gespeichert werden können. Somit funktioniert es auch nicht diesen Wert durch 250.000 keys per second zu teilen. Was wiederum zur Folge hat, dass die Formatierung des Ergebnisses nicht korrekt funktioniert und die Einteilung in Sicherheitsstufen ebenfalls nicht.
Des Weiteren wird bei einem Menü Button Klick immer wieder eine neue Activity gestartet, statt die jeweilige einfach nur fortzusetzen. Dies ist auf Dauer auch etwas lästig, allerdings kein zu großes Problem, da lediglich alle Fenster wieder im Startzustand dargestellt werden. Mit mehr Zeit und besseren Kenntnissen der Android App Programmierung wäre es sicherlich möglich gewesen, alle Activities von Beginn an zu starten und die jeweiligen nicht geöffneten zu pausieren und auf Klick des Nutzers weiter laufen zu lassen.

Außerdem wäre es schön gewesen, wenn man zwischen den Activities nach links und rechts hätte swipen können, statt nur auf einen Buttonklick. Aber auch hierzu fehlten Zeit und Erfahrung.
Insgesamt bin ich mit der App sehr zufrieden und werde die zuvor erwähnten Probleme höchst wahrscheinlich noch beheben, da mir die App Programmierung Spaß macht und ich gerne weitere Erfahrungen auf diesem Gebiet sammeln würde. Dafür, dass es meine erste App ist finde ich war der Umfang und die Schwierigkeit des Projekt zu Beginn des Seminars gut gewählt.

Anmerkung

In Android Studio selber habe ich zum testen der App als Emulator ein Nexus S mit API 23 verwendet oder per USB Debugging mein eigenes Smartphone, ein Sony Xperia Z1 Compact (Android Version 5.1.1., API 21), benutzt. Auf diesen beiden Geräten waren die Proportionen auch so wie gewünscht. Ich hoffe das dies auf den Testgeräten ebenfalls der Fall sein wird.

Quellenverzeichnis:

- Android Developers. (2016). Abgerufen von: „https://developer.android.com/index.html“
- App Icon. Creative Common Rechte: CC BY-SA 3.0. Abgerufen von: https://upload.wikimedia.org/wikipedia/commons/9/98/SafeWalletLogo.png
- Kaiser, M. (2016). *Passwort Sicherheit prüfen*. Abgerufen von: „http://www.wiesicheristmeinpasswort.de“
- linguasorb. (2016). *100 Most Common English Nouns*. Abgerufen von: http://www.linguasorb.com/english/most-common-nouns/
- Open Security Research sponsored by Foundstone. *Brute Force Calculator*. Abgerufen von: „http://calc.opensecurityresearch.com“
- ProgrammingKnowledge. (14.03.2015). *Android Tutorial for Beginners 19 #SeekBar* [Videodatei]. Abgerufen von: https://www.youtube.com/watch?v=l5FrTkGoeX8
- rams android. (19.02.2015). *Android Studio Tutorials - 32 : Options Menu in Android (Menu’s)* [Videodatei]. Abgerufen von: https://www.youtube.com/watch?v=empirRnFzwM
- Stackoverflow. (2 Years ago). *How to Copy Text to Clip Board in Android?* Abgerufen von: http://stackoverflow.com/questions/19253786/how-to-copy-text-to-clip-board-in-android