



Primfaktorzerlegung

Die App – Anleitung

Beim Start der App wird eine Zufallszahl zwischen 20 und 500 erzeugt und angezeigt. Im Wechsel geben zwei Spieler Primfaktoren der Zahl ein.

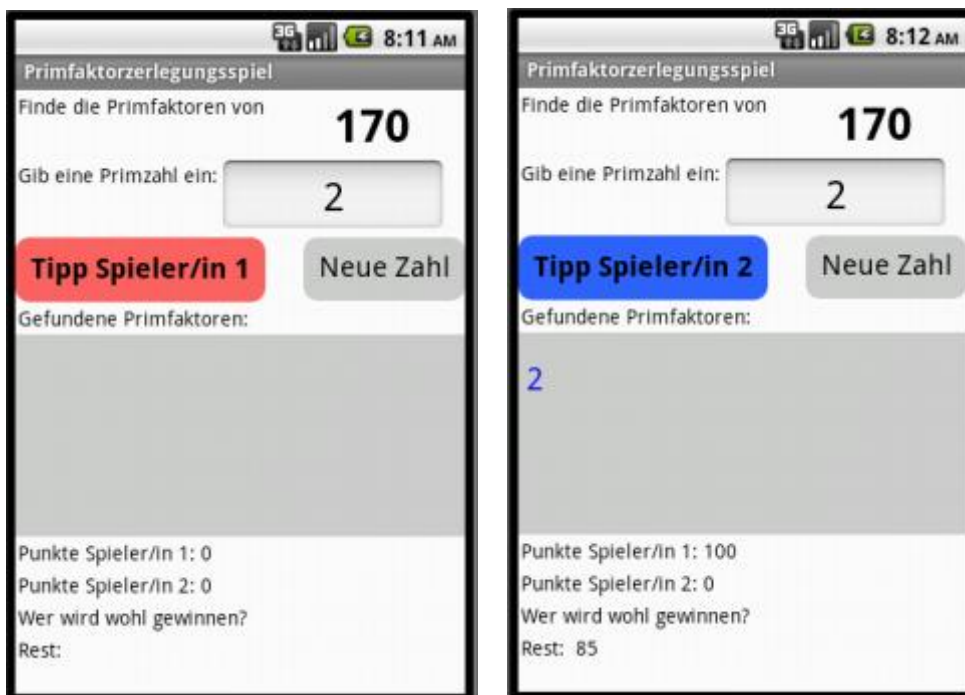
Ist die eingegebene Zahl ein Primfaktor, der noch fehlt, bekommt der aktuelle Spieler 100 Punkte. Ist es zwar eine Primzahl, aber kein Teiler bekommt die Spielerin 50 Punkte abgezogen. Bei einer Zahl, die nicht mal eine Primzahl ist, werden 100 Punkte abgezogen.

Ganz unten wird der Rest nach Division durch den Primfaktor ausgegeben. Ist der Rest 1, wird der Gewinner bzw. die Gewinnerin ermittelt und angezeigt.

Beim Klick auf „Neue Zahl“ startet das Spiel erneut.

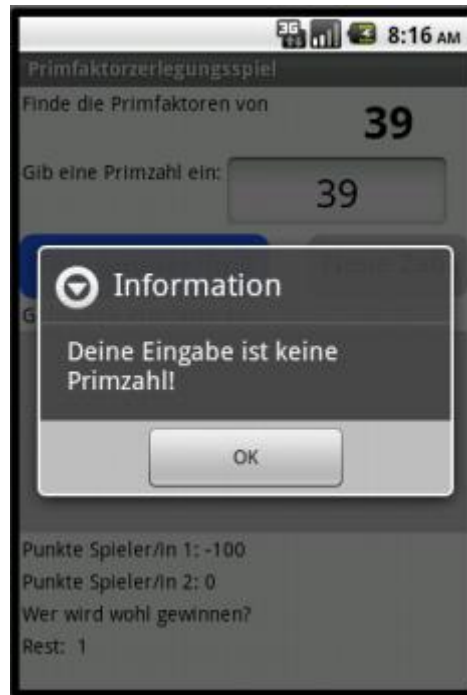
Der Screen

In der ersten Abbildung sieht man den Startbildschirm der App. Spieler/in 1 gibt die 2 als ersten Primfaktor ein, erhält dafür 100 Punkte, die 2 wird in der Liste gespeichert, angezeigt und die Beschriftung des Buttons wechselt. Ganz unten wird der Rest angegeben.

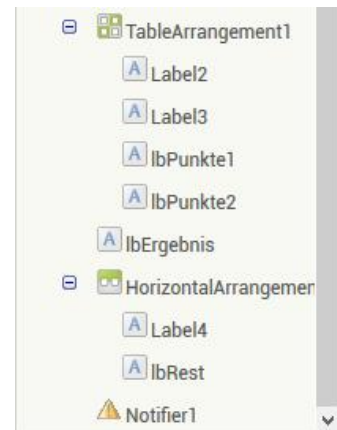
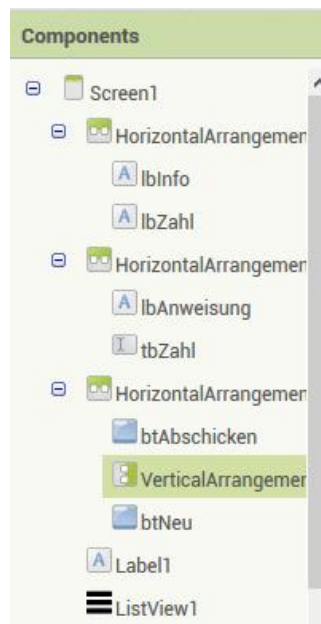
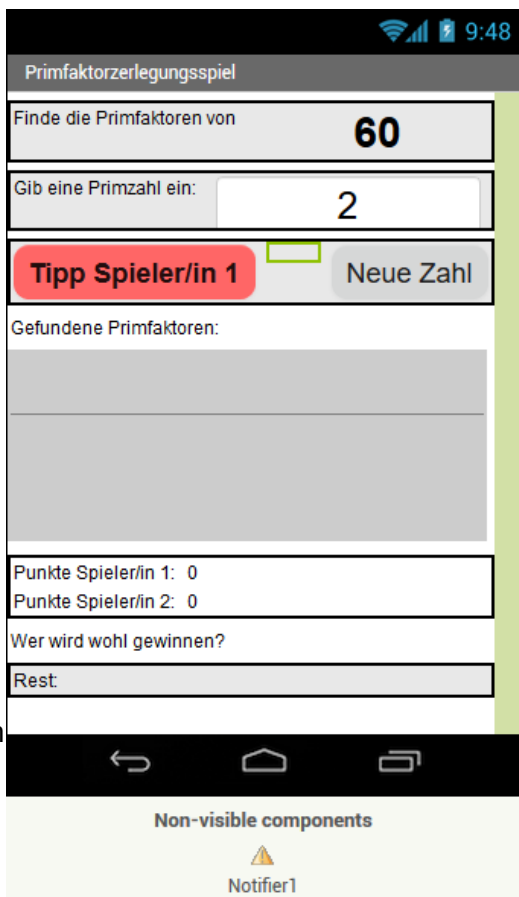


Ist der letzte Primfaktor gefunden, wird ausgewertet. In unserem Fall hat Spieler/in 1 gewonnen.

In der zweiten Abbildung unten sieht man die Information, die gegeben wird, wenn die Eingabe keine Primzahl ist.



Der Screen im MIT-App-Inventor



im

Komponenten, die lediglich Information geben aber Code nicht verwendet werden, wurden nicht umbenannt.



Der Code

Initialisierung der Variablen:



In der Variable *zahl* wird die zu Beginn erzeugte Zufallszahl abgespeichert, in *eingabe* die Zahl, die im Textfeld eingegeben wurde.

In der Liste *faktorliste* werden später alle gefundenen Primfaktoren als Elemente gespeichert.

Die boolesche Variable *spieler1* gibt an, ob Spieler/in 1 an der Reihe ist oder nicht, *ist_prim* gibt an, ob die aktuelle Eingabe eine Primzahl ist oder nicht.

In *punkte_s1* und *punkte_s2* werden die Punktzahlen der beiden Spieler/innen abgespeichert, in der Variable *rest*, der Rest nach der Division.

Initialisierung des Screens:



Sowohl in *zahl* als auch in *rest* wird die erzeugte Zufallszahl abgespeichert. Die Zahl wird angezeigt.

Beim Klick auf den Button mit der Aufschrift „Tipp Spieler/in 1“, wird zunächst die eingegebene Zahl in der Variable *eingabe* gespeichert. In einem Unterprogramm wird überprüft, ob die Zahl eine Primzahl ist. Falls ja, wird in einem weiteren Unterprogramm überprüft, ob sie auch ein Teiler der Zahl bzw. des aktuellen Restes ist.

Falls nein, wird eine Nachricht ausgegeben und dem aktuellen Spieler werden 100 Punkte abgezogen.

Am Ende wird das Unterprogramm *neue_eingabe* aufgerufen.





```

when btAbschicken .Click
do
  set global eingabe to tbZahl . Text
  call ueberpruefe_primzahl
  x = get global eingabe
  if get global ist_prim = true
  then
    call ist_faktor
    x = get global eingabe
    rest_akt = get global rest
    set lbRest . Text to get global rest
    if get global rest = 1
    then call gewonnen
  else
    call Notifier1 .ShowMessageDialog
    message "Deine Eingabe ist keine Primzahl!"
    title "Information"
    buttonText "OK"
    if get global spieler1 = true
    then
      set global punkte_s1 to get global punkte_s1 - 100
      set lbPunkte1 . Text to get global punkte_s1
    else
      set global punkte_s2 to get global punkte_s2 - 100
      set lbPunkte2 . Text to get global punkte_s2
  call neue_eingabe
  
```



Im Unterprogramm *ueberpruefe_primzahl* wird zunächst als Eingabe die 1 ausgeschlossen und dann werden Teiler der Zahl gesucht. Hierbei genügt es, in einer for-Schleife bis zur Wurzel (bzw. deren gerundeten Wert) der Zahl zu laufen.

Da die Variable *ist_prim* ursprünglich auf true gesetzt ist, genügt es hier beim Finden eines Teilers die Variable auf false zu setzen.

```

to ueberpruefe_primzahl x
do
  if (get x = 1)
  then
    set global ist_prim to false
  else
    for each number from 2
    to round (square root of get x)
    by 1
    do
      if (modulo of (get x ÷ get number) = 0)
      then
        set global ist_prim to false
  
```



Im Unterprogramm *ist_faktor* wird überprüft, ob die eingegebene Zahl (*x*) ein Teiler des aktuellen Restes (*rest_akt*) ist.

Falls ja, wird die Zahl als neues Element an die Primfaktorliste angehängt, diese angezeigt, der neue Rest berechnet und dem aktuellen Spieler 100 Punkte gut geschrieben.

Falls nein, wird eine Nachricht angezeigt und der aktuellen Spielerin werden Punkte abgezogen.

```

to ist_faktor x rest_akt
do
  if modulo of get rest_akt ÷ get x = 0
  then
    add items to list list get global faktorliste
    item get x
    set global rest to get rest_akt / get x
    set ListView1 . Elements to get global faktorliste
    if get global spieler1 = true
    then
      set global punkte_s1 to get global punkte_s1 + 100
      set lbPunkte1 . Text to get global punkte_s1
    else
      set global punkte_s2 to get global punkte_s2 + 100
      set lbPunkte2 . Text to get global punkte_s2
    else
      call Notifier1 . ShowMessageDialog
      message "Kein Teiler oder schon gefunden!"
      title "Information"
      buttonText "OK"
      if get global spieler1 = true
      then
        set global punkte_s1 to get global punkte_s1 - 50
        set lbPunkte1 . Text to get global punkte_s1
      else
        set global punkte_s2 to get global punkte_s2 - 50
        set lbPunkte2 . Text to get global punkte_s2
  end
end
    
```



Im Unterprogramm *neue_eingabe* wird die boolsche Variable *ist_prim* wieder auf true gesetzt, der aktuelle Spieler wird geändert und die Aufschrift des Buttons geändert.

```

to neue_eingabe
do
  set global ist_prim to true
  set global spieler1 to not get global spieler1
  if get global spieler1 = true
  then
    set btAbschicken . Text to "Tipp Spieler/in 1"
    set btAbschicken . BackgroundColor to red
  else
    set btAbschicken . Text to "Tipp Spieler/in 2"
    set btAbschicken . BackgroundColor to blue
  end
end
    
```

Es fehlen noch das Unterprogramm *gewonnen* und das OnClick-Ereignis des Buttons „Neue Zahl“.

```

to gewonnen
do
  if get global punkte_s1 > get global punkte_s2
  then
    set lbErgebnis . Text to "Spieler 1 hat gewonnen!"
    set lbPunkte1 . TextColor to orange
  else if get global punkte_s1 < get global punkte_s2
  then
    set lbErgebnis . Text to "Spieler 2 hat gewonnen!"
    set lbPunkte2 . TextColor to orange
  else
    set lbErgebnis . Text to "Gleichstand!"
    set lbPunkte1 . TextColor to orange
    set lbPunkte2 . TextColor to orange
  end
end
    
```

Beim Neustart werden alle nötigen Variablen und Eigenschaften der Komponenten zurückgesetzt auf den Ausgangszustand.



```

when btNeu .Click
do
  set global zahl to random integer from 20 to 500
  set global rest to get global zahl
  set lbZahl .Text to get global zahl
  set lbErgebnis .Text to "Wer wird wohl gewinnen?"
  set btAbschicken .Text to "Tipp Spieler/in 1"
  set btAbschicken .BackgroundColor to red
  set lbPunkte1 .TextColor to black
  set lbPunkte2 .TextColor to black
  set lbPunkte1 .Text to 0
  set lbPunkte2 .Text to 0
  set global faktorliste to create empty list
  set ListView1 .Elements to get global faktorliste
  set global ist_prim to true
  set global spieler1 to true
  set global punkte_s1 to 0
  set global punkte_s2 to 0
  
```

Bemerkungen

Es gibt keine Fehlerabfrage, sollte eine Dezimalzahl eingegeben werden.

Reizvoll wäre eine Ergänzung, die mithilfe einer Bluetoothverbindung das Spielen über zwei Endgeräte ermöglicht.

Außerdem könnte man für einen Primfaktor, der mehrfach in der Primfaktorzerlegung auftaucht, mehr Punkte geben.

Den Code zum Bearbeiten findet man unter *primfaktorzerlegung.aia*, die App zum Installieren unter *primfaktorzerlegung.apk*.