

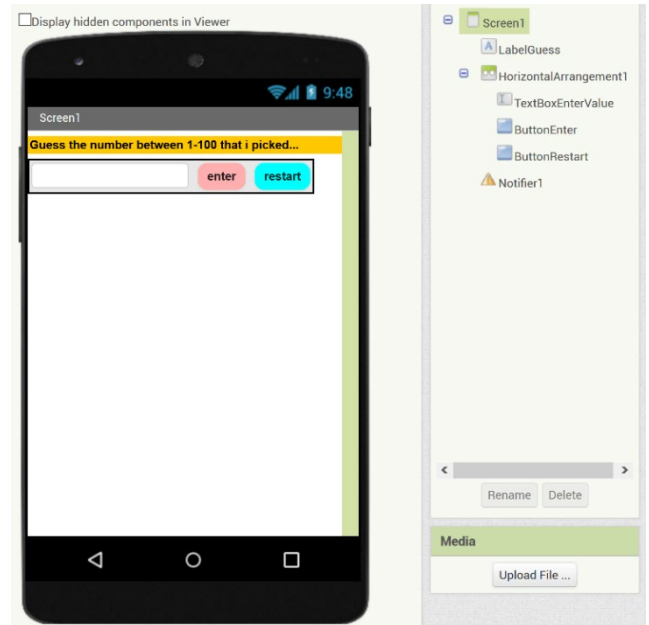
Variante-1 : Basic

Eigenschaften:

- Unbegrenzte Anzahl von Rateversuche.
- Benutzer-Feedback (zu hoch, zu niedrig, richtig) mit Benachrichtigung (Notifier).
- Neustart-Button, Erneuerung der zu erratenden Zufallszahl.

Frage:

Prüfe auf eine mögliche Verbesserung durch ein Prozedur!



```

initialize global NumberPicked to 0

initialize global NumberGuessed to 0

when Screen1.Initialize
do set global NumberPicked to random integer from 1 to 100

when ButtonRestart.Click
do set global NumberPicked to random integer from 1 to 100

when ButtonEnter.Click
do set global NumberGuessed to TextBoxEnterValue.Text
call Notifier1.ShowAlert
notice join "The number you guessed is: "
get global NumberGuessed
if is number? get global NumberGuessed
then if get global NumberGuessed < get global NumberPicked
then call Notifier1.ShowAlert
notice "Too low..."
else if get global NumberGuessed > get global NumberPicked
then call Notifier1.ShowAlert
notice "Too high..."
else call Notifier1.ShowAlert
notice "Congratulations! You got it right!"
else call Notifier1.ShowAlert
notice "Please enter an integer number.."
set TextBoxEnterValue.Text to ""
  
```

Variante-2: Mittelkomplex

Half-Brute-Force mit Irgendeine-Komponente (Any-Component)

Eigenschaften:

- Begrenzte und feste Anzahl von Rateversuche. Visualisierung mit Tabelle.
- Initialisierung-Prozedur.
- Benutzerfeedback mit Schieberegler.
- Bezeichnung und Schieberegler (labels and sliders) Blöcke sind irgendeine-Komponente (Any-Component) Blöcke, die innerhalb der Schleife angesprochen werden.
- Interessant, um Listen und Indizierung einzuführen.
- Neustart-Button, Zufallszahl erneuern, die zu erraten ist.

The screenshot displays the 'Viewer' window on the left, showing a mobile application interface. The interface includes a title bar 'Guess the Number_mid', a text input field with a placeholder 'Guess prompt text comes here...', and two buttons: 'enter' and 'restart'. Below the input is a table with the following data:

Guess	You chose...	Value...	And it is...
Nr.1		0	tbd
Nr.2		0	tbd
Nr.3		0	tbd
Nr.4		0	tbd

Below the table is another text input field. The 'Components' window on the right shows the following hierarchy:

- Screen1
 - LabelGuess
 - HorizontalArrangement1
 - TextBoxEnterValue
 - ButtonEnter
 - ButtonRestart
 - TableArrangement1 (highlighted with a blue arrow)
 - Table (highlighted with a blue arrow)
 - LabelGuessNr
 - LabelUChose
 - LabelValue
 - LabelAndIts
 - LabelNr1
 - LabelNr2
 - LabelNr3
 - LabelNr4
 - Slider1
 - Slider2
 - Slider3
 - Slider4
 - LabelVal1
 - LabelVal2
 - LabelVal3
 - LabelDiag1
 - TextRightNumber
 - Notifier1

Variante 2- Code

```

initialize global NumberPicked to 0
initialize global SliderList to create empty list
initialize global NumberGuessed to 0
initialize global LabelValList to create empty list
initialize global GuessNr to 0
initialize global LabeDiagList to create empty list
initialize global DiagTxt to ""

when Screen1.Initialize
do
  call initComponentsLists
  call initGame
  set LabelGuess.Text to join
    "
    \n Guess the number between 1-100 that i picked.... "
    "
    You have 4 chances. \n \n "
    "
    Type your guess and press enter \n "

when ButtonRestart.Click
do
  call initGame

to initComponentsLists
do
  add items to list list get global SliderList
  item Slider1
  item Slider2
  item Slider3
  item Slider4
  add items to list list get global LabelValList
  item LabelVal1
  item LabelVal2
  item LabelVal3
  item LabelVal4
  add items to list list get global LabeDiagList
  item LabelDiag1
  item LabelDiag2
  item LabelDiag3
  item LabelDiag4

to initGame
do
  set global GuessNr to 0
  set global NumberPicked to random integer from 1 to 100
  set TextBoxEnterValue.Text to ""
  set TextRightNumber.Text to ""
  for each number from 1 to 4 by 1
  do
    set Slider.ThumbPosition
      of component select list item list get global SliderList
      index get number
    to 0
    set Label.Text
      of component call SelectLabel
      type "value"
      index get number
    to ""
    set Label.Text
      of component call SelectLabel
      type "diag"
      index get number
    to ""

when ButtonEnter.Click
do
  set global GuessNr to get global GuessNr + 1
  set global NumberGuessed to TextBoxEnterValue.Text
  if get global GuessNr <= 4
  then
    call TestInputGuess
    call DisplayGuess
  else
    call Notifier1.ShowAlert
    notice "You lost this round. But can try again! Press r..."
    set TextRightNumber.Text to join
      "
      The number i picked was: "
      get global NumberPicked

to TestInputGuess
do
  if is number? get global NumberGuessed
  then
    if get global NumberGuessed < get global NumberPicked
    then set global DiagTxt to "Too low..."
    else if get global NumberGuessed > get global NumberPicked
    then set global DiagTxt to "Too high..."
    else set global DiagTxt to "Congratulations! You got it right!"
  else
    call Notifier1.ShowAlert
    notice "Please enter an integer number.."

to DisplayGuess
do
  set Slider.ThumbPosition
    of component select list item list get global SliderList
    index get global GuessNr
  to get global NumberGuessed
  set Label.Text
    of component call SelectLabel
    type "value"
    index get global GuessNr
  to get global NumberGuessed
  set Label.Text
    of component call SelectLabel
    type "diag"
    index get global DiagTxt
  to get global DiagList
  set TextBoxEnterValue.Text to ""

to SelectLabel type index
result
  if get type = "value"
  then select list item list get global LabelValList
  index get index
  else select list item list get global LabeDiagList
  index get index
  
```

Variante 3: Mittel-hoch-komplex

Einstellbare Listen für flexible Anzahl von Rateversuchen

Eigenschaften:

- Anzahl der Rateversuche in globaler Konstante definiert, und kann einfach geändert werden, ohne dass weitere Änderungen im Code notwendig sind.
- Umfangreiche Verwendung von Prozeduren. Nettes Beispiel für "Teile und herrsche". Aber da der Code in AI2 flach ist, wird er für einen einzelnen Bildschirm sehr groß.

The image shows a mobile application interface for a 'Guess the Number' game. The interface is displayed on a smartphone screen within a 'Viewer' window. The app has a title bar 'Guess the Number_top'. Below the title bar, there is a yellow banner with the text 'Guess prompt text comes here...'. Underneath the banner is a text input field, a red 'enter' button, and a cyan 'restart' button. Below the input field is another yellow banner with the text 'Trial Your guess... And it is...'. At the bottom of the screen, there is a text area with the text 'Your array is displayed here...'. The status bar at the top shows the time as 9:48 and various icons.

To the right of the viewer is a 'Components' panel. It shows a tree view of the application's components:

- Screen1
 - LabelGuess
 - HorizontalArrangement1
 - TextBoxEnterValue
 - ButtonEnter
 - ButtonRestart
 - TableArrangement1
 - LabelTrialNr
 - LabelUrGuess
 - LabelAndItIs
 - txtArrayValues
 - Notifier1

Below the component list are 'Rename' and 'Delete' buttons. At the bottom of the components panel is a 'Media' section with a file named 'img_3389...ess_A.png' and an 'Upload File ...' button.

Variante 3: Code Teil 1/3

```

initialize global ArraySize to 7

initialize global ArrayValue to create empty list

initialize global ArrayDiag to create empty list

when Screen1.Initialize
do
  call initGame
  set LabelGuess.Text to join
  " \n Guess the number between 1-100 that i picked.... "
  " You have "
  get global ArraySize
  " chances. \n \n "
  " Type your guess and press enter \n "

when ButtonRestart.Click
do
  call initGame

to initGame
do
  set global GuessNr to 0
  set global NumberPicked to random integer from 1 to 100
  set TextBoxEnterValue.Text to ""
  set global ArrayValue to create empty list
  set global ArrayDiag to create empty list
  for each index from 1
    to get global ArraySize
    by 1
  do
    add items to list list get global ArrayValue
    item ""
    add items to list list get global ArrayDiag
    item ""
  call txtArrayValues.HideKeyboard
  set txtArrayValues.Text to ""

to SetArray array index value
do
  replace list item list get array
  index get index
  replacement get value

to GetArray array index
result
  select list item list get array
  index get index

```


Variante 3: Code Teil 2/3

```

initialize global NumberPicked to 0
initialize global NumberGuessed to 0
initialize global GuessNr to 0
initialize global DiagTxt to ""

when ButtonEnter.Click
do
  set global GuessNr to get global GuessNr + 1
  set global NumberGuessed to TextBoxEnterValue.Text
  if get global GuessNr ≤ get global ArraySize
  then
    call TestInputGuess
    call StoreTrial
    call DisplayGuess
    set TextBoxEnterValue.Text to ""
  else
    call Notifier1.ShowDialog
      message "You lost this round.. But can try again! Press r..."
      title "Lost. Try again!"
      buttonText "OK"

to TestInputGuess
do
  if is number? get global NumberGuessed
  then
    if get global NumberGuessed < get global NumberPicked
    then
      set global DiagTxt to "Too low..."
    else if get global NumberGuessed > get global NumberPicked
    then
      set global DiagTxt to "Too high..."
    else
      set global DiagTxt to "Congratulations! You got it right!"
  else
    set Notifier1.BackgroundColor to blue
    call Notifier1.ShowAlert
      notice "Please enter an integer number.."

```

Variante 3: Code Teil 3/3

```

to StoreTrial
do
  call SetArray
  array get global ArrayValue
  index get global GuessNr
  value get global NumberGuessed
  call SetArray
  array get global ArrayDiag
  index get global GuessNr
  value get global DiagTxt
end

to DisplayGuess
do
  call txtArrayValues .HideKeyboard
  set txtArrayValues .Text to " "
  for each index from 1
  to get global ArraySize
  by 1
  do
    set txtArrayValues .Text to join
    "Nr."
    get index
    " "
    call GetArray
    array get global ArrayValue
    index get index
    " "
    call GetArray
    array get global ArrayDiag
    index get index
    "\n"
  end
end
  
```