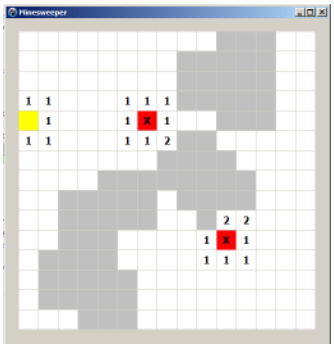
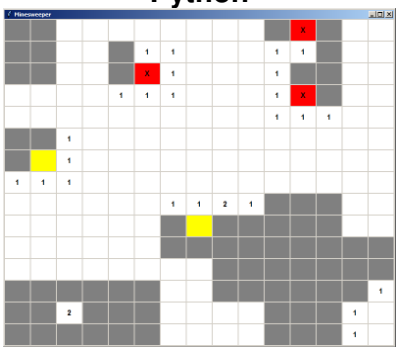
Zum Spiel

„Minesweeper“ wurde erstmals dem Betriebssystem Win 3.1 beigelegt.

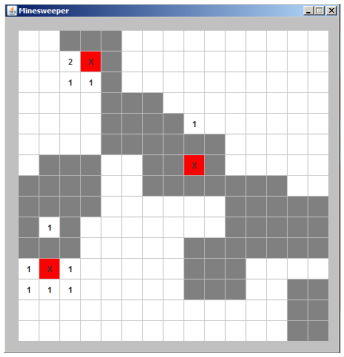
Lazarus



Python



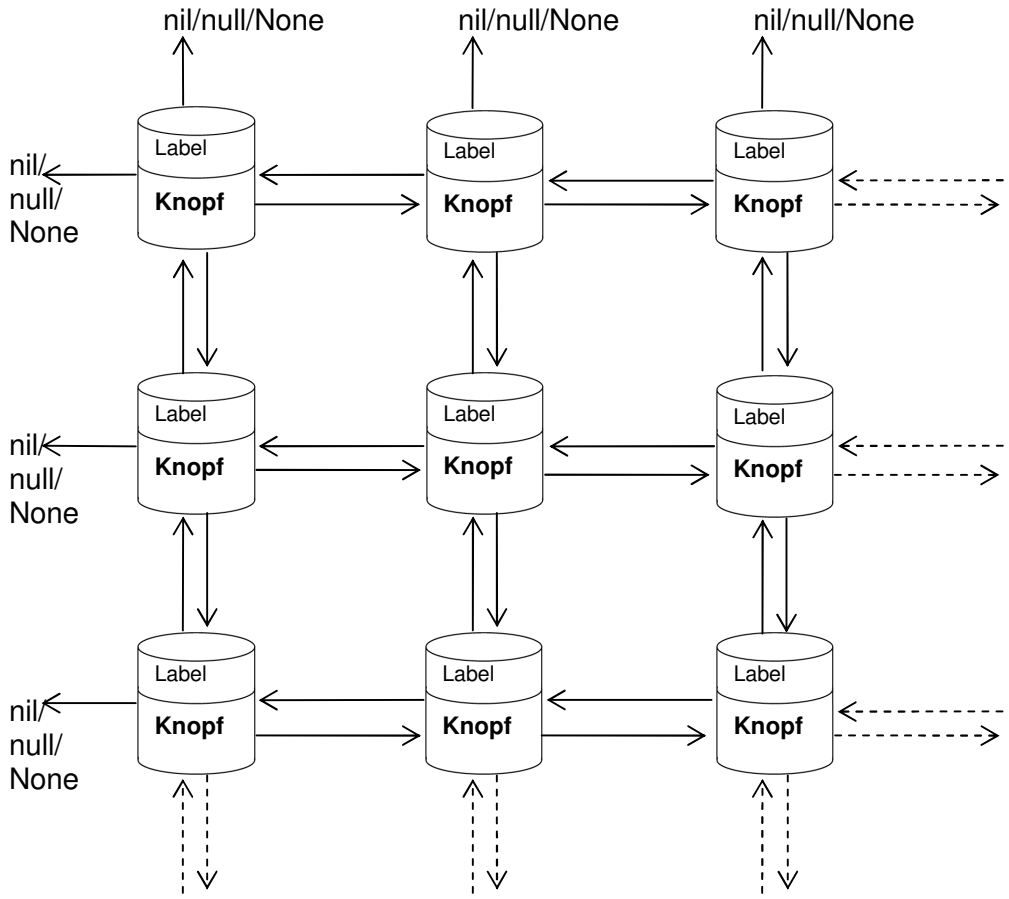
Java



Bei dem Spiel werden eine Anzahl "bomben" zufällig unter den Spielfelder verteilt. Beim Linksklick auf eine "Bombe" endet das Spiel, beim Linksklick auf eine freies Feld wird die Anzahl der Bomben in unmittelbarer Nachbarschaft angezeigt, oder es öffnen sich alle Nachbarfelder ohne Bombennachbarschaft.

Ziel des Spiels ist es, alle Bombenfelder mit der rechten Maustaste zu markieren, ohne eines zu öffnen.

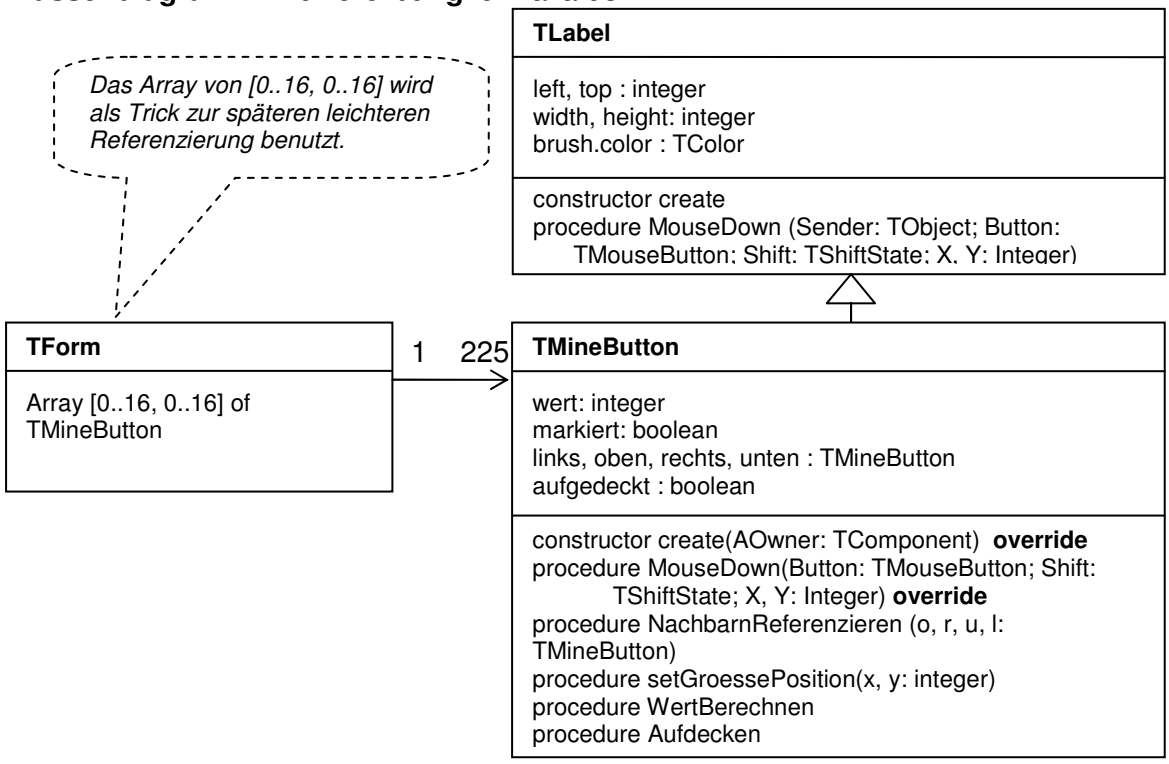
Referenzskizze der MinenKnöpfe in der linken oberen Ecke



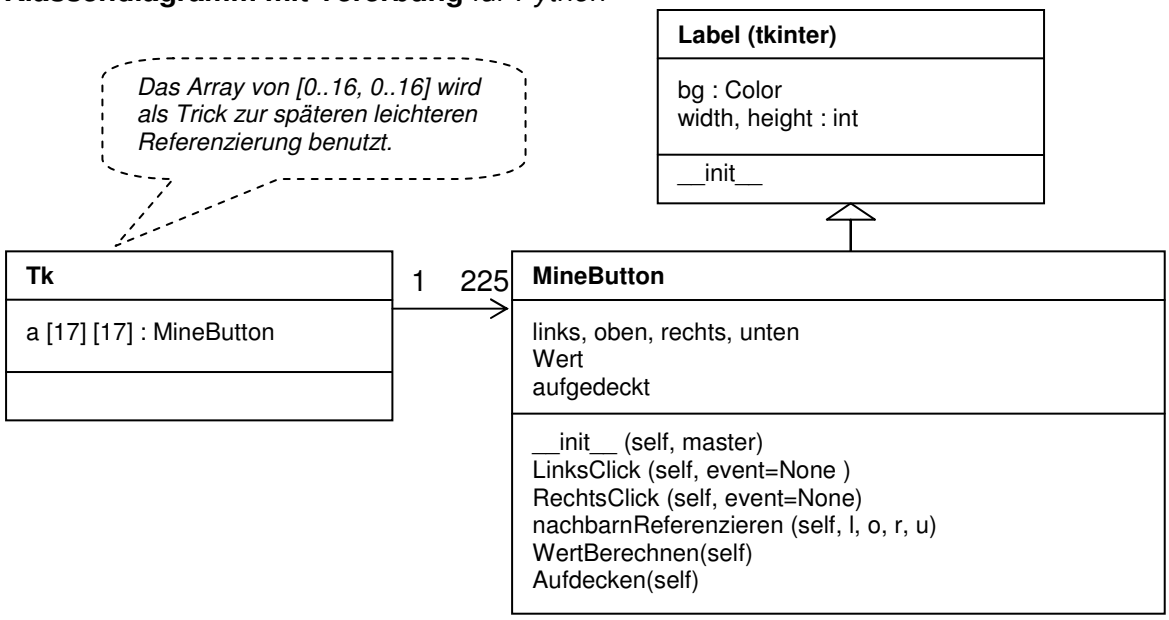
Grundidee zur Umsetzung
Jedes Feld wird als Objekt implementiert, das sowohl Methoden bereitstellt, in einem Formular oder Fenster angezeigt zu werden (Vererbung) als auch Mausereignisse (Klicks) zum Ändern der eigenen Farbe (Markierung) oder zum eigenen Aufdecken. Beim Aufdecken wird eine Nachricht an alle Nachbarn geschickt, sich auf aufzudecken, falls keine Bombe in der Nachbarschaft ist (weißes Feld). Das Aufdecken der Nachbarn geht also über deren Methodenaufruf zum eigenen Aufdecken (dem objektorientierten Prinzip des Sendens von Nachrichten).

Zum Senden der Nachricht des Aufdeckens erhält jedes Objekt Referenzen auf seine Nachbarn. Die Randlampen referenzieren dabei nur 3 bzw. 2 Objekte.

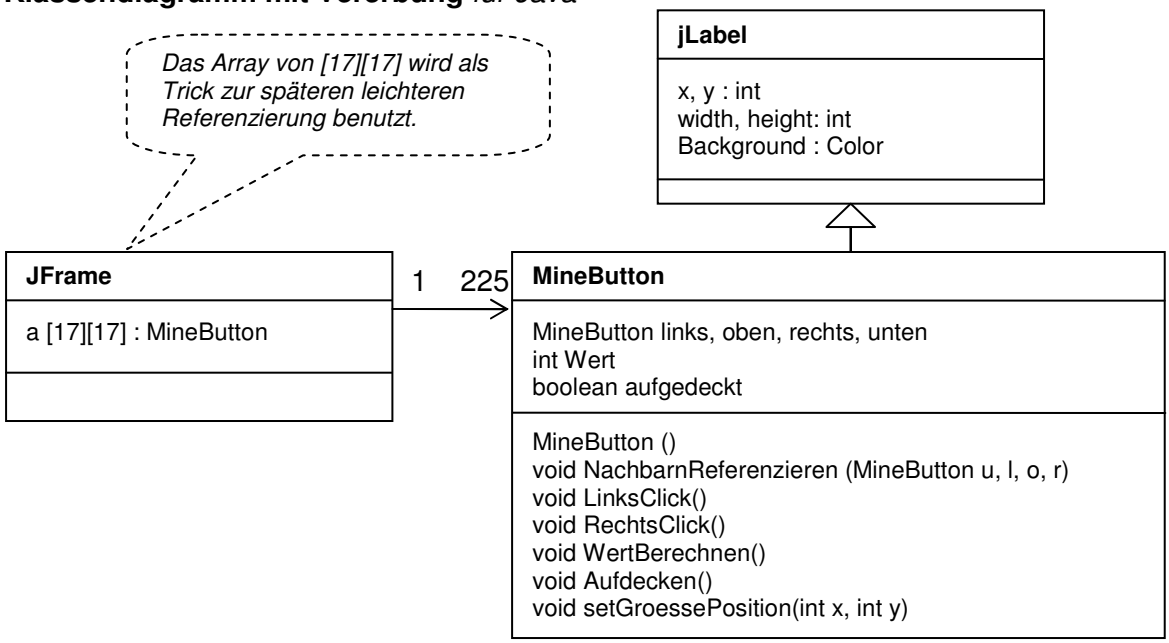
Klassendiagramm mit Vererbung *für Lazarus*



Klassendiagramm mit Vererbung *für Python*



Klassendiagramm mit Vererbung *für Java*



Schritt 0 – Bibliotheken einbinden

Lazarus	Wir leiten die Lampe von einer Label-Komponente ab. TLabel wird in ExtCtrls definiert. Zu ergänzen ist: `uses [... ] StdCtrls;`

Python	Wir benutzen für die GUI die Bibliothek tkinter und später einen Timer. Zu ergänzen ist:
	
	`import tkinter`
	`import random`
	`from tkinter import *`

Java	`import java.awt.*;`
	`import java.awt.event.*;`
	`import javax.swing.*;`
	`import javax.swing.event.*;`

Schritt 1 – Die Klasse TMineButton / MineButton

Das Ableiten und Erben von der visuellen Komponente TLabel / Label ist recht einfach. Die Vaterklasse (TLabel / Label) wird einfach in Klammern hinter die Sohnklasse gesetzt. Damit erbt die neue Klasse alle Attribute und Methoden der Vaterklasse.

Gleichzeitig deklarieren wir auch alle Attribute und Methoden.

Die Struktur der Ereignisprozeduren (*hier: MouseDown*) in **Lazarus** erhält man am leichtesten, wenn man für eine entsprechende Beispielkomponente auf dem Formular die Prozedur auswählt und dann die *Komponente* und *„Sender: TObject“* weglässt:

procedure ~~TForm1.Button1~~MouseDown ~~(Sender: TObject;~~
Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

Für die **Java**-Version braucht man den Constructor, in dem ein Listener später hinzugefügt werden kann.

Bei **Python** wird die Click-Methode separat als einfache Methode implementiert und über „bind“ an das Mausereignis <ButtonPress-1> für die linke Maustaste oder <ButtonPress-3> für die rechte Maustaste gebunden werden.

Lazarus

```
TMineButton = class(TLabel)
     public
      wert: integer;
      links, oben, rechts, unten : TMineButton;
      aufgedeckt : boolean;
      constructor create(AOwner: TComponent);  override;
      procedure NachbarnReferenzieren (o, r, u, l: TMineButton);
      procedure setGroessePosition(x, y: integer);
      procedure WertBerechnen;
      procedure Aufdecken;
      procedure MouseDown(Button: TMouseButton;
         Shift: TShiftState; X, Y: Integer); override;
```

Delphi

```
  end;

// Tipp: Maus in die Klasse und Strg+Shift+C drücken.
// Dann werden alle Methoden automatisch leer erzeugt :-)

procedure TMineButton.setGroessePosition(x, y: integer);
   begin end;

procedure TMineButton.WertBerechnen;
// alle 8 angrenzenden Felder müssen abgefragt werden
   begin  end;

procedure TMineButton.Aufdecken;
   begin  end;

constructor TMineButton.create(AOwner: TComponent);
begin
  inherited create(AOwner);
end;

procedure TMineButton.NachbarnReferenzieren(o, r, u, l: TMineButton);
begin
end;

procedure TMineButton.MouseDown(Button: TMouseButton;
       Shift: TShiftState; X,Y: Integer);
begin   end;
```

Python

```
class MineButton (Label):
    # Attribute müssen nicht unbedingt hier deklariert werden.
    # Aus Gründen der Übersicht mache ich es hier trotzdem.
    #Referenzen auf MineButton
    rechts = None
    links = None
    oben = None
    unten = None
    Wert = 0
    aufgedeckt = False

    def __init__ (self, master):
       Label.__init__(self, master)
       pass

    def setGroesse(self, inWidth, inHeight):
        pass

    def nachbarnReferenzieren (self, l, o, r, u):
        pass

    def WertBerechnen(self):
        pass

    def Aufdecken (self):
        pass
```

Java

```
public class MineButton extends JLabel {
    MineButton links, rechts,oben, unten;
    int Wert = 0;
    boolean aufgedeckt = false;

    MineButton () {
      this.setVerticalAlignment(JLabel.CENTER);
      this.setHorizontalAlignment(JLabel.CENTER);
```

```
    }

   void NachbarnReferenzieren (MineButton o, MineButton r,
          MineButton u, MineButton l) {   }

   void WertBerechnen () {  }

   void Aufdecken() {  }

   void setGroessePosition(int x, int y) {  }
  }
```

Anmerkung zu Python:
Statt der Deklaration der Attribute könnten auch „Slots“ benutzt werden.

```
__slots__ = ('rechts')
__slots__ = ('links')
```

Erläuterung: Eine Deklaration innerhalb einer Klasse, die Speicher spart, indem der Platz für Instanzattribute vorher deklariert wird und Exemplardictionaries eliminiert werden.

Schritt 2 – Die GUI

Nun sollen die *MineButtons* in der GUI erscheinen. In **Lazarus** und **Java** geht das recht einfach über die absolute Positionierung von visuellen Komponenten in einem Formular bzw. Frame, für **Python** legen wir die *MineButtons* in ein Gridlayout.

Zu beachten ist, dass wir zuerst ein größeres quadratisches Feld mit Referenzen auf (Nil, null, None) erzeugen, damit die Referenzierung der Nachbarlampen später in einer Zeile geschehen kann. Es wird dabei zunächst ein leeres Feld der Größe 17x17 mit möglichen Referenzen auf *MineButtons* definiert, dessen Referenzen zunächst aber noch auf (Nil, None, Null) zeigen. Danach werden für das innere 15x15 Feld 225 Objekte vom Typ „MineButton“ erzeugt.
Bei der Referenzierung der Nachbarfelder, um sie später evtl. aufdecken zu lassen, wird für jedes Objekt die Methode *referenzieren()* aufgerufen, die allerdings auch auf die nicht existierenden Lampen über den Rand hinaus zugegreift.
Durch das größere Feld werden bei dieser Methode keine Fehler geworfen und entsprechende Verweise zeigen gleich auf Null, Nil, None.

Falls diese Vorgehensweise für Schüler zu umständlich und verwirrend scheint, kann alternativ im Unterricht bei der Referenzierung fallweise unterschieden werden. Dabei wird der Quellcode allerdings etwas umfassender.

Lazarus

```
procedure TMineButton.setGroessePosition(x, y: integer);
begin
   self.Height:=30;
   self.Width:=30;
   self.Left:=x;
   self.Top:=y;
   self.Cursor:= crhandpoint;
end;
```

```
constructor TMineButton.create(AOwner: TComponent);
begin
  inherited create(AOwner);
  self.Font.Style := [fsbold];
  self.Font.Name := 'Tahoma';
  self.AutoSize := false;
  self.Alignment := taCenter;
  self.color := clltgray;
  self.layout := tlcenter;
  self.wert := 0;
  self.markiert := false;
  self.aufgedeckt:= false;
end;


procedure TForm1.FormCreate(Sender: TObject);
var Zeile, Spalte: integer;
begin
    // Damit das Referenzieren leichter geht
    for Zeile := 0 to 16 do
      for Spalte := 0 to 16 do
         begin
         a[Zeile,Spalte] := nil
         end;

    for Zeile := 1 to 15 do
      for Spalte := 1 to 15 do
         begin
         a[Zeile,Spalte] := TMineButton.create(self);
         a[Zeile,Spalte].Parent := Form1;
         a[Zeile,Spalte].setGroessePosition
                 (20 + (Spalte-1)*31, 20 + (Zeile-1)*31);
          // AUFPASSEN !!!!! Spalten , Zeilen , Größe +1
         end;

    // Fenstergröße
    self.Width:= 31*15 + 20*2;
    // (Kästchengröße +1 ) * Anzahl der Kästchen +  2* Rand
    self.Height:= 31*15 + 20*2; ;
end;
```

Python

```
Fenster = tkinter.Tk()
Fenster.title("Minesweeper")


a = [    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] ,
       ]
```

```
for Zeile in range (17):
 for Spalte in range (17):
     a[Zeile][Spalte] =  None

for Zeile in range (1,16):
  for Spalte in range (1,16):
     a[Zeile][Spalte] = MineButton(master = Fenster)
     a[Zeile][Spalte].config(width=5, height=2, cursor="hand2")
     a[Zeile][Spalte].config(font = "Arial 11 bold")
     a[Zeile][Spalte].grid(row=(Zeile+1),column=(Spalte+1),
                  padx =1, pady=1)
     a[Zeile][Spalte].config(bg="gray")

Fenster.mainloop()
```

Java

```
   void setGroessePosition(int x, int y) {
      this.setSize(30,30);
      this.setLocation(x,y);
      this.setOpaque(true);
      this.setBackground(Color.GRAY);
      this.setCursor(new Cursor(Cursor.HAND_CURSOR));
      this.setVisible(true);
    }

public Minesweeper(String title) {
[. . .]

MineButton[][] a = new MineButton[17][17];

    for (int Zeile=0; Zeile<17;Zeile++) {
      for (int Spalte=0; Spalte<17 ;Spalte++ ) {
        a[Zeile][Spalte] = null;
      }
    }

    for (int Zeile=1; Zeile<16;Zeile++) {
      for (int Spalte=1; Spalte<16 ;Spalte++ ) {
        a[Zeile][Spalte] = new MineButton();
        //a[Zeile][Spalte].setVerticalAlignment(JLabel.CENTER);
        //a[Zeile][Spalte].setHorizontalAlignment(JLabel.CENTER);
      } // end of for
    } // end of for

    for (int Zeile=1; Zeile<16;Zeile++) {
      for (int Spalte=1; Spalte<16 ;Spalte++ ) {
        cp.add(a[Zeile][Spalte]);
        a[Zeile][Spalte].setGroessePosition (20 + (Zeile-1)*31, 20 +
(Spalte-1)*31);
      }
    }

    frameWidth = 20*2 + 15*31;
    frameHeight = 20*3 + 15*31;
    setSize(frameWidth, frameHeight);

}
```

Schritt 3 – Werte berechnen

Liegt auf einem Feld eine Bombe, so wird dort dem Attribut "Wert" die -1 zugewiesen. Sonst geben die Werte die Anzahl der unmittelbaren Nachbarn mit Bombe wieder. Ist keine Bombe in der Nachbarschaft, wird "Wert" auf 0 gesetzt.

Lazarus

```
var Minenzahl : integer = 15;


procedure TMineButton.NachbarnReferenzieren(o, r, u, l: TMineButton);
begin
    self.links := l;
    self.oben := o;
    self.rechts := r;
    self.unten := u;
end;

procedure TMineButton.WertBerechnen;
// alle 8 angrenzenden Felder müssen abgefragt werden
begin
if self.wert <> -1 then
begin
 if (self.oben <> nil) and (self.oben.wert = -1) then
      self.wert:=self.wert +1;
 if (self.rechts <> nil) and (self.rechts.wert = -1) then
      self.wert:=self.wert +1;
 if (self.unten <> nil) and (self.unten.wert = -1) then
      self.wert:=self.wert +1;
 if (self.links <> nil) and (self.links.wert = -1) then
      self.wert:=self.wert +1;

 if (self.oben <> nil) and (self.oben.links <> nil) and
(self.oben.links.wert = -1) then
      self.wert:=self.wert +1;

 if (self.oben <> nil) and (self.oben.rechts <> nil) and
(self.oben.rechts.wert = -1) then
      self.wert:=self.wert +1;

 if (self.unten <> nil) and (self.unten.links <> nil) and
(self.unten.links.wert = -1) then
      self.wert:=self.wert +1;

 if (self.unten <> nil) and (self.unten.rechts <> nil) and
(self.unten.rechts.wert = -1) then
      self.wert:=self.wert +1;
end;

procedure TForm1.FormCreate(Sender: TObject);
var Zeile, Spalte, i, x, y: integer;
begin
[. . .]

for Zeile := 1 to 15 do
 for Spalte := 1 to 15 do
   begin
   //-->>  oben, rechts, unten, linke:
   a[Zeile,Spalte].NachbarnReferenzieren(a[Zeile-1, Spalte] ,
         a[Zeile,Spalte+1], a[Zeile+1,Spalte], a[Zeile, Spalte-1]);
   end;

  // Minen legen --> Minen können zufällig auch doppelt
  // belegt sein (macht hier aber nichts)
  randomize;

  for i:=1 to minenZahl do
     begin
     x :=  random (15)+1;
     y := random (15)+1;
     a[x,y].wert := -1; // Minen -1
  end;
```

```
   for Zeile := 1 to 15 do
      for Spalte := 1 to 15 do
         a[Zeile,Spalte].WertBerechnen;

 //!!!!!!!! Testausgabe !!!!!!!!!
   for Zeile := 1 to 15 do
      for Spalte := 1 to 15 do
         a[Zeile,Spalte].caption := IntToStr(a[Zeile,Spalte].wert);
end;
```

Python

```
MinenZahl = 15

def nachbarnReferenzieren (self, l, o, r, u):
    self.rechts = r
    self.links = l
    self.oben = o
    self.unten = u

def WertBerechnen(self):
   if self.Wert != -1:
       if (self.oben != None) and (self.oben.Wert == -1):
             self.Wert = self.Wert +1

       if (self.rechts != None) and (self.rechts.Wert == -1):
             self.Wert = self.Wert +1

       if (self.unten != None) and (self.unten.Wert == -1):
             self.Wert = self.Wert +1

       if (self.links != None) and (self.links.Wert == -1):
             self.Wert = self.Wert +1

       if (self.oben != None) and (self.oben.links != None) and
            (self.oben.links.Wert == -1):
                self.Wert = self.Wert +1

       if (self.oben != None) and (self.oben.rechts != None) and
          (self.oben.rechts.Wert == -1):
              self.Wert = self.Wert +1

       if (self.unten != None) and (self.unten.links != None) and
          (self.unten.links.Wert == -1):
              self.Wert = self.Wert +1

       if (self.unten != None) and (self.unten.rechts != None) and
          (self.unten.rechts.Wert == -1):
              self.Wert = self.Wert +1

[. . .]
Fenster = tkinter.Tk()
Fenster.title("Minesweeper")
[. . .]

for Zeile in range (1,16):
   for Spalte in range (1,16):
     a[Zeile][Spalte].nachbarnReferenzieren(a[Zeile][Spalte-1],
            a[Zeile-1][Spalte],a[Zeile][Spalte+1],a[Zeile+1][Spalte] )

# Minen legen -   Minen können zufällig auch doppelt belegt sein
#(soll hier aber nichts machen)

for i in range (1,MinenZahl):
    x = random.randint(1,15)
    y = random.randint(1,15)
    a[x][y].Wert = -1
    # Minen sind (-1)
```

```
for Zeile in range (1,16):
   for Spalte in range (1,16):
     a[Zeile][Spalte].WertBerechnen()



#Zur Anzeige der Zahlen beim Start hier auskommentieren:
for Zeile in range (1,16):
   for Spalte in range (1,16):
     a[Zeile][Spalte].config(text=(a[Zeile][Spalte].Wert))
     a[Zeile][Spalte].grid(ipadx=0, ipady=0)

Fenster.mainloop()
```

Java

```
int Minenzahl = 15;

void NachbarnReferenzieren (MineButton o, MineButton r,
     MineButton u, MineButton l) {
      this.oben = o;
      this.rechts = r;
      this.unten = u;
      this.links = l;
    }

void WertBerechnen () {
  if (this.Wert != -1) {
    if ((this.oben != null) && (this.oben.Wert == -1)) {
         this.Wert = this.Wert +1;
         }
    if ((this.rechts != null) && (this.rechts.Wert == -1)) {
         this.Wert = this.Wert +1;
         }
    if ((this.unten != null) && (this.unten.Wert == -1)) {
         this.Wert = this.Wert +1;
         }
    if ((this.links != null) && (this.links.Wert == -1)) {
         this.Wert = this.Wert +1;
         }

    if ((this.oben != null) && (this.oben.links != null) &&
        (this.oben.links.Wert == -1)) {
             this.Wert = this.Wert +1;
             }
    if ((this.oben != null) && (this.oben.rechts != null) &&
        (this.oben.rechts.Wert == -1)) {
             this.Wert = this.Wert +1;
             }
    if ((this.unten != null) && (this.unten.links != null) &&
         (this.unten.links.Wert == -1)) {
             this.Wert = this.Wert +1;
             }
    if ((this.unten != null) && (this.unten.rechts != null) &&
         (this.unten.rechts.Wert == -1)) {
             this.Wert = this.Wert +1;
             }
  }
}

for (int Zeile=1; Zeile<16;Zeile++) {
      for (int Spalte=1; Spalte<16 ;Spalte++ ) {
        // oben, rechts, unten, links
        a[Zeile][Spalte].NachbarnReferenzieren(a[Zeile-1][Spalte],
          a[Zeile][Spalte+1], a[Zeile+1][Spalte], a[Zeile][Spalte-1]);
      }
```

```
}

// Minen legen - zufällig auch doppelt (macht hier nichts)
Random r = new Random();
for (int i=1; i < Minenzahl+1; i++) {
      int Randx = r.nextInt(15)+1;
      int Randy = r.nextInt(15)+1;
      a[Randx][Randy].Wert = -1; // Minen -1
      }

for (int Zeile=1; Zeile<16; Zeile++) {
      for (int Spalte=1; Spalte<16 ; Spalte++ ) {
            a[Zeile][Spalte].WertBerechnen();
      }
    }

// Zur Überprüfung der richtigen Programmierung
// Ansonsten diese Zeilen auskommentieren !!!
for (int Zeile=1; Zeile<16; Zeile++) {
      for (int Spalte=1; Spalte<16 ; Spalte++ ) {

a[Zeile][Spalte].setText(Integer.toString(a[Zeile][Spalte].Wert));
      }
    }
```

Schritt 4 – Die Ereignisprozeduren

Es kommen die linke sowie rechte Maustaste zum Einsatz.

Bei **Lazarus** wird die MouseDown-Prozedur der TShape-Komponente überschrieben. Dazu wird die Prozedur in der Deklaration mit *override* gekennzeichnet und die Prozedur im Implementierungsteil einfach programmiert.

Die Click-Methode muss bei **Python** separate als einfache Methode implementiert und über die Methode „bind“ an das Mausereignis <ButtonPress-1> für die linke Maustaste bzw. <ButtonPress-3> für die rechte Maustaste gebunden werden.

```
widget.bind(event, handler)
```

linke Maustaste:	<Button-1>, <ButtonPress-1> oder <1>
mittlere Maustaste:	<Button-2>, <ButtonPress-2> oder <2>
rechte Maustaste	<Button-3>, <ButtonPress-3> oder <3>
linke Taste loslassen	<ButtonRelease-1>
Doppelclick	<Double-Button-1>

http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm

Python

```
    def __init__ (self, master):
       Label.__init__(self, master)
       self.bind(sequence="<ButtonPress-1>", func=self.LinksClick)
       self.bind(sequence="<ButtonPress-3>", func=self.RechtsClick)
```

In unserer **Java**-Version wird im Constructor einfach ein Mouse-Listener hinzugefügt, der bei **`mousePressed`** die Methode *LinksClick()* bzw. *RechtsClick()* aufruft. Die Maustasten werden durch das Attribut **`evt.isMetaDown() == true`** unterschieden. True bei rechter Maustaste, false bei linker Maustaste.

Lazarus

```
procedure TMineButton.MouseDown(Button: TMouseButton;
```

Lazarus

```
                Shift: TShiftState; X,Y: Integer);
begin
  if button = mbLeft then
  begin
    if self.wert > 0 then
      begin
      caption := inttostr(wert);
      color := clwhite
      end
    else if wert = -1 then
      begin
      caption := 'X';
      color := clred;
      end
    else if wert = 0 then
      begin
      self.Aufdecken;
      end;
  end;


  if button = mbRight then
  begin
    if color <> clyellow then
      color := clyellow
    else
      color := clltgray;
  end;

end;
```

Python

```
    def LinksClick(self, Event=None):
            if (self.Wert > 0):
               self.config(bg="white", text = self.Wert)
            elif (self.Wert == -1):
               self.config(bg="red", text = "X")
            elif (self.Wert == 0):
               self.Aufdecken()


    def RechtsClick(self, Event = None):
        if self["bg"] == "gray":
            self.config(bg="yellow")
        else:
            self.config(bg="gray")
```

Java

```
MineButton () {
      this.setVerticalAlignment(JLabel.CENTER);
      this.setHorizontalAlignment(JLabel.CENTER);

      this.addMouseListener(new MouseAdapter() {
        public void mousePressed(MouseEvent evt) {
          if (evt.isMetaDown() == true) {
              //rechte Maustaste- bei false linke Maustaste
             RechtsClick();
          } else {
             LinksClick();
             }
          }
      });
} // Ende MineButton

void LinksClick() {
      if (this.Wert > 0) {
          this.setBackground(Color.WHITE);
          this.setText(Integer.toString(this.Wert));
```

```
      } else {
          if (this.Wert == -1) {
              this.setText("X");
              this.setBackground(Color.RED);
          } else {
              if (this.Wert == 0) {
                  this.Aufdecken();
                  } // end of if
          } // end of if-else
      } // end of if-else
}

void RechtsClick() {
     if (this.getBackground() == Color.GRAY) {
          this.setBackground(Color.YELLOW);
          } else {
          this.setBackground(Color.GRAY);
          } // end of if-else
}
```

Schritt 5 – weiße Nachbarfelder aufdecken

Wird ein leeres Feld getroffen, so deckt es sich auf und sendet an alle Nachbarn, sich evtl. auch aufzudecken und dann die Nachricht weiterzuleiten. Damit keine Rückkopplung entsteht, wird bei einem aufgedeckten Feld das Attribut "aufgedeckt" auf true gesetzt.

Lazarus

```
procedure TMineButton.Aufdecken;
begin
if self.wert = 0 then
  begin
     self.Color:= clwhite;
     self.aufgedeckt:=true;


    if self.oben <> nil then
        if (self.oben.wert = 0) and (self.oben.aufgedeckt=false) then
           self.oben.Aufdecken;

     if self.rechts <> nil then
        if (self.rechts.wert = 0)  and (self.rechts.aufgedeckt=false) then
          self.rechts.Aufdecken;

     if self.unten <> nil then
        if (self.unten.wert = 0)  and (self.unten.aufgedeckt=false) then
          self.unten.Aufdecken;

     if self.links <> nil then
        if (self.links.wert = 0)  and (self.links.aufgedeckt=false) then
          self.links.Aufdecken;

  end;
end;
```

Python

```
def Aufdecken (self):
 if (self.Wert == 0):
    self.config(bg="white")
    self.aufgedeckt = True

    if self.oben != None:
       if (self.oben.Wert == 0) and (self.oben.aufgedeckt == False):
           self.oben.Aufdecken()
```

```
    if self.rechts != None:
      if (self.rechts.Wert == 0) and (self.rechts.aufgedeckt == False):
           self.rechts.Aufdecken()

    if self.unten != None:
      if (self.unten.Wert == 0) and (self.unten.aufgedeckt == False):
           self.unten.Aufdecken()

    if self.links != None:
      if (self.links.Wert == 0) and (self.links.aufgedeckt == False):
           self.links.Aufdecken()
```

Java

```
void Aufdecken() {
  if (this.Wert == 0){
    this.setBackground(Color.WHITE);
    this.aufgedeckt = true;
    if (this.oben != null) {
      if ((this.oben.Wert == 0) && (this.oben.aufgedeckt == false)) {
          this.oben.Aufdecken();
          }
    }
    if (this.rechts != null) {
      if ((this.rechts.Wert == 0) && (this.rechts.aufgedeckt == false)) {
          this.rechts.Aufdecken();
      }
    }

    if (this.unten != null) {
      if ((this.unten.Wert == 0) && (this.unten.aufgedeckt == false)) {
          this.unten.Aufdecken();
      }
    }

    if (this.links != null) {
      if ((this.links.Wert == 0) && (this.links.aufgedeckt == false)) {
          this.links.Aufdecken();
      }
    }
  }
}
```