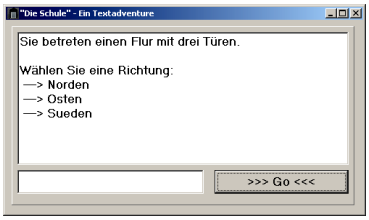
Zum Spiel

Entwicklung der Adventure-Konsolenspiele

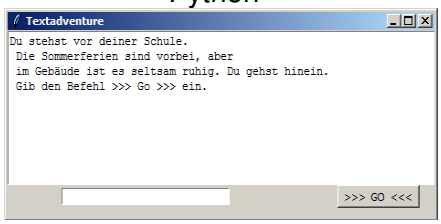
Textadventures sind Computerspiele der „ersten Generation“, also Spiele, bei denen der Spieler nur über Texteingaben mit dem Computer kommunizieren kann. Seine Figur steuert er über Befehle wie „go“, „east“, „west“, „pick up“, „turn“, etc. Die ersten Textadventures waren Don Woods’ „Collossal Cave Adventure“ (1976) und das von MIT-Studenten programmierte “Zork” (1977).

Das hier vorgestellte Textadventure orientiert sich stark an diesen ursprünglichen Spielen, soll Schüler/innen aber anregen, ihr eigenes Spiel zu entwickeln und umzusetzen.

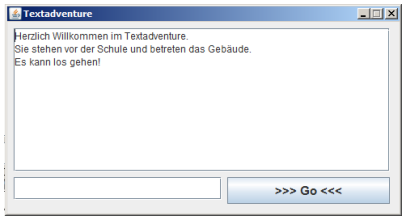
Lazarus



Python



Java



Zum objektorientierten Ansatz

Prinzipiell kann ein Textadventure imperativ als Folge von if-then-else- oder case-Anweisungen über das Modell eines Zustandsdiagramms implementiert werden. Die Räume würden dann durch Zustände repräsentiert werden. Diese Implementierung erweist sich jedoch aufgrund der geringen Erweiterbarkeit gegenüber einem objektorientierten Ansatz als deutlich schwieriger und wenig motivierend.

Beim objektorientierten Ansatz werden die Räume durch Objekte eines Grundtyps ***TRaum*** repräsentiert. Allen Räumen gemeinsam sind Raumbeschreibungen, Ausgänge, mögliche herumliegende Gegenstände, etc. Die Räume unterscheiden sich in der Belegung ihrer Attribute. Ausgänge von einem Raum zum anderen werden über eine Referenz des entsprechenden Objekts zu einem anderen nachgehalten, der Raumwechsel erfolgt über eine Referenz auf das Raumobjekt, in dem sich der Spieler aktuell *befinden* soll. Beim Raumwechsel werden jeweils die Attribute des Raums, auf den die Referenz des *Spielerraums* zeigt, ausgegeben.

Das hier vorgestellte Projekt soll den ersten Kontakt zur Objektorientierten Programmierung herstellen. Anhand der Entwicklung und Erweiterung eines Textadventures zu einem "großen" Abenteurspiel lernen Schüler die Vorteile eines objektorientierten Ansatzes kennen, indem sie mit Klassen und Objekten umgehen und gleichzeitig die vielen Vorteile entdecken, die ein objektorientierter Ansatz bietet.

Das Ziel der ersten Berührung mit der OOP sollte es sein, didaktisch reduziert alle weiterführenden Konzepte wie Konstruktoren, Methoden einer Klasse, Ableitung, Datenkapselung, MVC-Konzept, etc. auszublenden und statt dessen das Fundamentale Konzept der **Klasse als Entwurfsmuster für Objekte mit unterschiedlichen inneren Zuständen** herauszuheben. Dazu dürfen nicht viele Klassen mit wenigen Objekten betrachtet werden, sondern **wenige Klassen mit möglichst vielen davon erzeugten Objekten**.

Das Projekt **„Textadventure“** ist konzipiert als Einstieg in die Objektorientierung. Dieser Ausrichtung folgt die dargestellte Vorgehensweise. Viele zentrale Aspekte der

Objektorientierung (Klassenmethoden, Konstruktoren, Datenkapselung, Hat- und Kennt-Beziehung, Vererbung, UML-Modellierung) werden noch nicht eingeführt.

Im Fokus stehen weniger streng programmiertechnische Formalismen - diese sollten in Folgeprojekten unterrichtet werden - als vielmehr der aktive und intuitive Umgang der Schüler mit Objekten und Klassen. Es geht um das Erfahrbarmachen der grundlegenden objektorientierten Idee.

Ein besonderer Vorteil des Textadventures zeigt sich hier in der Möglichkeit der Erweiterbarkeit durch die Schüler/innen. Im Ergänzen von Räumen und Erweitern der Klassen durch Attribute üben sie den Umgang mit Objekten. Über die Ausgabe der Raumbeschreibung und referenzierten Raumobjekte erfassen die Schüler den Zustand eines Objekts als Belegung seiner Attribute.

Schritt 0 – Bibliotheken einbinden und GUI aufstellen

Lazarus

```
Button1: TButton;
Edit1: TEdit;
Memo1: TMemo;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Memo1.append ('Hallo, willkommen im Textadventure! Du stehst
         vor deiner Schule. Die Sommerferien sind vorbei,
         aber im Gebäude ist es seltsam ruhig.
         Du gehst hinein. ');
end;
```

Python

```
import tkinter
import tkinter.scrolledtext

[...]

#GUI
main = tkinter.Tk()
main.title("Textadventure")
TextBox = tkinter.scrolledtext.ScrolledText(main,
             width=70, height = 10)
TextBox.grid(row=1 , column = 1, columnspan=2)
TextBox.insert("end", "Du stehst vor deiner Schule. \n\
Die Sommerferien sind vorbei, aber im Gebäude ist es seltsam ruhig. \n\
Du gehst hinein. \n\
Gib den Befehl >>> Go >>> ein.\n \n")

TextFeld = tkinter.Entry(main, width = 30)
TextFeld.grid(row=2, column = 1)
Button = tkinter.Button(main, text = " >>>  GO  <<< ",
        command = ButtonClick)
Button.grid(row=2, column = 2)

main.mainloop()
```

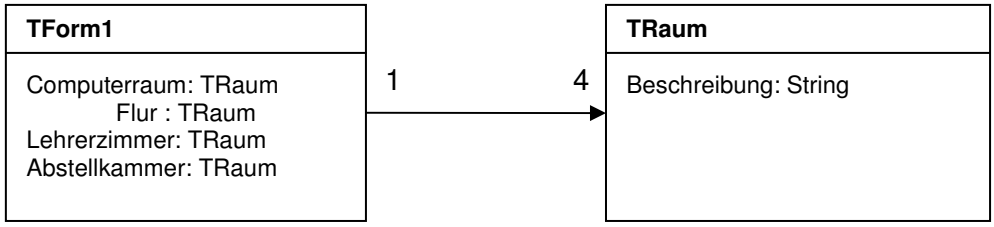
Java

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

[...]

  private JTextField jTextField1 = new JTextField();
  private JTextArea jTextArea1 = new JTextArea("");
  private JButton jButton1 = new JButton();
  private JScrollPane jTextArea1ScrollPane =
            new JScrollPane(jTextArea1);
```

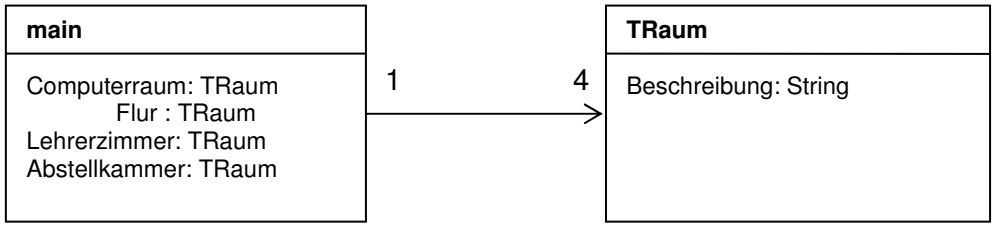
	[...] jTextArea1.append("Herzlich Willkommen im Textadventure.\n"); jTextArea1.append("Du stehst vor der Schule und betrittst \n das Gebäude. Es kann los gehen!");

Schritt 1 - Das Klassendiagramm und die Klasse TRaum

Klassendiagramm Lazarus



Klassendiagramm Python / Java



Lazarus

```
type
  TRaum = class
     Beschreibung : String;
  end;

var
  Form1: TForm1;
  Computerraum, Flur, Lehrerzimmer, Abstellkammer : TRaum;

implementation
{$R *.dfm}
```

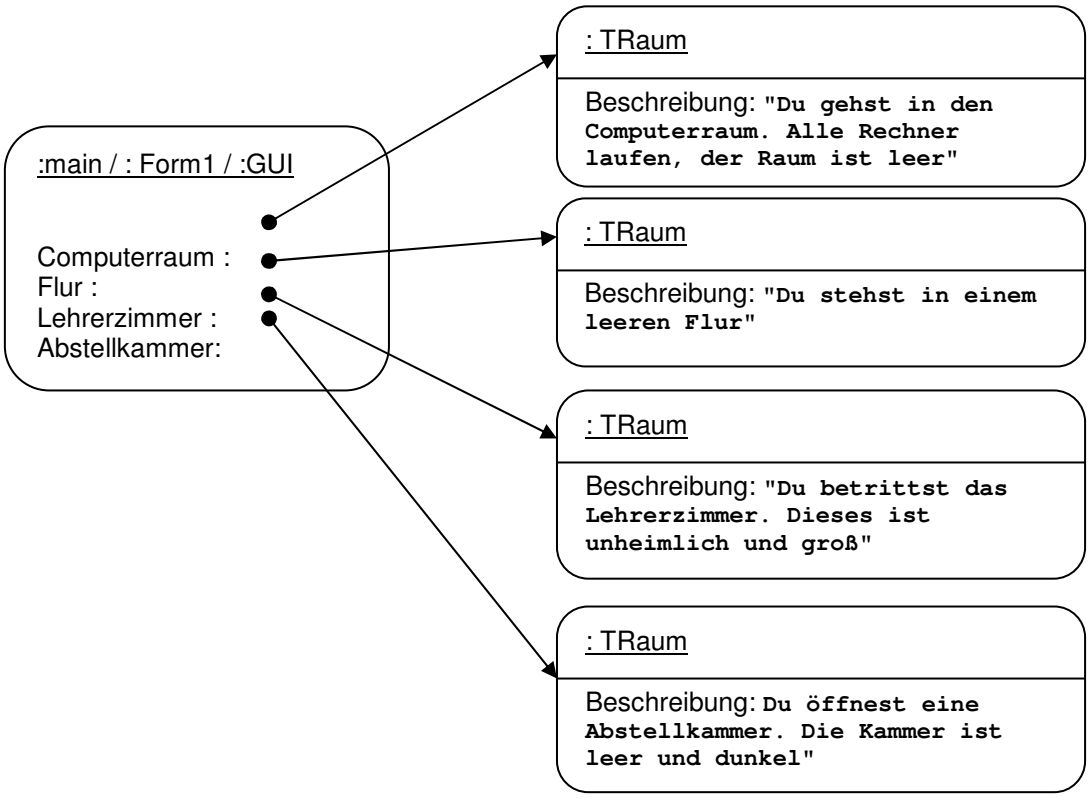
Python

```
# Definition der Klasse TRaum
class TRaum:
    Beschreibung = ""
```

Java

```
public class Textadventure2016 extends JFrame {
  private JTextArea jTextArea1 = new JTextArea("");
  private JScrollPane jTextArea1ScrollPane = new
            JScrollPane(jTextArea1);
  private JButton jButton1 = new JButton();
  private JTextField jTextField1 = new JTextField();

public class TRaum {
    public String Beschreibung;
    }
```

Schritt 2 – Räume erzeugen (Objektdiagramm)



Lazarus

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Memo1.Lines.Add ('Hallo, willkommen im Textadventure');
  Computerraum := TRaum.Create;
  Flur := TRaum.Create;
  Lehrerzimmer := TRaum.Create;
  Abstellkammer := TRaum.Create;
  Computerraum.Beschreibung := ('Sie stehen im Computerraum,
                                 ein Rechner läuft.');
  Flur.Beschreibung := ('Sie betreten einen Flur mit
      drei Türen.');
  Lehrerzimmer.Beschreibung := ('Das ist das Lehrerzimmer,
                                 alle Fächer sind leer.');
  Abstellkammer.Beschreibung := ('Das ist eine Abstellkammer, sie
ist leer.');
end;
```

Python

```
# Räume erzeugen
Flur = TRaum()
Lehrerzimmer = TRaum()
Computerraum = TRaum()
Abstellkammer = TRaum()


#Attribut "Beschreibung" der Räume zuweisen
Flur.Beschreibung = "Du stehst in einem leeren Flur \n"
Lehrerzimmer.Beschreibung = "Du betrittst das Lehrerzimmer.  \
Es ist unheimlich und groß. \n"
Computerraum.Beschreibung = "Du gehst in den Computerraum. \
Alle Rechner laufen, der Raum ist leer. \n"
Abstellkammer.Beschreibung = "Du öffnest eine Abstellkammer. \
Die Kammer ist leer und dunkel. \n"
```

Java

```
public class TRaum {
    public String Beschreibung;
    }

// Räume erzeugen und Attribute setzen
  TRaum Computerraum = new TRaum();
  TRaum Flur = new TRaum();
  TRaum Lehrerzimmer = new TRaum();
  TRaum Abstellkammer = new TRaum();

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

Flur.Beschreibung = "Sie stehen im Flur, es gehen drei Türen ab.";
    Lehrerzimmer.Beschreibung = "Hier ist das Lehrerzimmer, alle Fächer
sind leergeräumt.";
    Abstellkammer.Beschreibung = "Man sieht eine leere Abstellkammer.";
    Computerraum.Beschreibung = "sie betreten den Computerraum. Alle
Rechner sind aus!";

}
```

Schritt 3 – Referenz auf den aktuellen Raum und Referenzen für die Ausgänge

Referenzvariable: aktueller Raum

Lazarus

```
   var
     Form1: TForm1;
     Computerraum, Flur, Lehrerzimmer, Abstellkammer : TRaum;
     aktuellerRaum : TRaum;

   implementation
   {$R *.dfm}

   procedure TForm1.Button1Click(Sender: TObject);
     begin
     aktuellerRaum := Flur;
     Memo1.Lines.add (aktuellerRaum.Beschreibung);
     end;
```

Python

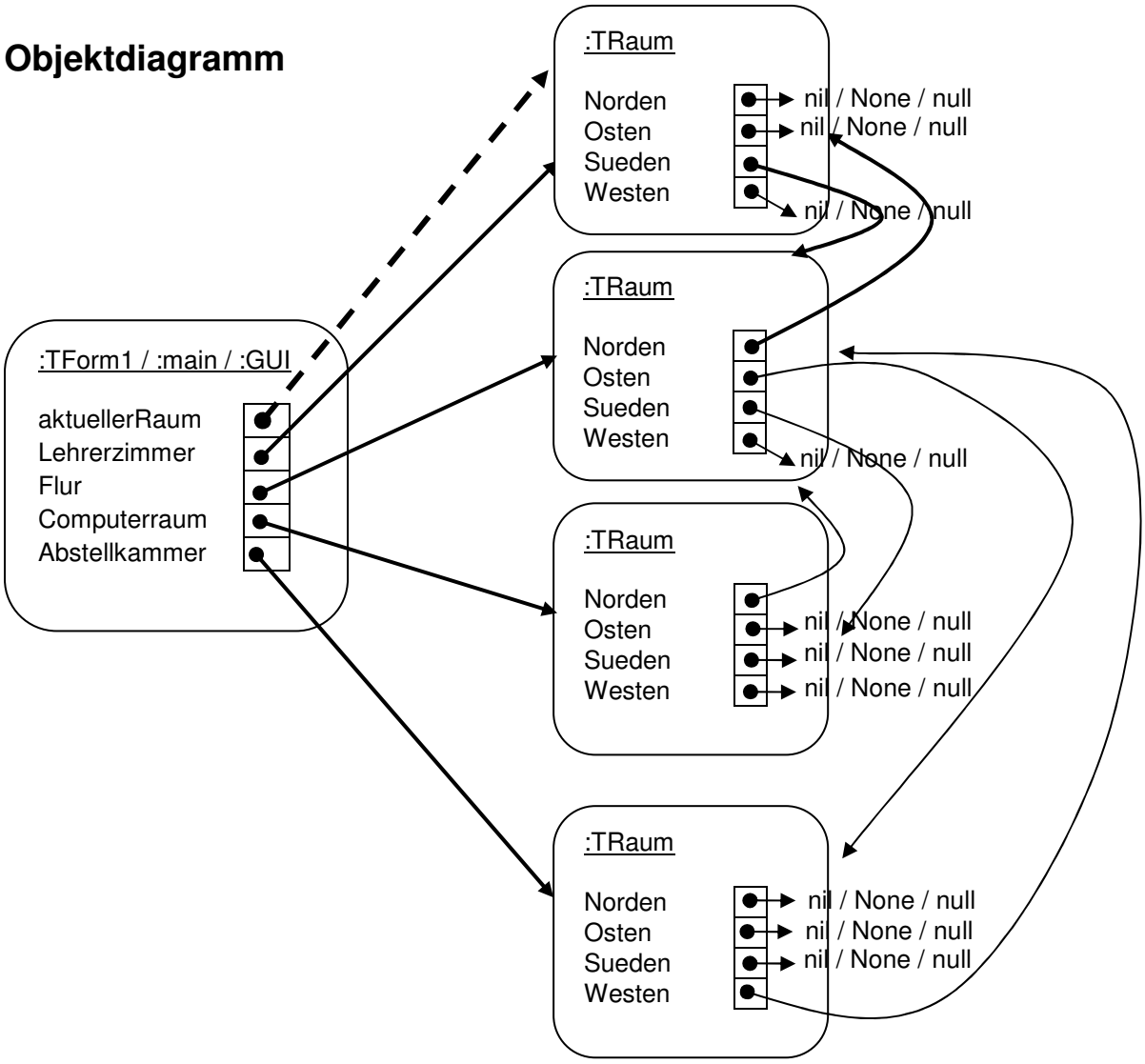
```
     # Räume erzeugen
     Flur = TRaum()
     Lehrerzimmer = TRaum()
     Computerraum = TRaum()
     Abstellkammer = TRaum()
     aktuellerRaum = Flur



     # ButtonClick - Prozedur
     def ButtonClick():
             TextBox.see("end")
             global aktuellerRaum
             TextBox.insert("end",aktuellerRaum.Beschreibung)
```

Java

```
public class TRaum {

  }

   TRaum aktuellerRaum;
```

```
public GUI(String title) {
[... ]
    aktuellerRaum = Flur;
}
```

Objektdiagramm



Ausgänge deklarieren und setzen

Lazarus

```
type
  TRaum = class
     Beschreibung : String;
     Norden, Sueden, Westen, Osten : TRaum;
     procedure setAusgaenge (nord, ost, sued, west : TRaum);
  end;

var
[ . . . ]
implementation
{$R *.dfm}

procedure TRaum.setAusgaenge (nord, ost, sued, west :TRaum);
begin
  // bitte auf den Parameter self achten
  self.Norden := nord;
```

Lazarus

```
      self.Osten := ost;
      self.Sueden := sued;
      self.Westen := west;
    end;

    procedure TForm1.FormCreate(Sender: TObject);
    begin
    […]
    Flur.setAusgaenge(Lehrerzimmer, Abstellkammer, Computerraum, nil);
    Computerraum.setAusgaenge(Flur, nil, nil, nil);
    Abstellkammer.setAusgaenge(nil, nil, nil, Flur);
    Lehrerzimmer.setAusgaenge(nil, nil, Flur, nil);
    aktuellerRaum := Flur;
    end;
```

Python

```
    class TRaum:
        Beschreibung = ""   #String Variablen zur Raumbeschreibung
        Norden = None   # Norden, Sueden, Westen, Osten sind
        Sueden = None   # Attribute vom Typ TRaum.
        Osten = None    # Sie zeigen auf Raumobjekte.
        Westen = None   # None ist das Schlüsselwort für 'kein Objekt'

        def setAusgaenge(self, nord, ost, sued, west):
            self.Norden = nord
            self.Sueden = sued
            self.Osten = ost
            self.Westen = west

    #Ausgänge – Räume referenzieren
    Flur.setAusgaenge (Lehrerzimmer, Abstellkammer, Computerraum,
    None)
    Lehrerzimmer.setAusgaenge (None, None, Flur, None)
    Abstellkammer.setAusgaenge (None, None, None, Flur)
    Computerraum.setAusgaenge (Flur, None, None, None)
```

Java

```
public class TRaum {

    public String Beschreibung;

    TRaum Norden;
    TRaum Osten;
    TRaum Sueden;
    TRaum Westen;

void setzeAusgaenge (TRaum Norden, TRaum Osten,
                     TRaum Sueden, TRaum Westen) {
      this.Norden = Norden;
      this.Osten = Osten;
      this.Sueden = Sueden;
      this.Westen = Westen;
      }
  }
```

Schritt 4 - Raum wechseln

Lazarus

```
    procedure TForm1.Button1Click(Sender: TObject);
      var Eingabe : String;
      begin
      Memo1.clear;
      Eingabe := Edit1.text;
      // --- Raum wechseln ---
      if Eingabe = 'Norden' then aktuellerRaum := aktuellerRaum.Norden;
      if Eingabe = 'Osten' then aktuellerRaum := aktuellerRaum.Osten;
```

	if Eingabe = 'Sueden' then aktuellerRaum := aktuellerRaum.Sueden; **if Eingabe = 'Westen' then aktuellerRaum := aktuellerRaum.Westen;** ***// --- Beschreibung des aktuellen Raums ausgeben*** **Memo1.Lines.add (aktuellerRaum.Beschreibung);** end;
Python	# ButtonClick - Prozedur def ButtonClick(): TextBox.see("end") global aktuellerRaum **Befehl = TextFeld.get()** **if Befehl == "Go":** **aktuellerRaum = Flur** **if (Befehl == "Norden"):** **aktuellerRaum = aktuellerRaum.Norden** **if (Befehl == "Sueden"):** **aktuellerRaum = aktuellerRaum.Sueden** **if (Befehl == "Osten"):** **aktuellerRaum = aktuellerRaum.Osten** **if (Befehl == "Westen"):** **aktuellerRaum = aktuellerRaum.Westen** TextBox.insert("end",aktuellerRaum.Beschreibung) **TextBox.insert("end","\n---------- \n")** **TextBox.see("end") #TextBox ans Ende scrollen** **TextFeld.delete(0,100)**
Java	public void jButton1_ActionPerformed(ActionEvent evt) { jTextArea1.setText(""); **String Eingabe = jTextField1.getText();** **if (Eingabe.equals("Norden"))** **{** **aktuellerRaum = aktuellerRaum.Norden;** **}** **if (Eingabe.equals("Osten")) {** **aktuellerRaum = aktuellerRaum.Osten;** **}** **if (Eingabe.equals("Sueden")) {** **aktuellerRaum = aktuellerRaum.Sueden;** **}** **if (Eingabe.equals("Westen")) {** **aktuellerRaum = aktuellerRaum.Westen;** **}** **jTextArea1.append(aktuellerRaum.Beschreibung + "\n\n");** }

Schritt 5 – Richtung angeben, in der ein Raum verlassen werden kann

Lazarus	procedure TForm1.Button1Click(Sender: TObject); var Eingabe : String; begin Eingabe := Edit1.text; *// --- Raum wechseln ---* if Eingabe = 'Norden' then aktuellerRaum := aktuellerRaum.Norden; if Eingabe = 'Osten' then aktuellerRaum := aktuellerRaum.Osten; if Eingabe = 'Sueden' then aktuellerRaum := aktuellerRaum.Sueden; if Eingabe = 'Westen' then aktuellerRaum := aktuellerRaum.Westen; *// --- Beschreibung des aktuellen Raums ausgeben* Memo1.Lines.add (aktuellerRaum.Beschreibung); ***// --- mögliche Ausgänge der Räume angeben ---***

	if aktuellerRaum.Norden <> nil then Memo1.Lines.add (' ---> Norden'); **if aktuellerRaum.Osten <> nil then Memo1.Lines.add (' ---> Osten');** **if aktuellerRaum.Sueden <> nil then Memo1.Lines.add (' ---> Sueden');** **if aktuellerRaum.Westen <> nil then Memo1.Lines.add (' ---> Westen');** end;
Python	TextBox.insert("end",aktuellerRaum.Beschreibung) **TextBox.insert("end","Folgende Ausgänge können genommen werden: \n")** **if (aktuellerRaum.Norden != None):** **TextBox.insert("end","--> Norden\n")** **if (aktuellerRaum.Osten != None):** **TextBox.insert("end","--> Osten\n")** **if (aktuellerRaum.Sueden != None):** **TextBox.insert("end","--> Sueden\n")** **if (aktuellerRaum.Westen != None):** **TextBox.insert("end","--> Westen\n")** TextBox.insert("end","\n---------- \n") TextBox.see("end") #TextBox ans Ende scrollen TextFeld.delete(0,100)
Java	public void jButton1_ActionPerformed(ActionEvent evt) { jTextArea1.setText(""); String Eingabe = jTextField1.getText(); [...] if (Eingabe.equals("Westen")) { aktuellerRaum = aktuellerRaum.Westen; } jTextArea1.append(aktuellerRaum.Beschreibung + "\n\n"); **if (aktuellerRaum.Norden != null) {** **jTextArea1.append(" ---> Norden \n");** **}** **if (aktuellerRaum.Osten != null) {** **jTextArea1.append(" ---> Osten \n");** **}** **if (aktuellerRaum.Sueden != null) {** **jTextArea1.append(" ---> Sueden \n");** **}** **if (aktuellerRaum.Westen!= null) {** **jTextArea1.append(" ---> Westen \n");** **}** }

Schritt 6 – Eingabefehler abfangen

Lazarus	procedure TForm1.Button1Click(Sender: TObject); var Eingabe : String; begin Eingabe := Edit1.text; // --- Raum wechseln --- if (Eingabe = 'Norden') **and (aktuellerRaum.Norden <> nil)** then aktuellerRaum := aktuellerRaum.Norden; if (Eingabe = 'Osten') **and (aktuellerRaum.Osten <> nil)** then aktuellerRaum := aktuellerRaum.Osten;

	`if (Eingabe = 'Sueden') and (aktuellerRaum.Sueden <> nil) then` `aktuellerRaum := aktuellerRaum.Sueden;` `if (Eingabe = 'Westen') and (aktuellerRaum.Westen <> nil) then` `aktuellerRaum := aktuellerRaum.Westen;` `// --- Beschreibung des aktuellen Raums ausgeben` `Memo1.Lines.add (aktuellerRaum.Beschreibung);` `// --- mögliche Ausgänge der Räume angeben ---` `[ . . . ]` `end;`
Python	`if (Befehl == "Norden") and (aktuellerRaum.Norden != None):` `aktuellerRaum = aktuellerRaum.Norden` `if (Befehl == "Sueden") and (aktuellerRaum.Sueden != None) :` `aktuellerRaum = aktuellerRaum.Sueden` `if (Befehl == "Osten") and (aktuellerRaum.Osten != None) :` `aktuellerRaum = aktuellerRaum.Osten` `if (Befehl == "Westen") and (aktuellerRaum.Westen != None) :` `aktuellerRaum = aktuellerRaum.Westen`
Java	`public void jButton1_ActionPerformed(ActionEvent evt) {` `jTextArea1.setText("");` `String Eingabe = jTextField1.getText();` `if ((Eingabe.equals("Norden")) && (aktuellerRaum.Norden != null))` `{` `aktuellerRaum = aktuellerRaum.Norden;` `}` `if ((Eingabe.equals("Osten")) && (aktuellerRaum.Osten != null)) {` `aktuellerRaum = aktuellerRaum.Osten;` `}` `if ((Eingabe.equals("Sueden")) && (aktuellerRaum.Sueden != null)) {` `aktuellerRaum = aktuellerRaum.Sueden;` `}` `if ((Eingabe.equals("Westen")) && (aktuellerRaum.Westen != null)) {` `aktuellerRaum = aktuellerRaum.Westen;` `}` `jTextArea1.append(aktuellerRaum.Beschreibung + "\n\n");` `[...]` `}`