

- <<interface>> markieren
- Pfeile bei Interface Vererbung gestrichelt
- Pfeilspitzen gefüllt (Klassenvererbung)
- Gerichtete Verbindung mit Kopf (nicht gefüllt)
- Gerichtete Linie von Observer zu Subject (Komposition, Aggregation)
- Verbindung von Observer Test zu TV lösen -> stattdessen Kasten um Observer Test
- Einfache Assoziationen | einfache Linien zu den Objekten

Scoutverwaltung

StringBuilder bauen in **Scout.java**

```
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Name:")
        .append(firstname)
        .append(" ")
        .append(lastname)
        .append("\n")
        .append("Geburtsdatum: ")
        .append(birthdate)
        .append("\n")
        .append("Aktiv: ")
        .append(active);

    return sb.toString();
}
```

Felder:

```
public class Scout {
    private int id;
    private String firstname;
    private String lastname;
    private LocalDate birthdate;
    private boolean active;
}
```

Scanner Eingabe to UpperCase setzen:

```
while(!exit) {
    switch(scanner.next().toUpperCase()) {
        case "N": System.out.println("New Scout");
            break;
    }
}
```

In App.java

```
public class App {
    private List<Scout> scouts = new ArrayList<>();

    private static final String CONTROLS = "\n-----\n"
        + " N: Neuen Scout anlegen\n"
        + " L: Alle Scouts auflisten\n"
        + " X: Programm verlassen\n"
        + "-----\n";

    public static void main(String[] args) {
    }
```

Controls setzen in psvm:

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("START");
    System.out.println(CONTROLS);

    App app = new App();
}
```

Neuen Scout erzeugen (in App.java):

```
while(!exit) {  
    switch(scanner.next().toUpperCase()) {  
        case "N":  
            Scout s = new Scout();  
            s.setFirstname(scanner.next());  
            s.setLastname(scanner.next());  
            // TODO: Deutsches Datumsformat  
            s.setBirthdate(LocalDate.of(scanner.nextInt(), sca  
            break;  
        case "l": System.out.println("List of Scout");
```

Oder parse benutzen:

```
s.setLastname(scanner.next());  
// 2010-06-10  
//TODO: Deutsches Datumsformat  
s.setBirthdate(LocalDate.parse(scanner.next()));  
s.setActive(scanner.nextBoolean());  
break;
```

//TODO Feature in NetBeans

Wo wird Scout abgespeichert?

Im Bereich vom Garbage Collection (temporär)

➔ Dauerhafte Aufbewahrung: Lokal im Programm: ArrayList -> später in Datei

Also (Apps.java):

```
s.setActive(scanner.nextBoolean());  
scouts.add(s);  
break;
```

bzw:

```
s.setActive(scanner.nextBoolean());  
app.scouts.add(s);  
break;
```

Liste (in App.java):

Konflikt mit scout s von ,N' -> Exception

```
//TODO: Labels  
Scout s = new Scout();  
s.setFirstname(scanner.next());  
s.setLastname(scanner.next());  
// 2010-06-10  
//TODO: Deutsches Datumsformat  
s.setBirthdate(LocalDate.parse(scanner.next()));  
s.setActive(scanner.nextBoolean());  
app.scouts.add(s);  
System.out.println("Scout wurde gespeichert.");  
break;  
  
case "L":  
    System.out.println("\n-----");  
    for(Scout s : app.scouts) {
```

```
        case "L":  
        {  
            System.out.println("\n-----");  
            for(Scout s : app.scouts) {  
                I  
            }  
            System.out.println("\n-----\n\n");  
        }  
        break;
```

➔ Also: mit {}

File Input/Output Stream:

Try with Resources <> normaler Try

Ressourcen implementieren ist. Beim Rausgehen wird auch die Ressource geschlossen.

Helper Klasse

Bsp: ArrayKlassen

Bsp: **ScoutHelper**

Klasse erzeugen, Scout importieren

```
package de.gfn.oca.scoutcamp.helper;

import de.gfn.oca.scoutcamp.entity.Scout;
import java.util.Scanner;

/**
 * @author tlubowiecki
 */
public class ScoutHelper {

    public static Scout setByInput(Scout scout, Scanner scanner) {

        return scout;
    }
}
```

TODOS: Eingaben validieren, Alte Werte anzeigen

Übernehmen der Input-Abfragen aus

```
public static Scout setByInput(Scout scout, Scanner scanner) {

    // TODO: Eingaben validieren
    // TODO: Alten Wert anzeigen
    // TODO: Mehrsprachigkeit

    System.out.print("Vorname: ");
    s.setFirstname(scanner.next());

    s.setLastname(scanner.next());
    // 2010-06-10
    //TODO: Deutsches Datumsformat
    s.setBirthdate(LocalDate.parse(scanner.next()));
    s.setActive(scanner.nextBoolean());

    return scout;
}
```

Für deutsches Datumsformat -> Parser:

```
public class ScoutHelper {

    public final static String DATE_FORMAT = "dd.MM.yyyy";
}
```

+ Formatter (Factory Method, man versteckt die Konstruktoren und gibt dem Entw ein fertiges Objekt zur Verwendung)

```
final static String DATE_FORMAT = "dd.MM.yyyy";
final static DateTimeFormatter FORMATTER = DateTimeFormatter.ofPattern(DATE_FORMAT);
```

(ScoutHelper.java)

```
scout.setLastname(scanner.next());

//TODO: Deutsches Datumsformat
System.out.print("Geburtsdatum (TT.MM.JJJJ): ");
scout.setBirthdate(LocalDate.parse(scanner.next(), FORMATTER));

System.out.print("Aktiv: ");
s.setActive(scanner.nextBoolean());

return scout;
}
```

Helper einbauen (jetzt bisheriges InputGedöns in App.java löschen):

```
case "N":
{
    System.out.println("Gib die Daten des Scouts ein.");
    Scout s = new Scout();
    app.scouts.add(s);
    System.out.println("Scout wurde gespeichert.");
}
break;
```

Noch kürzer:

```
switch(scanner.next().toUpperCase()) {

    case "N":
    {
        System.out.println("Gib die Daten des Scouts ein.");
        app.scouts.add(ScoutHelper.setByInput(new Scout(), scanner));
        System.out.println("Scout wurde gespeichert.");
    }
    break;
```

Klammern in den Cases entfernen da kein Konflikt mehr mit dem s-Objekt

```
switch(scanner.next().toUpperCase()) {

    case "N":
        System.out.println("Gib die Daten des Scouts ein.");
        app.scouts.add(ScoutHelper.setByInput(new Scout(), scanner));
        System.out.println("Scout wurde gespeichert.");
        break;

    case "L":
        System.out.println("\n-----");
        for(Scout s : app.scouts) {
            System.out.println(s);
        }
        System.out.println("\n-----\n\n");
        break;

    case "X": System.out.println("Exit");
        exit = true;
        break;
```


Serialisierung

Klasse erzeugen:

```
package de.gfn.oca.persistanze;

import java.io.FileInputStream;
import java.io.FileNotFoundException;

/**
 *
 * @author tlubowiecki
 */
public class SerializeTest {

    public static void main(String[] args) throws FileNotFoundException {

        try(FileInputStream fis = new FileInputStream("test.ser")) {

        }

    }

}
```

Exception Handling:

```
try(FileInputStream fis = new FileInputStream("test.ser")) {

}
catch(FileNotFoundException ex) {
    System.out.println("Datei nicht gefunden");
}
catch(IOException ex) {
    System.out.println("Daten konnten nicht gelesen werden");
}
```

Datei einlesen

```
// Datei einlesen
try(FileOutputStream fis = new FileOutputStream("test.ser")) {

}
catch(FileNotFoundException ex) {

}
```

Klasse für Speichern mit getter u setter:

```
class SerTestObj {

    private String text;

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

}
```

FileReaderWriter(Unicode Zeichen) <> **InputStream** (Binärdaten)

Hier: Objekt erzeugen und speichern (mit 2. Helper: ??)

Class SerializeTest

```
public static void main(String[] args) {  
    SerTestObj obj = new SerTestObj();  
    obj.setText("Das ist ein Haus von Nikigraus!");  
  
    // Datei einlesen  
    try(FileOutputStream fos = new FileOutputStream("test.ser");  
        ObjectOutputStream oos = new ObjectOutputStream(fos)) {  
  
        oos.writeObject(obj);  
    }
```

Mit Marker Interface (), macht das Objekt jetzt serialisierbar.

```
class SerTestObj implements Serializable {  
    private String text;  
  
    public String getText() {  
        return text;  
    }  
  
    public void setText(String text) {  
        this.text = text;  
    }  
}
```

Serialized Gedöns:

```
1 1
```

Deserialisierung

```
// Datei einlesen  
try(FileInputStream fis = new FileInputStream("test.ser");  
    ObjectInputStream ois = new ObjectInputStream(fis)) {  
  
    Object o = ois.readObject();  
  
    }  
catch(FileNotFoundException ex) {
```

Casten auf das, was es ursprünglich gewesen ist.

```
// Datei einlesen  
try(FileInputStream fis = new FileInputStream("test.ser");  
    ObjectInputStream ois = new ObjectInputStream(fis)) {  
  
    SerTestObj fromFile = (SerTestObj) ois.readObject();  
  
    }  
catch(FileNotFoundException ex) {
```

ClassNotFoundException noch behandeln:

```
    SerTestObj fromFile = (SerTestObj) ois.readObject();  
  
    }  
catch(ClassNotFoundException ex) {  
    System.out.println("Datei nicht gefunden");  
}
```

... Except. kommt nicht vom cast

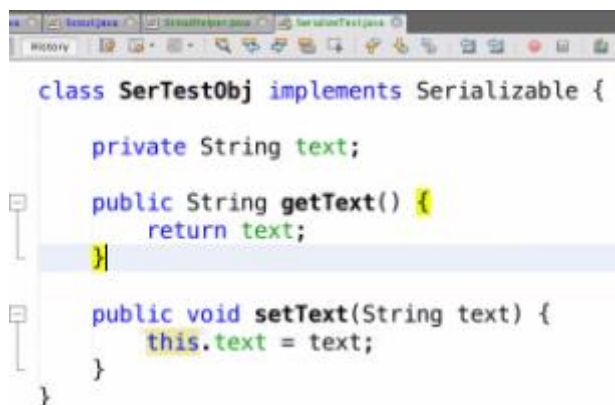
Um Sicherzustellen das objekt gelesen wurde

```
try(FileOutputStream fos = new FileOutputStream("test.ser");
    ObjectOutputStream oos = new ObjectOutputStream(fos)) {
    oos.writeObject(obj);
}

public static void main(String[] args) {
    SerTestObj obj = new SerTestObj();
    obj.setText("Das ist ein Haus von Nikigraus!");

    // Datei einlesen
    try(FileInputStream fis = new FileInputStream("test.ser");
        ObjectInputStream ois = new ObjectInputStream(fis)) {
        SerTestObj fromFile = (SerTestObj) ois.readObject();
        System.out.println(fromFile.getText());
    }
}
```

SerializeTest



```
class SerTestObj implements Serializable {
    private String text;

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }
}
```

ArrayList aufbauen

```
public class SerializeTest {

    public static void main(String[] args) {

        List<SerTestObj> liste = new ArrayList<>();
        SerTestObj o1 = new SerTestObj();
        o1.setText("Das ist ein Haus von Nikigraus!");
        liste.add(o1);

        SerTestObj o2 = new SerTestObj();
        o2.setText("Das ist ein anderer Text...");
        liste.add(o2);
    }
}
```

Liste übergeben zum Serialisieren:

Damit wird die Liste abgespeichert als Objektbaum:


```
// Datei einlesen
try(FileOutputStream fos = new FileOutputStream("test.ser");
    ObjectOutputStream oos = new ObjectOutputStream(fos)) {

    oos.writeObject(liste);

}
catch(FileNotFoundException ex) {
    System.out.println("Datei nicht gefunden");
}
catch(IOException ex) {
    System.out.println("Daten konnten nicht geschrieben werden");
    System.out.println(ex.getMessage());
}

run:
Daten konnten nicht geschrieben werden
de.gfn.oqa.persistance.SerTestObj
BUILD SUCCESSFUL (total time: 0 seconds)
```

➔ ... implements Serializable wieder herstellen.

Jetzt Schreiben deaktiv und Lesen aktiv setzen

```
// Datei einlesen
try(FileInputStream fis = new FileInputStream("test.ser");
    ObjectInputStream ois = new ObjectInputStream(fis)) {

    SerTestObj fromFile = (SerTestObj) ois.readObject();
    System.out.println(fromFile.getText());

}
catch(ClassNotFoundException ex) {
    System.out.println("Problem mit Daten");
}
catch(FileNotFoundException ex) {
    System.out.println("Datei nicht gefunden");
}
catch(IOException ex) {
    System.out.println("Daten konnten nicht gelesen werden");
}

// Datei einlesen
try(FileInputStream fis = new FileInputStream("test.ser");
    ObjectInputStream ois = new ObjectInputStream(fis)) {

    ArrayList<SerTestObj> fromFile = (ArrayList<SerTestObj>) ois.readObject();
    for(SerTestObj o : fromFile) {
        System.out.println(o.getText());
    }

}

run:
Das ist ein Haus von Nikigraus!
Das ist ein anderer Text...
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jetzt:

Serialized UID

```
// Datei einlesen
try(FileInputStream fis = new FileInputStream("test.ser");
    ObjectInputStream ois = new ObjectInputStream(fis)) {

    ArrayList<SerTestObj> fromFile = (ArrayList<SerTestObj>) ois.readObject()
    for(SerTestObj o : fromFile) {
        System.out.println(o.getText());
        System.out.println(o.getZahl());
    }
}
```

- ➔ Error-Ausgabe wegen getZahl()
- ➔ serialVersionUID hinzufügen zur Weiterverarbeitung (Wiederherstellung etc)

```
class SerTestObj implements Serializable {

    static final long serialVersionUID = 1L;

    private String text;

    private String zahl;

    public String getText() {
        return text;
    }
}
```