Universität Augsburg
Institut für Informatik

Deckblatt: Übung zur Vorlesung Informatik 2 SS 2018
Fakultät für Angewandte Informatik
Lehrprofessur für Informatik
Prof. Dr. Lorenz, Marius Brendle, Johannes Metzger

Hinweis: Es sind alle Felder auszufüllen! Abgabe der Übungsblätter immer **mittwochs** (Ausnahme wenn Feiertag: donnerstags) bis **spätestens 12:00 Uhr** in die entsprechend gekennzeichneten Briefkästen der Veranstaltung im Erdgeschoss des Instituts für Informatik (Gebäude N). Zuwiderhandlung wird mit Strafe geahndet! (Punktabzug)

Übungsblatt	

(hier die Nummer des bearbeiteten **Übungsblatts eintragen**)

	Übung 01 (1057 N) Montag 08:15 - 09:45 Uhr (Isabell Rücker)
	Übung 02 (1056 N) Montag 14:00 - 15:30 Uhr (Henning Cui)
	Übung 03 (1057 N) Montag 15:45 - 17:15 Uhr (Josef Kircher)
	Übung 04 (1054 N) Montag 17:30 - 19:00 Uhr (Mosaab Slimani)
	Übung 05 (1057 N) Montag 17:30 - 19:00 Uhr (David Hacker)
	Übung 06 (1055 N) Dienstag 12:15 - 13:45 Uhr (André Schweiger)
X	Übung 07 (1054 N) Dienstag 17:30 - 19:00 Uhr (Benjamin Sertolli)
	Übung 08 (1057 N) Dienstag 17:30 - 19:00 Uhr (Dat Le Thanh)
	Übung 09 (1054 N) Mittwoch 08:15 - 09:45 Uhr (Erik Pallas)
	Übung 10 (1055 N) Mittwoch 08:15 - 09:45 Uhr (Moritz Feldmann)
	Übung 11 (1054 N) Mittwoch 10:00 - 11:30 Uhr (Denise Böhm)
	Übung 12 (1056 N) Donnerstag 08:15 - 09:45 Uhr (Florian Magg)
	Übung 13 (1054 N) Donnerstag 15:45 - 17:15 Uhr (Marvin Drexelius)
	Übung 14 (1054 N) Donnerstag 17:30 - 19:00 Uhr (Patrick Eckert)
	Übung 15 (1057 N) Donnerstag 17:30 - 19:00 Uhr (Alexander Szöke)
	Übung 16 (1057 N) Freitag 08:15 - 09:45 Uhr (Philipp Braml)
	Übung 17 (1054 N) Freitag 10:00 - 11:30 Uhr (Elisabeth Korndörfer)
	Übung 18 (1054 N) Freitag 12:15 - 13:45 Uhr (Philipp Häusele)
	Übung 19 (1056 N) Freitag 12:15 - 13:45 Uhr (Maximilian Demmler)
	Übung 20 (1054 N) Freitag 14:00 - 15:30 Uhr (Florian Straßer)

(hier die eingeteilte **Übungsgruppe ankreuzen**)

Teamnummer	6

(hier die Nummer des eingeteilten **Teams eintragen**)

Tarik Selimovic
Anton Lydike
Dominic Cesnak

(hier die **Vor- und Nachnamen** aller Teammitglieder **eintragen**)

Aufgabe		
Aufgabe		
Aufgabe		
Aufgabe		
Gesamt		(vom Tutor auszufüllen)

Übungsblatt 8

29_30)

data)

EmployeeContainer)

```
package blatt08.data;

import blatt08.store.DatenhaltungDB;
import blatt08.store.Employee;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Observable;
import java.util.zip.DataFormatException;

public class EmployeeContainer extends Observable implements Iterable<Employee> {

    private static EmployeeContainer unique = null;
    private ArrayList<Employee> allEmployees;
    private int count = 0;

    private DatenhaltungDB store = null;

    private EmployeeContainer(String user, String password) throws ClassNotFoundException, SQLException {
        allEmployees = new ArrayList<Employee>();
        connect(user, password);
    }

    public void load() throws SQLException {
        store.load(this);
    }

    private void connect(String user, String password) throws ClassNotFoundException, SQLException {
        store = DatenhaltungDB.instance(user, password);
    }

    public static EmployeeContainer instance(String user, String password) throws ClassNotFoundException,
            SQLException {
        if (unique == null)
            unique = new EmployeeContainer(user, password);
        return unique;
    }


    public void linkEmployeeFromDB(Employee a) throws DataFormatException {
        if (this.allEmployees.contains(a))
            throw new DataFormatException("Angestellter schon vorhanden: "
                    + a.getNumber());
        this.allEmployees.add(a);
        setChanged();
        notifyObservers();
        ++count;
    }

    public Iterator<Employee> iterator() {
        return this.allEmployees.iterator();
    }

    public Employee searchEmployee(int number) {
        for (Employee a : allEmployees) {
            if (a.getNumber() == number)
                return a;
        }
        return null;
    }
```

```
    public void close() {
        if (store != null) {
            store.close();
        }
    }
}
```

TemporaryEmployee)

```
package blatt08.data;

import blatt08.store.Employee;

import java.time.LocalDate;
import java.util.zip.DataFormatException;

public class TemporaryEmployee extends Employee {

    private LocalDate endOfContract;

    public TemporaryEmployee(int number, String name, LocalDate beginOfContract,
                             LocalDate endOfContract) throws DataFormatException {
        super(number, name, beginOfContract);
        setEndOfContract(endOfContract);
    }

    private boolean checkEndOfContract(LocalDate endOfContract) {
        return this.getBeginOfContract().isBefore(endOfContract);
    }

    private void setEndOfContract(LocalDate endOfContract)
            throws DataFormatException {
        if (!checkEndOfContract(endOfContract))
            throw new DataFormatException(
                    "Objekt Zeitangestellter: Ungueltiges Vertragsende");
        this.endOfContract = endOfContract;
    }

    public LocalDate getEndOfContract() {
        return endOfContract;
    }

    @Override
    public String toString() {
        return super.toString() + ", bis " + getEndOfContract();
    }

}
```

gui)

ListEmployee)

```
package blatt08.gui;

import blatt08.data.EmployeeContainer;
import blatt08.store.Employee;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.sql.SQLException;
import java.util.Observable;
import java.util.Observer;

public class ListEmployee extends Dialog implements ActionListener, Observer {

    private EmployeeContainer container;
    private List allEmployeesList;

    public ListEmployee(Window f, EmployeeContainer container) throws SQLException {

        super(f, "Alle Angestellten anzeigen", false);

        this.setLayout(new GridLayout(0, 1));
        this.container = container;

        Panel unten = new Panel();
        add(unten);
        unten.setLayout(new BorderLayout());

        Label alleAngestelltenLabel = new Label("Alle Angestellten: ");
        unten.add(alleAngestelltenLabel, BorderLayout.NORTH);

        allEmployeesList = new List();
        unten.add(allEmployeesList, BorderLayout.CENTER);
        allEmployeesList.setEnabled(true);

        Panel s = new Panel();
        unten.add(s, BorderLayout.SOUTH);

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });

        setLocation(f.getLocation().x + 200, f.getLocation().y + 200);


        container.addObserver(this);
        update(container, null);
        container.load();

        pack();
        setVisible(true);
    }

    public void update(Observable o, Object arg) {
        allEmployeesList.removeAll();

        for (Employee a : container) {
            allEmployeesList.add(a.toString());
        }
    }
```

```
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("Angestellten löschen")) {
            // onDelete();
        } else if (e.getActionCommand().equals("Abbrechen")) {
            onCancel();
        }
    }

    private void onCancel() {
        dispose();
    }

}
```

LoginModal)

```
package blatt08.gui;

import blatt08.store.Credentials;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class LoginModal extends Dialog implements ActionListener {
    private static final long serialVersionUID = 1L;
    private TextField tf_user = new TextField();
    private TextField tf_password = new TextField();
    private Button btn_login = new Button("login");
    private Button btn_cancel = new Button("cancel");

    private Credentials credentials = new Credentials();


    public LoginModal(Frame f) {
        super(f, "Enter Login credentials", true);

        FlowLayout fl = new FlowLayout();

        btn_login.addActionListener(this);
        btn_cancel.addActionListener(this);

        tf_user.setPreferredSize(new Dimension(150, 16));
        tf_user.setText("student");
        tf_password.setText("inFormatik2");
        this.add(tf_user);

        tf_password.setPreferredSize(new Dimension(150, 16));
        this.add(tf_password);
        this.add(btn_login);
        this.add(btn_cancel);


        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                dispose();
            }
        });

        this.setLayout(fl);
        this.pack();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource().equals(this.btn_login)) {
            saveCredentials();
            this.setVisible(false);
            this.dispose();
        } else if (e.getSource().equals(this.btn_cancel)) {
            System.out.println("cancel clicked");
            this.setVisible(false);
            this.dispose();
        }
    }

    private void saveCredentials() {
        credentials.setUsername(tf_user.getText());
        credentials.setPassword(tf_password.getText());
    }
```

```
    public Credentials showDialog() {
        this.setVisible(true);
        return this.credentials;
    }
}
```

Window)

```
package blatt08.gui;

import blatt08.data.EmployeeContainer;
import blatt08.store.Credentials;

import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;
import java.util.*;

public class Window extends Frame implements ActionListener {
    private static final long serialVersionUID = 1L;
    private ArrayList<Double> history = new ArrayList<Double>();
    private Button btn_connect = new Button("connect");
    private Button btn_load = new Button("load");
    private LoginModal loginModal;

    private EmployeeContainer container;
    private Label anzeige;
    private Credentials credentials;
    private ListEmployee listEmployee;


    public Window() {
        super("Aufgabe29+30");

        this.loginModal = new LoginModal(this);
        FlowLayout fl = new FlowLayout();

        btn_connect.addActionListener(this);
        this.add(btn_connect);

        btn_load.setEnabled(false);
        btn_load.addActionListener(this);
        this.add(btn_load);

        anzeige = new Label("Programmfenster");
        this.add(anzeige, BorderLayout.CENTER);
        anzeige.setText("Bitte logge dich zuerst ein.");

        this.setLayout(fl);

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                container.close();
                dispose();
            }
        });

        this.pack();
        this.setVisible(true);
    }

    public static void main(String[] args) {
        new Window();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource().equals(this.btn_connect)) {
            showLoginModal();
        } else if (e.getSource().equals(this.btn_load)) {
            loadData();
        }
    }
```

```
    private void showLoginModal() {
        credentials = this.loginModal.showDialog();
        if (credentials != null) {
            try {
                container = EmployeeContainer.instance(credentials.getUsername(),
credentials.getPassword());
                anzeige.setText("Login erfolgreich.");
            } catch (SQLException e) {
                //m.setEnabled(false);
                anzeige.setText("Laden aus Datenbank fehlgeschlagen" + e.getMessage());
            } catch (ClassNotFoundException e) {
                anzeige.setText("Datenbanktreiber nicht gefunden: " + e.getMessage());
                // m.setEnabled(false);
            }

            btn_load.setEnabled(true);
        }
    }

    private void loadData() {
        try {
            anzeige.setText("Daten werden geladen...");
            listEmployee = new ListEmployee(this, container);
            anzeige.setText("Daten erfolgreich geladen.");
        } catch (SQLException e) {
            listEmployee.setVisible(false);
            anzeige.setText("Laden aus Datenbank fehlgeschlagen" + e.getMessage());
        }
    }
}
```

store)

Credentials)

```
package blatt08.store;

public class Credentials {

    private String username;
    private String password;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

DatenhaltungDB)

```
package blatt08.store;


import blatt08.data.EmployeeContainer;
import blatt08.data.TemporaryEmployee;

import java.sql.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.zip.DataFormatException;

public class DatenhaltungDB {

    private static final String driver = "com.mysql.cj.jdbc.Driver";
    private static final String url = "jdbc:mysql://educos-srv01.informatik.uni-
augsburg.de:3306/theDatabase?useSSL=false";
    // private static final String user = "student";
    //  private static final String password = "inFormatik2";
    private Connection con = null;
    private static DatenhaltungDB unique = null;
    final DateTimeFormatter DATE_FORMAT = DateTimeFormatter.ofPattern("dd.MM.yyyy");

    /*
     * Verbindung mit Datenbank herstellen
     */
    private DatenhaltungDB(String user, String password) throws ClassNotFoundException, SQLException {
        Class.forName(driver);
        con = DriverManager.getConnection(url, user, password);
    }

    /*
     * Zugriff auf Singleton-Objekt
     */
    public static DatenhaltungDB instance(String user, String password) throws ClassNotFoundException,
            SQLException {
        if (unique == null)
            unique = new DatenhaltungDB(user, password);
        return unique;
    }

    /*
     * Alle Angestelltendaten aus Tabelle laden
     */
    public void load(EmployeeContainer container) throws SQLException {
        Employee a = null;
        Statement abfrage = null;
        ResultSet employees = null;
        if (con == null)
            return;
        abfrage = con.createStatement();

        //SQL-Befehl erstellen und abschicken
        String befehl1 = "select * from Angestellter";
        employees = abfrage.executeQuery(befehl1);

        //Ergebnistabelle durchlaufen
        while (employees.next()) {
            try {
                //Fallunterscheidung, ob Zeitangestellter oder nicht
                if (employees.getBoolean(5) == false) {
                    LocalDate beginDate = LocalDate.parse(employees
                            .getString(3), DATE_FORMAT);
                    //Angestellten-Objekt erstellen
                    a = new Employee(employees.getInt(1),
                            employees.getString(2), beginDate);
                } else {
```

```
                    if (employees.getBoolean(5) == true) {
                        LocalDate beginDate = LocalDate.parse(employees
                                .getString(3), DATE_FORMAT);
                        LocalDate endDate = LocalDate.parse(employees
                                .getString(4), DATE_FORMAT);
                        //Zeitangestellten-Objekt erstellen
                        a = new TemporaryEmployee(employees.getInt(1),
                                employees.getString(2), beginDate, endDate);
                    }
                }
                //Erstelltes Objekt in Container aufnehmen
                container.linkEmployeeFromDB(a);
            } catch (DateTimeParseException e) {
                System.out.println("Angestellter " + employees.getInt(1)
                        + " kann nicht geladen werden: " + e.getMessage());
            } catch (DataFormatException e) {
                System.out.println("Angestellter " + employees.getInt(1)
                        + " kann nicht geladen werden: " + e.getMessage());
            }
        }
        abfrage.close();
    }

    /*
     * Verbindung schließen
     */
    public void close() {
        try {
            if (con != null)
                con.close();
        } catch (SQLException e) {
            System.out.println("Datenbankfehler beim Schließen");
        }
    }
}
```

Employee)

```
package blatt08.store;

import java.time.LocalDate;
import java.util.zip.DataFormatException;

public class Employee {

    private int number;
    private String name;
    private LocalDate beginOfContract;

    public Employee(int number, String name, LocalDate beginOfContract)
            throws DataFormatException {
        setNumber(number);
        setName(name);
        setBeginOfContract(beginOfContract);
    }

    private static boolean checkNumber(int number) {
        return true;
    }

    public void setNumber(int number) throws DataFormatException {
        if (!checkNumber(number))
            throw new DataFormatException("Nummer muss größer als 0 sein: "
                    + number);
        this.number = number;
    }

    public int getNumber() {
        return number;
    }

    private static boolean checkName(String name) {
        return true;
    }

    public void setName(String name) throws DataFormatException {
        if (!checkName(name))
            throw new DataFormatException(
                    "Objekt Angestellter: Ungueltiger Name");
        this.name = name;
    }

    private static boolean checkBeginOfContract(LocalDate beginOfContract) {
        return true;
    }

    private void setBeginOfContract(LocalDate beginOfContract)
            throws DataFormatException {
        if (!checkBeginOfContract(beginOfContract))
            throw new DataFormatException(
                    "Objekt Angestellter: Ungueltiger Vertragsbeginn");
        this.beginOfContract = beginOfContract;
    }

    public String getName() {
        return name;
    }

    public LocalDate getBeginOfContract() {
        return beginOfContract;
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (o == null)
            return false;
        if (!(o instanceof Employee))
            return false;
        Employee a = (Employee) o;
        return (a.getNumber() == this.getNumber());
    }

    @Override
    public String toString() {
        return "Angestellter " + getNumber() + ": " + getName() + ", ab "
                + getBeginOfContract();
    }
}
```

31)

a)

Bibliotheksverwaltung)

```
package blatt08.a31;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class Bibliotheksverwaltung extends Frame implements ActionListener{
    private Button close;

    public Bibliotheksverwaltung() {
        super("Aufgabe 31");
        close = new Button("Close");
        close.addActionListener(this);

        this.add(close, BorderLayout.CENTER);
        try {
            BuchContainer.instance();
        } catch (LoadSaveException e) {
            dispose();
        }

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                dispose();
            }
        });

        this.pack();
        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if( e.getSource().equals(close) ) {
            setVisible(false);
            dispose();
        }
    }
}
```

BuchContainer)

```
package blatt08.a31;

import blatt08.data.EmployeeContainer;

import java.util.Observable;

public class BuchContainer extends Observable {

    private static BuchContainer unique = null;

    public static BuchContainer instance() throws LoadSaveException {
        if (unique == null)
            unique = new BuchContainer();
        return unique;
    }
}
```

LoadSaveException)

```
package blatt08.a31;

public class LoadSaveException extends Exception {
}
```

User)

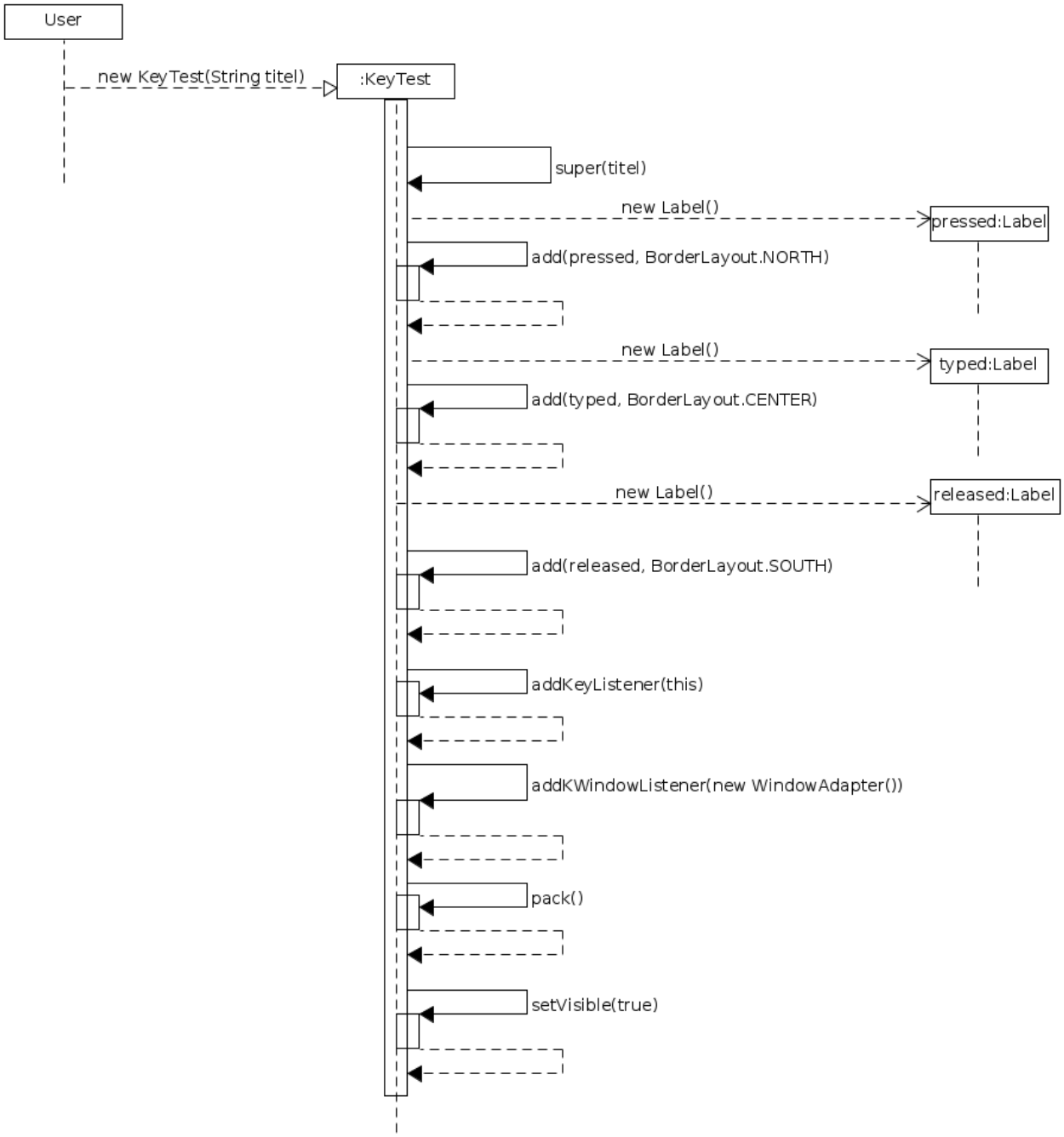
```
package blatt08.a31;

public class User {

    public static void main(String[] args) {
        new Bibliotheksverwaltung();
    }

}
```

b)

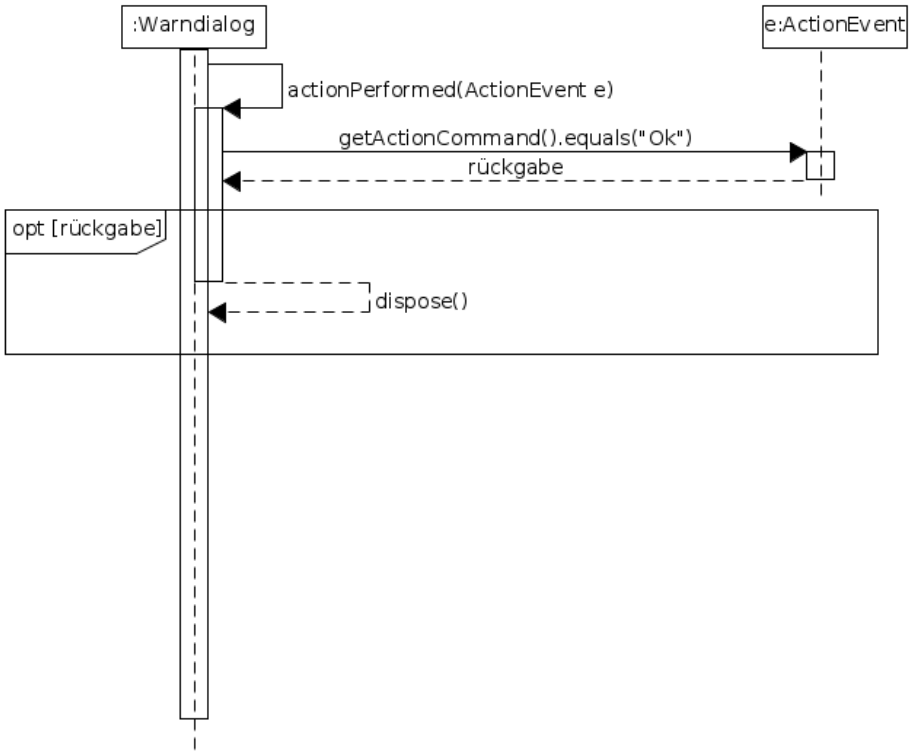
```
package übungsblatt8;

//Aufgabe 31b

import java.util.zip.DataFormatException;
import java.util.ArrayList;

public class BuchContainer {

public void linkBuch(Buch buch) throws DataFormatException {
  if (contains(buch)) {
    new DataFormatException("schon vorhanden");
  }else {
    this.add(buch);
  }
}
}
```

32)

a)


User
new KeyTest(String titel)
:KeyTest
super(titel)
new Label()
pressed:Label
add(pressed, BorderLayout.NORTH)
new Label()
typed:Label
add(typed, BorderLayout.CENTER)
new Label()
released:Label
add(released, BorderLayout.SOUTH)
addKeyListener(this)
addKWindowListener(new WindowAdapter())
pack()
setVisible(true)

b)


:Warndialog
e:ActionEvent
actionPerformed(ActionEvent e)
getActionCommand().equals("Ok")
rückgabe
opt [rückgabe]
dispose()