SWE4	Übung zu Softwareentwicklung mit modernen Plattformen 4	SS 2022, Übung 6

☐ Gruppe 1 (J. Heinzelreiter)

☒ Gruppe 2 (P. Kulczycki) Name: Angelos Angelis Aufwand [h]: 15

☐ Gruppe 3 (M. Hava) Peer-Review von: _____

Beispiel	Lösungsidee (max. 100%)	Implement. (max. 100%)	Testen (max. 100%)
1a (50 P)	100	100	100
1b (50 P)	100	100	100

Ausbaustufe 2: Pomagaju-Server

In der Übung 5 haben Sie bereits eine Java-Anwendung mit grafischer Benutzeroberfläche (auf JavaFX-Basis) für unsere Spendenplattform erstellt. Im bisherigen Ausbaustadium sollte es die Verwaltungsanwendung im Wesentlichen ermöglichen, Annahmestellen zu administrieren, den Bedarf an Hilfsgütern zu erfassen und Spendenankündigungen anzuzeigen. In der Spendenanwendung sollten Benutzer registriert, der Bedarf an Hilfsgütern aufgelistet und Spendenankündigungen erstellt werden können.

In dieser Ausbaustufe soll Ihr Programm zu einer Client-Server-Anwendung erweitert werden. Die Clients und der Server stellen eigenständige Programme dar, die über RMI miteinander kommunizieren.

Im Detail sollten Sie folgende Anforderungen umsetzen:

a) Implementieren Sie die Serverkomponente, welche die Daten der Anwendung verwaltet. Stellen Sie die Funktionalität der Komponente über eine RMI-Schnittstelle zur Verfügung. Zur Serverkomponente sollen sich parallel beliebig viele Clients verbinden können.

   Die Datenhaltung kann in dieser Ausbaustufe noch vollständig im Hauptspeicher der Serverkomponente erfolgen. In der nächsten Ausbaustufe ist die Anwendung um die dauerhafte Speicherung der Daten in einer relationalen Datenbank zu erweitern.

b) Integrieren Sie die Verwaltungs- und die Spendenanwendung in die Client-Server-Architektur. Beide Anwendungen sind über RMI mit der Serverkomponente verbunden und tauschen so mit dieser Daten aus. Beachten Sie, dass die beiden Clients ausschließlich für die Darstellung der Daten und die Benutzerinteraktion zuständig sind. Alle darüber hinausgehenden Aufgaben sind an die Serverkomponente zu delegieren. Stellen Sie sicher, dass die GUI bedienbar bleibt, auch wenn z. B. gerade auf eine Antwort vom Server gewartet wird.

Achten Sie darauf, dass Sie Ihre Anwendung nach den Grundprinzipien des objektorientierten Designs entwerfen, jede Klasse für einen Aspekt der Anwendung verantwortlich ist und die Methoden nicht zu umfangreich sind. Erstellen Sie für Ihre Anwendung sowohl Unit- als auch Integrations-Tests.

Lösungsidee:

Server:

Als Kommunikationsschicht zwischen Client und Server wird RMI genutzt. Die Schnittstelle zwischen Client und Server werden Service Interfaces angelegt, welche eine konkrete Implementierung am Server erhalten. Weiters werde ich meine bisherige Lösung anhand vom Feedback erweitern und Fakerepos implementieren.

Client:

Am Client muss mulitthreading eingeführt werden. Um die Daten, welche angezeigt werden, zentral zu verwalten, wird ein DataService eingeführt, dieser hat Methoden um Daten zurückzugeben, bzw. um sie zu aktualisieren. Zum Aktualisieren der Daten ist der RefreshService zuständig, dieser ist ein Thread, welcher periodisch die Daten aktualisiert. Um so wenig Multithreading wie möglich in den JavaFXControllern zu haben, werden Client-Services eingeführt, welche sich um die das Hinzufügen/Ändern/Löschen von Daten mittels Threads kümmern

Quellcode: Code von UE5 und Interfaces nicht im pdf

Server:

```
package swe4.server;

import javafx.embed.swing.JFXPanel;
import swe4.server.config.ServiceConfig;
import swe4.server.services.*;

import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

public class Server {
    private static BenutzerService benutzerService = new
BenutzerServiceImpl();
    private static AnnahmestelleService annahmestelleService = new
AnnahmestelleServiceImpl();
    private static BedarfService bedarfService = new BedarfServiceImpl();
    private static SpendenankündigungService spendenankündigungService = new
SpendenankündigungServiceImpl();

    public static void main(String... args) {
        //TODO: Fix Edits
        new InitFakeDataService().load();

        try {
            BenutzerService benutzerServiceStub = (BenutzerService)
UnicastRemoteObject
                    .exportObject(benutzerService, 0);

            AnnahmestelleService annahmestelleServiceStub =
(AnnahmestelleService) UnicastRemoteObject
                    .exportObject(annahmestelleService, 0);

            BedarfService bedarfServiceStub = (BedarfService)
UnicastRemoteObject
```

```
                    .exportObject(bedarfService, 0);

            SpendenankündigungService spendenankündigungServiceStub =
(SpendenankündigungService) UnicastRemoteObject
                    .exportObject(spendenankündigungService, 0);

            Registry registry =
LocateRegistry.createRegistry(ServiceConfig.port());

            registry.rebind(ServiceConfig.benutzerServiceName(),
benutzerServiceStub);
            registry.rebind(ServiceConfig.annahmestelleServiceName(),
annahmestelleServiceStub);
            registry.rebind(ServiceConfig.bedarfServiceName(),
bedarfServiceStub);
            registry.rebind(ServiceConfig.spendenankündigungServiceName(),
spendenankündigungServiceStub);

            System.out.println("Server Running...");
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

Serverconfig:

```
package swe4.server.config;

public class ServiceConfig {
    public static String benutzerServiceName() {
        return "BenutzerService";
    }

    public static String annahmestelleServiceName() {
        return "AnnahmestelleService";
    }

    public static String bedarfServiceName() {
        return "BedarfService";
    }

    public static String spendenankündigungServiceName() {
        return "SpendenankündigungService";
    }
    public static int port(){
        return 1099;
    }
}
```

AnnahmestelleServiceImpl:

```
package swe4.server.services;

import javafx.collections.ObservableList;
import swe4.server.repositories.AnnahmestelleRepository;
import swe4.server.repositories.RepositoryFactory;
import swe4.ui.Annahmestelle;
```

```
import java.rmi.RemoteException;
import java.util.List;

public class AnnahmestelleServiceImpl implements AnnahmestelleService {
    private final AnnahmestelleRepository annahmestelleRepository =
RepositoryFactory.annahmestelleRepositoryInstance();

    @Override
    public List<Annahmestelle> findAllAnnahmestellen() throws RemoteException
{
        return annahmestelleRepository.findAllAnnahmestellen();
    }

    @Override
    public Annahmestelle findByName(String name) throws RemoteException {
        for (Annahmestelle
a:annahmestelleRepository.findAllAnnahmestellen()){
            if (a.getName()== name){
                return a;
            }
        }
        return null;
    }

    @Override
    public void insertAnnahmestelle(Annahmestelle annahmestelle) throws
RemoteException {
        annahmestelleRepository.insertAnnahmestelle(annahmestelle);
    }

    @Override
    public void deleteAnnahmestelle(Annahmestelle annahmestelle) throws
RemoteException {
        annahmestelleRepository.deleteAnnahmestelle(annahmestelle);
    }

    @Override
    public void updateAnnahmestelle(Annahmestelle annahmestelle) throws
RemoteException {
        annahmestelleRepository.updateAnnahmestelle(annahmestelle);
    }
}
```

BedarfServiceImpl:

```
package swe4.server.services;

import javafx.collections.ObservableList;
import swe4.server.repositories.BedarfRepository;
import swe4.server.repositories.RepositoryFactory;
import swe4.ui.Hilfsgüter;

import java.util.List;

public class BedarfServiceImpl implements BedarfService {
    private final BedarfRepository bedarfRepository =
RepositoryFactory.BedarfRepositoryInstance();
```

```
    @Override
    public List<Hilfsgüter> findAllBedarf() {
        return bedarfRepository.findAllBedarf();
    }

    @Override
    public void insertBedarf(Hilfsgüter bedarf) {
        bedarfRepository.insertBedarf(bedarf);
    }

    @Override
    public void deleteBedarf(Hilfsgüter bedarf) {
        bedarfRepository.deleteBedarf(bedarf);
    }
}
```

BenutzerServiceImpl:

```
package swe4.server.services;

import swe4.server.repositories.RepositoryFactory;
import swe4.server.repositories.UserRepository;

import java.util.Map;

public class BenutzerServiceImpl implements BenutzerService {
    private final UserRepository userRepository =
RepositoryFactory.userRepositoryInstance();
    @Override
    public Map<String, String> findAllUsers() {
        return userRepository.findAllUsers();
    }

    @Override
    public String findUserByUsername(String username) {
        return userRepository.findUserByUsername(username);
    }

    @Override
    public void insertUser(String username, String password) {
        userRepository.insertUser(username,password);
    }

    @Override
    public void deleteUser(String user) {
        userRepository.deleteUser(user);
    }

    @Override
    public boolean containsKey(String key) {
        return userRepository.findAllUsers().containsKey(key);
    }

    @Override
    public boolean containsValue(String val) {
        return userRepository.findAllUsers().containsValue(val);
    }
}
```

InitFakeDataService:

```
package swe4.server.services;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import swe4.server.repositories.AnnahmestelleRepository;
import swe4.server.repositories.BedarfRepository;
import swe4.server.repositories.RepositoryFactory;
import swe4.server.repositories.SpendenankündigungRepository;
import swe4.ui.Annahmestelle;
import swe4.ui.Hilfsgüter;

import java.time.LocalDate;

public class InitFakeDataService {
    private final BenutzerServiceImpl benutzerService = new
BenutzerServiceImpl();
    private final AnnahmestelleRepository annahmestelleRepository =
RepositoryFactory.annahmestelleRepositoryInstance();
    private final BedarfRepository bedarfRepository =
RepositoryFactory.BedarfRepositoryInstance();
    private final SpendenankündigungRepository spendenankündigungRepository =
RepositoryFactory.spendenankündigungRepositoryInstance();

    public void load(){
        //user
        benutzerService.insertUser("Rickroll","123");

        //Annahmestellen
        annahmestelleRepository.insertAnnahmestelle(new
Annahmestelle("A1","OÖ","Softwarepark 69","Kiev","Aktiv"));
        annahmestelleRepository.insertAnnahmestelle(new
Annahmestelle("A2","OÖ","Softwarepark 69","Kiev","Aktiv"));

        //Bedarf
        ObservableList<String> kategorien =
FXCollections.observableArrayList();
        kategorien.add("1");
        kategorien.add("2");
        kategorien.add("3");
        ObservableList<String> bedarf = FXCollections.observableArrayList();
        bedarf.add("niedrig");
        bedarf.add("mittel");
        bedarf.add("hoch");
        bedarfRepository.insertBedarf(new Hilfsgüter("100% Wolle", "T
SHirt","Sehr gut","10","1","niedrig"));
        bedarfRepository.insertBedarf(new Hilfsgüter("100% Wolle", "T
SHirt","gut","5","1","niedrig"));
        bedarfRepository.insertBedarf(new Hilfsgüter("100% Wolle", "T
SHirt","Schlecht","10","1","niedrig"));
        bedarfRepository.insertBedarf(new Hilfsgüter("100% Wolle", "T
SHirt","Sehr gut","10","1","niedrig"));

        //Spendenankündigung
        //spendenankündigungRepository.insertSpendenankündigung(new
Hilfsgüter("Random Token","100% Wolle", "Hose","Sehr gut","10",
```

```
LocalDate.now(), "NÖ"));
        spendenankündigungRepository.insertSpendenankündigung(new
Hilfsgüter("Random Token","100% Wolle", "T SHirt","gut","5",
LocalDate.of(2022,5,2), "NÖ"));
        spendenankündigungRepository.insertSpendenankündigung(new
Hilfsgüter("Random Token","100% Wolle", "T SHirt","Schlecht","10",
LocalDate.of(2022,5,2), "NÖ"));
        spendenankündigungRepository.insertSpendenankündigung(new
Hilfsgüter("Random Token","100% Wolle", "T SHirt","Sehr gut","10",
LocalDate.of(2022,5,2), "NÖ"));
    }

}
```

SpendenankündigungServiceImpl:

```
package swe4.server.services;

import javafx.collections.ObservableList;
import swe4.server.repositories.RepositoryFactory;
import swe4.server.repositories.SpendenankündigungRepository;
import swe4.ui.Hilfsgüter;

import java.util.List;

public class SpendenankündigungServiceImpl implements
SpendenankündigungService {
    private final SpendenankündigungRepository spendenankündigungRepository =
RepositoryFactory.spendenankündigungRepositoryInstance();
    @Override
    public List<Hilfsgüter> findAllSpendenankündigung() {
        return spendenankündigungRepository.findAllSpendenankündigung();
    }

    @Override
    public void insertSpendenankündigung(Hilfsgüter spendenankündigung) {

spendenankündigungRepository.insertSpendenankündigung(spendenankündigung);
    }

    @Override
    public void deleteSpendenankündigung(Hilfsgüter spendenankündigung) {

spendenankündigungRepository.deleteSpendenankündigung(spendenankündigung);
    }
}
```

FakeannahmestelleRepository:

```
package swe4.server.repositories;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import swe4.ui.Annahmestelle;

import java.util.ArrayList;
import java.util.List;

public class FakeAnnahmestelleRepository implements AnnahmestelleRepository{
    public static List<Annahmestelle> annahmestellen = new ArrayList<>();
```

```
    @Override
    public List<Annahmestelle> findAllAnnahmestellen() {
        return annahmestellen;
    }

    @Override
    public Annahmestelle findByName(String name) {
        for (Annahmestelle a:annahmestellen){
            if (a.getName()==name){
                return a;
            }
        }
        return null;
    }

    @Override
    public void insertAnnahmestelle(Annahmestelle annahmestelle) {
        annahmestellen.add(annahmestelle);
    }

    @Override
    public void deleteAnnahmestelle(Annahmestelle annahmestelle) {
        annahmestellen.remove(annahmestelle);
    }

    @Override
    public void updateAnnahmestelle(Annahmestelle annahmestelle) {
        int index = annahmestellen.indexOf(annahmestelle);
        if (index < 0) return;
        annahmestellen.set(index,annahmestelle);
    }
}
```

FakeBedarfRepository:

```
package swe4.server.repositories;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import swe4.ui.Hilfsgüter;

import java.util.ArrayList;
import java.util.List;

public class FakeBedarfRepository implements BedarfRepository{
    public static List<Hilfsgüter> bedarfs = new ArrayList<>();
    @Override
    public List<Hilfsgüter> findAllBedarf() {
        return bedarfs;
    }

    @Override
    public void insertBedarf(Hilfsgüter bedarf) {
        bedarfs.add(bedarf);
    }

    @Override
```

```
    public void deleteBedarf(Hilfsgüter bedarf) {
        bedarfs.remove(bedarf);
    }
}
```

FakeSpendenankündigungRepository:

```
package swe4.server.repositories;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import swe4.ui.Annahmestelle;
import swe4.ui.Hilfsgüter;

import java.util.ArrayList;
import java.util.List;

public class FakeSpendenankündigungRepository implements
SpendenankündigungRepository{
    public static List<Hilfsgüter> spendenankündigungen = new ArrayList<>();
    @Override
    public List<Hilfsgüter> findAllSpendenankündigung() {
        return spendenankündigungen;
    }

    @Override
    public void insertSpendenankündigung(Hilfsgüter spendenankündigung) {
        spendenankündigungen.add(spendenankündigung);
    }

    @Override
    public void deleteSpendenankündigung(Hilfsgüter spendenankündigung) {
        spendenankündigungen.remove(spendenankündigung);
    }
}
```

FakeUserRepository:

```
package swe4.server.repositories;

import java.util.Map;
import java.util.TreeMap;

public class FakeUserRepository implements UserRepository{
    public static Map<String,String> users = new TreeMap<>();

    @Override
    public Map<String, String> findAllUsers() {
        return users;
    }

    @Override
    public String findUserByUsername(String username) {
        return users.get(username);
    }

    @Override
    public void insertUser(String username, String password) {
        users.put(username,password);
    }
```

```
    @Override
    public void deleteUser(String user) {
        users.remove(user);
    }
}
```

RepositoryFactory:

```
package swe4.server.repositories;

public class RepositoryFactory {
    private static UserRepository userRepository = null;
    private static AnnahmestelleRepository annahmestelleRepository = null;
    private static BedarfRepository bedarfRepository = null;
    private static SpendenankündigungRepository spendenankündigungRepository
= null;


    private RepositoryFactory() {
        throw new AssertionError("No RepositoryFactory instances for you!");
    }

    public static UserRepository userRepositoryInstance() {
        if(userRepository == null) {
            userRepository = new FakeUserRepository();
        }
        return userRepository;
    }

    public static AnnahmestelleRepository annahmestelleRepositoryInstance() {
        if(annahmestelleRepository == null) {
            annahmestelleRepository = new FakeAnnahmestelleRepository();
        }
        return annahmestelleRepository;
    }

    public static BedarfRepository BedarfRepositoryInstance() {
        if(bedarfRepository == null) {
            bedarfRepository = new FakeBedarfRepository();
        }
        return bedarfRepository;
    }

    public static SpendenankündigungRepository
spendenankündigungRepositoryInstance() {
        if(spendenankündigungRepository == null) {
            spendenankündigungRepository = new
FakeSpendenankündigungRepository();
        }
        return spendenankündigungRepository;
    }
}
```

DataService:

```
package swe4.client.services;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
```

```
import swe4.server.services.AnnahmestelleService;
import swe4.server.services.BedarfService;
import swe4.server.services.BenutzerService;
import swe4.server.services.SpendenankündigungService;
import swe4.ui.Annahmestelle;
import swe4.ui.Hilfsgüter;

import java.rmi.RemoteException;

public class DataService {
    private final AnnahmestelleService annahmestelleService =
ServiceFactory.annahmestelleServiceInstance();
    private final BenutzerService benutzerService =
ServiceFactory.benutzerServiceInstance();
    private final BedarfService bedarfService =
ServiceFactory.bedarfServiceInstance();
    private final SpendenankündigungService spendenankündigungService =
ServiceFactory.spendenankündigungServiceInstance();

    private final ObservableList<Annahmestelle> annahmestellen =
FXCollections.observableArrayList();
    private final ObservableList<Hilfsgüter> bedarf =
FXCollections.observableArrayList();
    private final ObservableList<Hilfsgüter> spendenankündigungen =
FXCollections.observableArrayList();

    public void refresh() throws RemoteException {
        annahmestellen.setAll(annahmestelleService.findAllAnnahmestellen());
        bedarf.setAll((bedarfService.findAllBedarf()));

spendenankündigungen.setAll(spendenankündigungService.findAllSpendenankündigu
ng());
    }

    public ObservableList<Annahmestelle> getAnnahmestellen(){
        return annahmestellen;
    }

    public ObservableList<Hilfsgüter> getBedarf(){
        return bedarf;
    }

    public ObservableList<Hilfsgüter> getSpendenankündigungen(){
        return spendenankündigungen;
    }
}
```

RefreshService:

```
package swe4.client.services;

import java.rmi.RemoteException;

public class RefreshService extends Thread {

    private static final int WAIT_TIME = 1000;
    public static final int WAIT_TIME_MULT = 10;
    private final DataService dataService =
```

```
ServiceFactory.dataServiceInstance();
    private boolean stop = false;
    private int ticker = 0;

    @Override
    public synchronized void run() {
        while(!stop) {
            try {
                wait(WAIT_TIME);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            ticker++;
            if(ticker == WAIT_TIME_MULT) {
                try {
                    dataService.refresh();
                } catch (RemoteException e) {
                    e.printStackTrace();
                }
                System.out.println("Refreshing!");
            }
            ticker = ticker % WAIT_TIME_MULT;
        }
    }

    public synchronized void stopRefreshing() {
        stop = true;
    }
}
```

AnnahmestelleClientService:

```
package swe4.client.services.client;

import swe4.client.services.DataService;
import swe4.client.services.ServiceFactory;
import swe4.server.services.AnnahmestelleService;
import swe4.ui.Annahmestelle;

import java.rmi.RemoteException;

public class AnnahmestelleClientService {
    private final DataService dataService =
ServiceFactory.dataServiceInstance();
    private final AnnahmestelleService annahmestelleService =
ServiceFactory.annahmestelleServiceInstance();

    public void insertAnnahmestelle(Annahmestelle annahmestelle){
        new Thread(() -> {
            try {
                annahmestelleService.insertAnnahmestelle(annahmestelle);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            try {
                dataService.refresh();
            } catch (RemoteException e) {
                e.printStackTrace();
```

```
            }
        }).start();
    }

    public void deleteAnnahmestelle(Annahmestelle annahmestelle){
        new Thread(() -> {
            try {
                annahmestelleService.deleteAnnahmestelle(annahmestelle);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            try {
                dataService.refresh();
            } catch (RemoteException e) {
                e.printStackTrace();
            }
        }).start();
    }
    public void updateAnnahmestelle(Annahmestelle annahmestelle){
        new Thread(() -> {
            try {
                annahmestelleService.updateAnnahmestelle(annahmestelle);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            try {
                dataService.refresh();
            } catch (RemoteException e) {
                e.printStackTrace();
            }
        }).start();
    }
}
```

BenutzerClientService:

```
package swe4.client.services.client;

import swe4.client.services.DataService;
import swe4.client.services.ServiceFactory;
import swe4.server.services.AnnahmestelleService;
import swe4.server.services.BenutzerService;
import swe4.ui.Annahmestelle;

import java.rmi.RemoteException;

public class BenutzerClientService {
    private final DataService dataService =
ServiceFactory.dataServiceInstance();
    private final BenutzerService benutzerService =
ServiceFactory.benutzerServiceInstance();

    public void insertBenutzer(String name, String password){
        new Thread(() -> {
            try {
                benutzerService.insertUser(name, password);
            } catch (RemoteException e) {
                e.printStackTrace();
```

```
            }
            try {
                dataService.refresh();
            } catch (RemoteException e) {
                e.printStackTrace();
            }
        }).start();
    }
}
```

SpendenankündigungClientService:

```
package swe4.client.services.client;

import swe4.client.services.DataService;
import swe4.client.services.ServiceFactory;
import swe4.server.services.BenutzerService;
import swe4.server.services.SpendenankündigungService;
import swe4.ui.Hilfsgüter;

import java.rmi.RemoteException;

public class SpendenAnkündigungClientService {
    private final DataService dataService =
ServiceFactory.dataServiceInstance();
    private final SpendenankündigungService spendenankündigungService =
ServiceFactory.spendenankündigungServiceInstance();

    public void insertSpendenankündigung(Hilfsgüter hilfsgüter){
        new Thread(() -> {
            try {

spendenankündigungService.insertSpendenankündigung(hilfsgüter);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            try {
                dataService.refresh();
            } catch (RemoteException e) {
                e.printStackTrace();
            }
        }).start();
    }
}
```

ServiceFactory:

```
package swe4.client.services;

import swe4.client.services.client.AnnahmestelleClientService;
import swe4.client.services.client.BenutzerClientService;
import swe4.client.services.client.SpendenAnkündigungClientService;
import swe4.server.config.ServiceConfig;
import swe4.server.services.AnnahmestelleService;
import swe4.server.services.BedarfService;
import swe4.server.services.BenutzerService;
import swe4.server.services.SpendenankündigungService;

import java.rmi.NotBoundException;
import java.rmi.Remote;
```

```
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class ServiceFactory {
    private static BenutzerService benutzerService = null;
    private static AnnahmestelleService annahmestelleService = null;
    private static BedarfService bedarfService = null;
    private static SpendenankündigungService spendenankündigungService =
null;
    private static RefreshService refreshService = null;
    private static Registry registry = null;
    private static DataService dataService = null;

    private static AnnahmestelleClientService annahmestelleClientService =
null;
    private static BenutzerClientService benutzerClientService = null;
    private static SpendenAnkündigungClientService
spendenAnkündigungClientService = null;

    private ServiceFactory() {
        throw new AssertionError("No ServiceFactory Instances for you!");
    }

    public static AnnahmestelleClientService
annahmestelleClientServiceInstance() {
        if(annahmestelleClientService == null){
            annahmestelleClientService = new AnnahmestelleClientService();
        }
        return annahmestelleClientService;
    }

    public static SpendenAnkündigungClientService
spendenAnkündigungClientServiceInstance() {
        if(spendenAnkündigungClientService == null){
            spendenAnkündigungClientService = new
SpendenAnkündigungClientService();
        }
        return spendenAnkündigungClientService;
    }

    public static BenutzerClientService benutzerClientServiceInstance() {
        if(benutzerClientService == null){
            benutzerClientService = new BenutzerClientService();
        }
        return benutzerClientService;
    }

    public static BenutzerService benutzerServiceInstance() {
        if (benutzerService == null) {
            benutzerService = (BenutzerService)
findRmiObject(ServiceConfig.benutzerServiceName());
        }
        return benutzerService;
    }

    public static DataService dataServiceInstance() {
```

```
        if (dataService == null) {
            dataService = new DataService();
        }
        return dataService;
    }

    public static AnnahmestelleService annahmestelleServiceInstance() {
        if (annahmestelleService == null) {
            annahmestelleService = (AnnahmestelleService)
findRmiObject(ServiceConfig.annahmestelleServiceName());
        }
        return annahmestelleService;
    }

    public static BedarfService bedarfServiceInstance() {
        if (bedarfService == null) {
            bedarfService = (BedarfService)
findRmiObject(ServiceConfig.bedarfServiceName());
        }
        return bedarfService;
    }

    public static SpendenankündigungService
spendenankündigungServiceInstance() {
        if (spendenankündigungService == null) {
            spendenankündigungService = (SpendenankündigungService)
findRmiObject(ServiceConfig.spendenankündigungServiceName());
        }
        return spendenankündigungService;
    }

    private static Remote findRmiObject(String name) {
        Registry registry = initializeOrGetRegistry();
        try {
            return registry.lookup(name);
        } catch (RemoteException | NotBoundException e) {
            handleRemoteException(e);
        }
        return null;
    }

    private static Registry initializeOrGetRegistry() {
        if (registry == null) {
            try {
                registry = LocateRegistry.getRegistry(ServiceConfig.port());
            } catch (RemoteException e) {
                handleRemoteException(e);
            }
        }
        return registry;
    }

    public static void startRefreshService() {
        if (refreshService == null || !refreshService.isAlive()) {
            refreshService = new RefreshService();
        }
        refreshService.start();
```

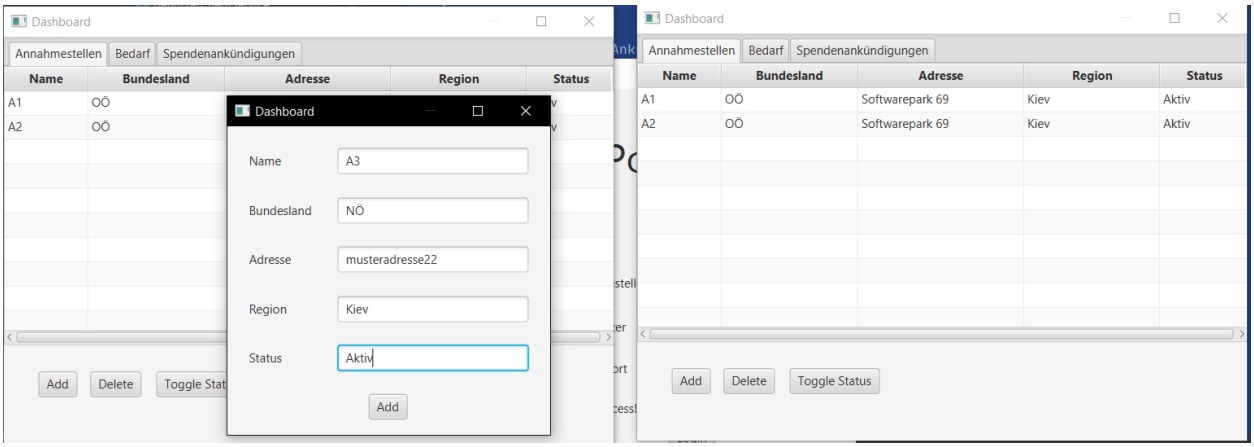
```
    }

    public static void stopRefreshService() {
        if (refreshService != null) {
            refreshService.stopRefreshing();
        }
    }

    private static void handleRemoteException(Exception e) {
        throw new RuntimeException(e);
    }

}
```

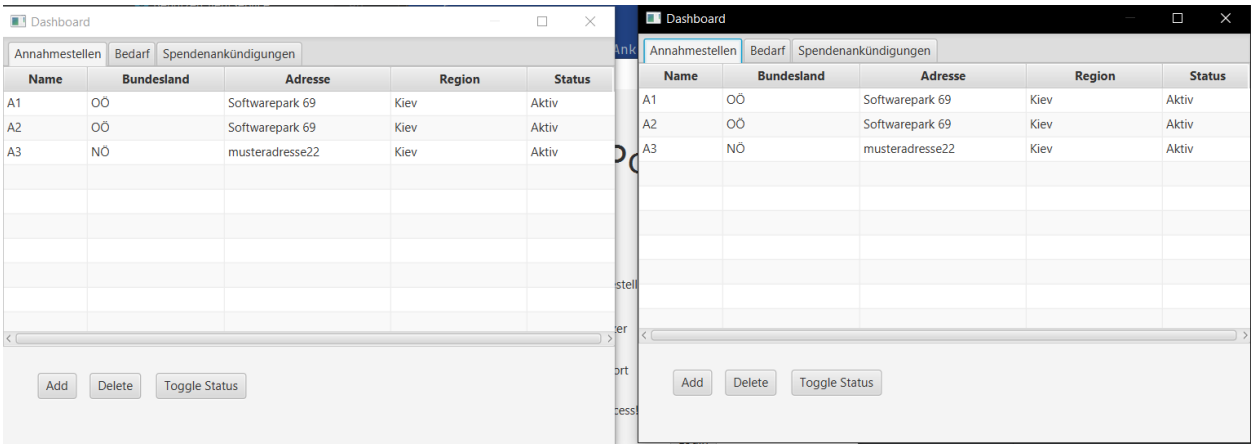
Tests:

ServerStart:

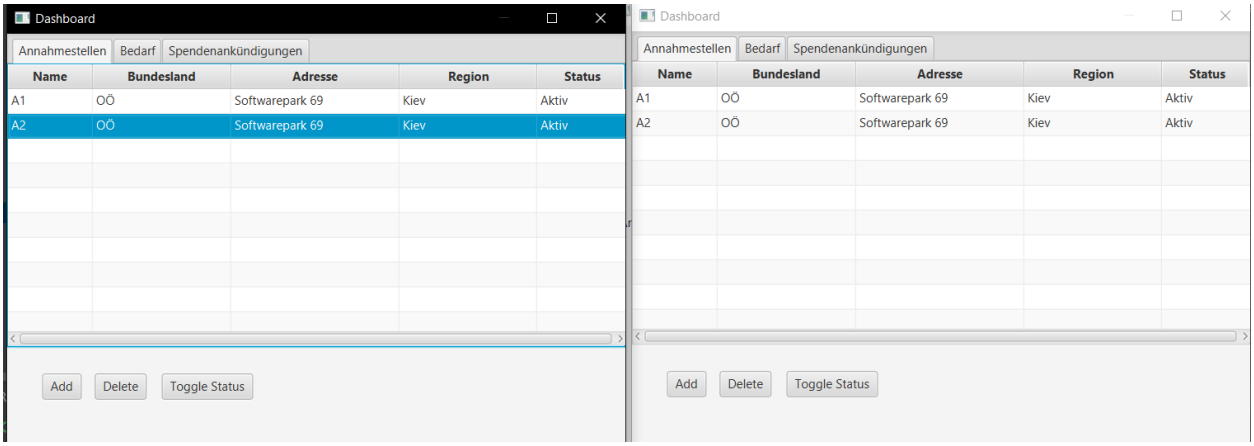
```
Server Running...
```

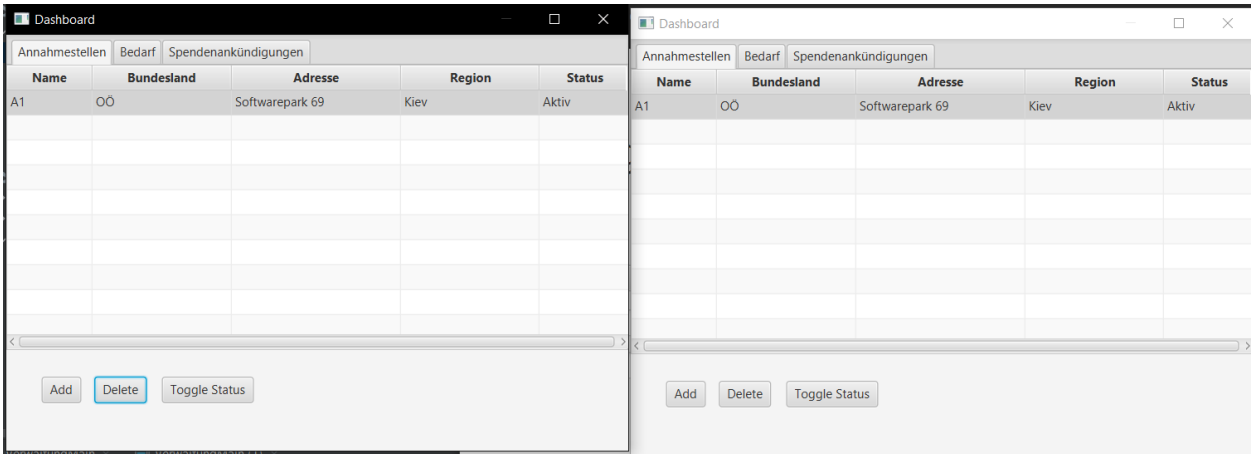
AddNewAnnahmestelle:



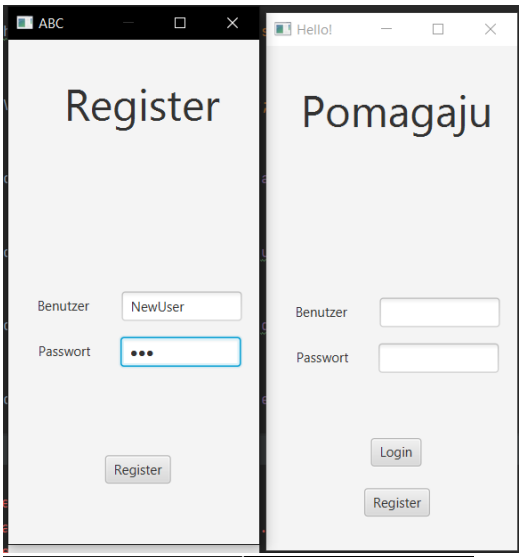


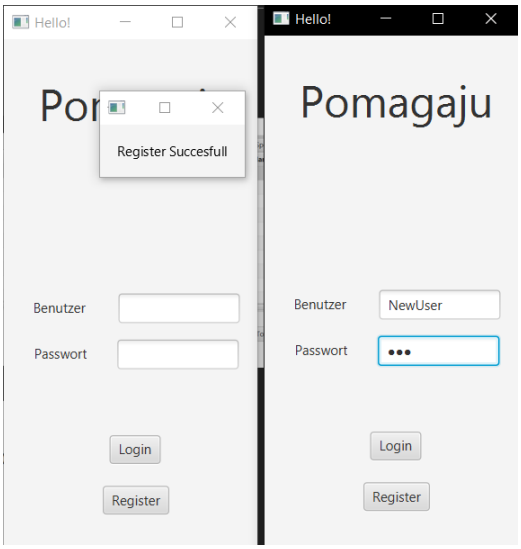
DeleteAnnahmestelle:

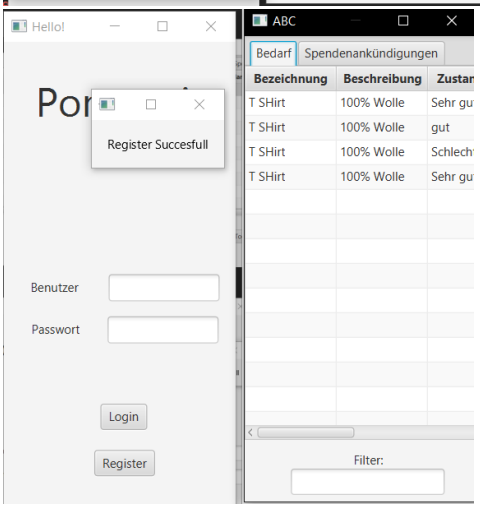




RegisterUser:







AddSpendenankündigung: