Event-Management-System in Java

Projekthierarchie

Packages:

1. `models`:
    - `Person` (Interface), `Kunde`, `Mitarbeiter`, `Event`, `Ort`.
2. `services`:
    - `PersonService`, `EventService`.
3. `utils`:
    - Allgemeine Hilfsmethoden wie E-Mail-Validierung.
4. `main`:
    - Hauptklasse zur Programmausführung.

Package strukture

```
src/
├── main/
│   └── Main.java
├── models/
│   ├── Person.java
│   ├── Kunde.java
│   ├── Mitarbeiter.java
│   ├── Event.java
│   └── Ort.java
├── services/
│   ├── PersonService.java
│   └── EventService.java
├── utils/
│   └── EmailValidator.java
└── exceptions/
    ├── DuplicateEmailException.java
    └── CapacityExceededException.java
```

Erklärung der Package-Struktur

`main/`

- Enthält die Hauptklasse, die das Programm startet.
- Verantwortlich für die Benutzerinteraktion und das Anzeigen des Menüs.
- Zentrale Schnittstelle zwischen Benutzer und Programmlogik.

`models/`

- **Beschreibung**: Definiert die Hauptdatentypen des Systems.
- **Inhalt**:
  - `Person.java`: Interface für die Abstraktion von Personen.
  - `Kunde.java`: Repräsentiert einen Kunden mit spezifischen Attributen.
  - `Mitarbeiter.java`: Repräsentiert einen Mitarbeiter mit einer zusätzlichen Position.
  - `Event.java`: Modelliert ein Event mit Teilnehmerliste, Name und Ort.
  - `Ort.java`: Modelliert einen Veranstaltungsort mit Kapazität.

`services/`

- **Beschreibung**: Beinhaltet die Geschäftslogik des Systems.
- **Inhalt**:
  - `PersonService.java`: Verwaltet Personen (Erstellen, Validieren und Auflisten).
  - `EventService.java`: Kümmert sich um die Verwaltung von Events und deren Teilnehmer.

`utils/`

- **Beschreibung**: Stellt allgemeine Hilfsfunktionen zur Verfügung.
- **Inhalt**:
  - `EmailValidator.java`: Validiert E-Mail-Adressen und verhindert doppelte Einträge.

`exceptions/`

- **Beschreibung**: Beinhaltet benutzerdefinierte Fehlerbehandlungen.
- **Inhalt**:
  - `DuplicateEmailException.java`: Wird ausgelöst, wenn eine E-Mail-Adresse mehrfach verwendet wird.
  - `CapacityExceededException.java`: Wird ausgelöst, wenn die Teilnehmerzahl die Kapazität eines Ortes überschreitet.

Beispiel-Implementierung

Interface `Person`

```
package models;

public interface Person {
    String getId();
    String getName();
    String getEmail();
    String getDetails();
}
```

Klasse `Kunde`

```java
package models;

public class Kunde implements Person {
    private String id;
    private String name;
    private String email;

    public Kunde(String id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }

    @Override
    public String getId() {
        return id;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public String getEmail() {
        return email;
    }

    @Override
    public String getDetails() {
        return "Kunde: " + name + " (Email: " + email + ")";
    }
}
```

Klasse `Mitarbeiter`

```java
package models;

public class Mitarbeiter implements Person {
    private String id;
    private String name;
    private String email;
    private String position;

    public Mitarbeiter(String id, String name, String email, String position) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.position = position;
    }
```

```
    @Override
    public String getId() {
        return id;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public String getEmail() {
        return email;
    }

    public String getPosition() {
        return position;
    }

    @Override
    public String getDetails() {
        return "Mitarbeiter: " + name + " (Position: " + position + ", Email: " +
email + ")";
    }
}
```

Klasse `Ort`

```
package models;

public class Ort {
    private String name;
    private int maxKapazitaet;

    public Ort(String name, int maxKapazitaet) {
        this.name = name;
        this.maxKapazitaet = maxKapazitaet;
    }

    public String getName() {
        return name;
    }

    public int getMaxKapazitaet() {
        return maxKapazitaet;
    }
}
```

Klasse `Event`

```java
package models;

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

public class Event {
    private String name;
    private Ort ort;
    private List<Person> teilnehmerListe;

    public Event(String name, Ort ort) {
        this.name = name;
        this.ort = ort;
        this.teilnehmerListe = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public Ort getOrt() {
        return ort;
    }

    public List<Person> getTeilnehmerListe() {
        return teilnehmerListe;
    }

    public boolean addTeilnehmer(Person person) {
        if (teilnehmerListe.size() >= ort.getMaxKapazitaet()) {
            throw new IllegalArgumentException("Maximale Kapazität des Ortes
erreicht.");
        }
        return teilnehmerListe.add(person);
    }

    public boolean removeTeilnehmer(Person person) {
        return teilnehmerListe.remove(person);
    }
}
```

Service-Klassen

`PersonService`

```java
package services;

import models.Person;
import java.util.ArrayList;
```

```
import java.util.List;

public class PersonService {
    private List<Person> personen;

    public PersonService() {
        this.personen = new ArrayList<>();
    }

    public void addPerson(Person person) {
        for (Person p : personen) {
            if (p.getEmail().equalsIgnoreCase(person.getEmail())) {
                throw new IllegalArgumentException("E-Mail-Adresse bereits
vorhanden.");
            }
        }
        personen.add(person);
    }

    public List<Person> getPersonen() {
        return personen;
    }
}
```

EventService

```
package services;

import models.Event;
import java.util.ArrayList;
import java.util.List;

public class EventService {
    private List<Event> events;

    public EventService() {
        this.events = new ArrayList<>();
    }

    public void addEvent(Event event) {
        events.add(event);
    }

    public List<Event> getEvents() {
        return events;
    }
}
```

Hilfsklasse für Validierungen

`EmailValidator`

```
package utils;

import java.util.regex.Pattern;

public class EmailValidator {
    private static final String EMAIL_PATTERN =
            "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,6}$";

    public static boolean isValid(String email) {
        return Pattern.matches(EMAIL_PATTERN, email);
    }
}
```

Main-Klasse

```
package main;

import models.*;
import services.*;
import utils.EmailValidator;

public class Main {
    public static void main(String[] args) {
        PersonService personService = new PersonService();
        EventService eventService = new EventService();

        // Beispiel: Hinzufügen von Personen
        Person kunde1 = new Kunde("1", "Max Mustermann", "max@domain.com");
        Person mitarbeiter1 = new Mitarbeiter("2", "Anna Schmidt",
"anna@domain.com", "Manager");

        personService.addPerson(kunde1);
        personService.addPerson(mitarbeiter1);

        // Beispiel: Hinzufügen eines Events
        Ort ort = new Ort("Auditorium", 50);
        Event event = new Event("Konferenz", ort);
        event.addTeilnehmer(kunde1);
        event.addTeilnehmer(mitarbeiter1);

        eventService.addEvent(event);

        // Ausgabe
        System.out.println("Personen:");
        personService.getPersonen().forEach(p ->
System.out.println(p.getDetails()));
```

```
        System.out.println("\nEvents:");
        eventService.getEvents().forEach(e -> {
            System.out.println("Event: " + e.getName() + ", Ort: " +
e.getOrt().getName());
            System.out.println("Teilnehmer:");
            e.getTeilnehmerListe().forEach(t ->
System.out.println(t.getDetails()));
        });
    }
}
```

Möglicher Output

Personen: Kunde: Max Mustermann (Email: max@domain.com) Mitarbeiter: Anna Schmidt (Position: Manager, Email: anna@domain.com)

Events: Event: Konferenz, Ort: Auditorium Teilnehmer: Kunde: Max Mustermann (Email: max@domain.com) Mitarbeiter: Anna Schmidt (Position: Manager, Email: anna@domain.com)
