Detaillierte Dokumentation: Event-Management-System für Anfänger

Dieses Dokument enthält eine schrittweise Anleitung zur Erstellung eines Event-Management-Systems, das in Meilensteine unterteilt ist. Jeder Meilenstein enthält erklärenden Code und detaillierte Beschreibungen.

Meilenstein 1: Projektstruktur und Basisklassen

Ziel

- Einführung in grundlegende OOP-Konzepte wie Klassen, Methoden und Attribute.

Implementierung

Erstelle eine `Kunde`-Klasse mit den Attributen `id`, `name` und `email`. Schreibe Methoden für Getter, Setter und eine `toString()`-Methode.

```
public class Kunde {
    private int id;
    private String name;
    private String email;

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

    // Getter und Setter
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
```

```
    }

    public void setEmail(String email) {
        this.email = email;
    }

    // toString-Methode
    @Override
    public String toString() {
        return "Kunde{" +
               "id=" + id +
               ", name='" + name + '\'' +
               ", email='" + email + '\'' +
               '}';
    }
}
```

Hauptprogramm

```
public class Main {
    public static void main(String[] args) {
        Kunde kunde = new Kunde(1, "Max Mustermann", "max@example.com");
        System.out.println(kunde);
    }
}
```

Ergebnis: Die Details des Kunden werden in der Konsole ausgegeben.

Meilenstein 2: Einführung von Listen

Ziel

- Verwalten mehrerer Objekte mit einer Liste.

Implementierung

Erweitere das Programm, um mehrere Kunden in einer `ArrayList` zu speichern und zu verwalten.

```
import java.util.ArrayList;

public class Main {
    private static ArrayList<Kunde> kundenListe = new ArrayList<>();

    public static void main(String[] args) {
        addKunde(new Kunde(1, "Max Mustermann", "max@example.com"));
        addKunde(new Kunde(2, "Anna Schmidt", "anna@example.com"));

        listKunden();
```

```
    }

    // Kunde zur Liste hinzufügen
    public static void addKunde(Kunde kunde) {
        kundenListe.add(kunde);
    }

    // Alle Kunden anzeigen
    public static void listKunden() {
        for (Kunde kunde : kundenListe) {
            System.out.println(kunde);
        }
    }
}
```

Ergebnis: Mehrere Kunden werden verwaltet und in der Konsole angezeigt.

Meilenstein 3: Mitarbeiter und Vererbung

Ziel

- Nutzung von Vererbung und Polymorphismus.

Implementierung

Erstelle ein Interface `Person`, das von `Kunde` und `Mitarbeiter` implementiert wird.

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

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

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

    @Override
    public int getId() {
```

```
        return id;
    }

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

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

    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }

    @Override
    public String toString() {
        return "Mitarbeiter{" +
               "id=" + id +
               ", name='" + name + '\'' +
               ", email='" + email + '\'' +
               ", position='" + position + '\'' +
               '}';
    }
}
```

Hauptprogramm

```
public class Main {
    private static ArrayList<Person> personenListe = new ArrayList<>();

    public static void main(String[] args) {
        addPerson(new Kunde(1, "Max Mustermann", "max@example.com"));
        addPerson(new Mitarbeiter(2, "Anna Schmidt", "anna@example.com",
"Manager"));

        listPersonen();
    }

    public static void addPerson(Person person) {
        personenListe.add(person);
    }

    public static void listPersonen() {
        for (Person person : personenListe) {
```

```
            System.out.println(person);
        }
    }
}
```

Ergebnis: Sowohl Kunden als auch Mitarbeiter können verwaltet und angezeigt werden.

Meilenstein 4: Events und Orte

Ziel

- Einführung von Klassen für Events und Orte.

Implementierung

```
import java.util.ArrayList;

public class Ort {
    private int id;
    private String name;
    private int capacity;

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

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public int getCapacity() {
        return capacity;
    }

    @Override
    public String toString() {
        return "Ort{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", capacity=" + capacity +
                '}';
    }
}
```

```
public class Event {
    private int id;
    private String name;
    private Ort ort;
    private ArrayList<Kunde> teilnehmer;

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

    public void addTeilnehmer(Kunde kunde) {
        if (teilnehmer.size() < ort.getCapacity()) {
            teilnehmer.add(kunde);
        } else {
            System.out.println("Kapazität überschritten!");
        }
    }

    @Override
    public String toString() {
        return "Event{" +
               "id=" + id +
               ", name='" + name + '\'' +
               ", ort=" + ort +
               ", teilnehmer=" + teilnehmer +
               '}';
    }
}
```

Hauptprogramm

```
public class Main {
    public static void main(String[] args) {
        Ort ort = new Ort(1, "Konferenzraum", 2);
        Event event = new Event(1, "Tech Talk", ort);

        Kunde kunde1 = new Kunde(1, "Max Mustermann", "max@example.com");
        Kunde kunde2 = new Kunde(2, "Anna Schmidt", "anna@example.com");
        Kunde kunde3 = new Kunde(3, "Tom Müller", "tom@example.com");

        event.addTeilnehmer(kunde1);
        event.addTeilnehmer(kunde2);
        event.addTeilnehmer(kunde3); // Kapazität überschritten!

        System.out.println(event);
    }
}
```

Ergebnis: Ein Event mit Teilnehmern und Kapazitätsprüfung wird erfolgreich verwaltet.

Weitere Meilensteine folgen der gleichen Struktur, mit Erweiterungen wie Validierungen, Benutzerinteraktionen und Statistiken.