Aufgabe: Entwicklung eines erweiterten Event-Management-Systems in Java

Das Ziel dieses Projekts ist es, ein umfassendes Event-Management-System zu entwickeln, das die wichtigsten Java-Konzepte und Best Practices integriert. Dieses System soll Kunden, Mitarbeiter, Events, Teilnehmer und Orte verwalten, sowie zusätzliche Validierungen und erweiterte Funktionen bieten.

Anforderungen

1. Grundlegende Funktionalitäten

- Verwaltung von **Personen**:
  - Personen werden über ein Interface `Person` abstrahiert, welches von `Kunde` und `Mitarbeiter` implementiert wird.
  - Jede Person hat eine eindeutige ID, einen Namen und eine E-Mail-Adresse.
  - Die E-Mail-Adresse muss gültig sein und darf nicht doppelt vorkommen.
  - Mitarbeiter haben zusätzlich eine Position (z. B. Manager, Techniker, Kundenbetreuer).
- Verwaltung von **Events**:
  - Jedes Event hat einen eindeutigen Namen, einen Ort und eine Teilnehmerliste.
  - Teilnehmer können hinzugefügt und entfernt werden.
  - Ein Event darf die Kapazität seines zugeordneten Ortes nicht überschreiten.
- Verwaltung von **Orten**:

- Jeder Ort hat einen eindeutigen Namen und eine maximale Kapazität.
- Orte können Events zugewiesen werden.

2. Technische Anforderungen

Validierungen und Fehlerbehandlung:

1. **E-Mail-Validierung**:
   - Überprüfen Sie, ob die E-Mail-Adresse einem gültigen Muster entspricht (z. B. `name@domain.com`).
   - Verhindern Sie doppelte E-Mail-Adressen in der Personenliste.
2. **Try-Catch-Blöcke**:
   - Behandeln Sie Ausnahmen für ungültige Eingaben (z. B. nicht vorhandene IDs, falsche Formate).
   - Werfen und fangen Sie benutzerdefinierte Exceptions (z. B. für doppelte E-Mail-Adressen).
3. **Kapazitätsprüfung**:
   - Stellen Sie sicher, dass ein Event nicht mehr Teilnehmer aufnehmen kann, als der zugewiesene Ort erlaubt.

Objektorientierte Programmierung:

1. **Interface** `Person`:
   - Gemeinsame Attribute und Methoden für Kunden und Mitarbeiter.
   - Implementieren Sie Vererbung, um Redundanz zu vermeiden.
2. **Kapselung**:
   - Nutzen Sie private Attribute und öffentlich zugängliche Getter/Setter.
3. **Abstrakte Klassen**:
   - Optional für zusätzliche Abstraktionen und Standardimplementierungen.

3. Erweiterte Funktionen

1. **Eindeutige ID-Zuweisung**:

    - Jede Person und jedes Event erhält eine eindeutige ID, die mit `Random` generiert wird.
2. **Statistiken**:
    - Zeigen Sie die Anzahl von Events, Kunden und Mitarbeitern an.
    - Zeigen Sie die durchschnittliche Teilnehmeranzahl pro Event.
3. **Such- und Filterfunktionen**:
    - Suche nach Events oder Personen basierend auf Namen oder IDs.
4. **Menübasierte Interaktion**:
    - Bieten Sie eine benutzerfreundliche Oberfläche mit einem textbasierten Menü.

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.

Beispiele für Klassenstruktur

Interface `Person` :

- Gemeinsame Attribute: `id` , `name` , `email` .
- Gemeinsame Methoden: `getDetails()` .

Klassen `Kunde` und `Mitarbeiter` :

- `Kunde` : Implementiert `Person` .
- `Mitarbeiter` : Hat zusätzlich eine Position (z. B. `Manager` , `Techniker`).

Event-Klasse:

- Attribute:
  - Name des Events.
  - Ort (Objekt).
  - Teilnehmerliste (Kunden).
- Methoden:
  - Teilnehmer hinzufügen/entfernen.
  - Validierung der Kapazität.

Ort-Klasse:

- Attribute:
  - Name.
  - Maximale Kapazität.
- Methoden:
  - Zuweisung von Events.

Exception Handling:

- Benutzerdefinierte Exception `DuplicateEmailException` für doppelte E-Mails.
- `IllegalArgumentException` für ungültige IDs oder Kapazitätsüberschreitungen.

Benutzeroberfläche

Menüpunkte:

1. Personenmanagement:
   - Kunde hinzufügen.
   - Mitarbeiter hinzufügen.
   - Liste aller Personen anzeigen.
2. Eventmanagement:
   - Event erstellen.
   - Teilnehmer hinzufügen/entfernen.
   - Eventdetails anzeigen.

3. Statistiken anzeigen.
4. Beenden.

Beispiel-Ablauf:

1. Benutzer fügt Kunden und Mitarbeiter hinzu.
2. Benutzer erstellt Events und weist Teilnehmer zu.
3. Benutzer zeigt Statistiken an (z. B. Anzahl von Events, durchschnittliche Teilnehmeranzahl).
4. Fehlerhafte Eingaben (z. B. doppelte E-Mails, ungültige IDs) werden durch Exceptions abgefangen und gemeldet.

Ziel

Das Ziel dieses Projekts ist es, ein robustes und vollständig objektorientiertes System zu erstellen, das alle zentralen Java-Konzepte integriert. Es soll flexibel erweiterbar sein und die besten Praktiken für Programmierung und Softwaredesign demonstrieren.