**Aufgabe 4: Entitäten und Aggregates**

*Entitäten und Aggregates definieren: Definieren Sie die Entitäten und Aggregates innerhalb Ihrer Bounded Contexts, die aus den identifizierten Events und dem Domänenmodell abgeleitet werden. Bestimmen Sie, welche Attribute und Methoden sie haben sollten.*

**Bounded Contexts**

* ***Bürger-Kontext:****Bürger erstellt, reicht ein und verwaltet seine Beschwerden.*
* ***Stadtverwaltungs-Kontext****:   
  (Stadt-)Mitarbeiter bearbeitet eingereichte Beschwerden, aktualisiert deren Fortschritt.*
* ***Fortschrittsverfolgungs-Kontext****:   
  zur Verfolgung und Verwaltung des (Fortschritts-)Status von Beschwerden*
* ***Kategorisierungs-Kontext****:   
  zur Verwaltung der Kategorien, die den Beschwerden zugeordnet werden können.*

**Entitäten und ihre Attribute & Methoden**

**Entität: Bürger**

|  |  |
| --- | --- |
| **Attribute** | **Methoden** |
| * *bürgerID: int* * *vorname: String* * *nachname: String* * *email: String* * *telefon: String* * *beschwerden: List<Beschwerden>* | * *beschwerdeErstellen(titel: String, kategorie: Kategorie): Beschwerde* * *beschwerdeEinreichen(beschwerdeID: int): void* * *beschwerdeAnzeigen(beschwerdeID: int): Beschwerde* * *alleBeschwerdenAnzeigen(): List <Beschwerde>* * *Getter & Setter* |

**Entität: (Stadt-)Mitarbeiter**

|  |  |
| --- | --- |
| **Attribute** | **Methoden** |
| * *mitarbeiterID: int* * *vorname: String* * *nachname: String* * *email: String* * *telefon: String* * *beschwerden: List<Beschwerde>* | * *beschwerdeBearbeitungStarten(beschwerdeID): void* * *beschwerdeAnzeigen(beschwerdeID: int): Beschwerde* * *alleBeschwerdenAnzeigen(): List <Beschwerde>* * *Getter & Setter* |

***Nachtrag für Implementierung:***

*Superclass: Benutzer, Subclasses: Bürger, Mitarbeiter, Admin*

**Entität: Beschwerde**

|  |  |
| --- | --- |
| **Attribute** | **Methoden** |
| * *beschwerdeID: int* * *titel: String* * *kategorie: Kategorie* * *fortschritt: Fortschritt* | * *Getter & Setter* |

**Entität: Kategorie**

|  |  |
| --- | --- |
| **Attribute** | **Methoden** |
| * *kategorieID: int* * *titel: String* | * *Getter & Setter* |

**Entität: Fortschritt**

|  |  |
| --- | --- |
| **Attribute** | **Methoden** |
| * *status: String* * *datumLetzteÄnderung: date* * *mitarbeiterIDLetzteÄnderung: int* | * *zeigeStatus(): String* * *aktualisiereStatus(neuerStatus: String, mitarbeiterID: int, datum: date)* * *Getter & Setter* |

Hinweis: Getter & Setter sind nicht explizit ausformuliert, aber vorgesehen. Ebenso die Konstruktoren.

**Aggregates & Beziehungen:**

* **Beschwerde ist Bestandteil eines Bürgers** (von ihm erstellt), kann aber auch unabhängig von ihm existieren (z.B. wenn Bürger nicht mehr im System ist, bleibt Beschwerde im System bestehen) 🡪 Aggregate
  + Ein Bürger kann mehrere Beschwerden haben (1..\*)
  + Eine Beschwerde gehört genau einem Bürger (1..1)
* **Beschwerde ist Bestandteil eines Stadtmitarbeiters** (von ihm bearbeitet), kann aber auch unabhängig von ihm existieren (z.B. wenn Mitarbeiter nicht mehr im System ist, bleibt Beschwerde im System bestehen) 🡪 Aggregate
  + Ein Mitarbeiter kann mehrere Beschwerden bearbeiten (1..\*)
  + Eine Beschwerde wird von genau einem Mitarbeiter (hauptverantwortlich) bearbeitet (1..1)
* **Fortschritt ist Bestandteil einer Beschwerde**, und existiert nur in Verbindung mit ihr (wenn Beschwerde gelöscht wird, wird auch Fortschritt gelöscht) 🡪 Komposition
  + Eine Beschwerde hat genau einen Fortschritt (1..1)
  + Ein Fortschritt existiert nur in Verbindung mit einer Beschwerde (1..1)
* **Kategorie ist Bestandteil einer Beschwerde** (beschreibt Art der Beschwerde), existiert jedoch auch unabhängig von ihr (z.B. wenn die Beschwerde gelöscht wird, bleibt die Kategorie weiterhin verfügbar). 🡪 Aggregate
  + Eine Beschwerde hat genau eine Kategorie (1..1)
  + Eine Kategorie kann vielen Beschwerden zugeordnet sein (0..\*)

*Ziel: Ein detailliertes Verständnis der Datenstruktur und der Logik in Ihrer Anwendung.*

*Output: Erstellen Sie ein Dokument, das jede Entität und ihr zugehöriges Aggregate beschreibt, einschließlich ihrer Beziehungen.*

**Visualisierung per UML**

