Klausur Anwendungsentwicklung, Oberstufe, 1. Hj.

# Aufgabe 1 – Projektmanagement & graphische Benutzeroberflächen (8 Pkt.)

## 1a

**Welche Aussage zur Projektleitung trifft zu?**

Die Projektleitung legt die Regeln und Rollen im Projekt fest und bestimmt maßgeblich die Teambildung.

Die Projektleitung ist für die Durchführung des Projekts verantwortlich.

Die Projektleitung berichtet direkt an den Lenkungsausschuss.

Alle Aussagen sind richtig.

## 1b

**Welche Elemente der Projektorganisation nach DIN 69901 gehören nicht zu den klassischen Projektinstanzen?**

Lenkungsausschuss

Gesamtprojektleitung

Teilprojektleitungen

Projektbeteiligte

## 1c

**Welche Aussage zu den Projektbeteiligten trifft nicht zu?**

Der Auftraggeber initiiert den Projektauftrag und entscheidet über Abbruch oder Fortführung des Projekts.

Der Lenkungsausschuss trifft in allen Projektphasen Grundsatzentscheidungen.

Stakeholder sind nur externe Personen, die vom Projekt betroffen sind.

Das Projektteam bearbeitet unter der Leitung der Projektleitung den Projektauftrag.

## 1d

**Welche Aussage beschreibt am besten den Unterschied zwischen User Interface (UI) und User Experience (UX) im Kontext von graphischen Benutzeroberflächen?**

UI bezieht sich auf die Struktur und Funktionalität eines Produkts, während UX sich mit der visuellen Gestaltung und Ästhetik beschäftigt.

UI umfasst alle interaktiven Elemente, die ein Benutzer sehen und verwenden kann, während UX das Gesamtgefühl und die Zufriedenheit des Benutzers mit der Verwendung des Produkts beschreibt.

UI ist ausschließlich auf mobile Anwendungen bezogen, während UX nur für Desktop-Anwendungen gilt.

UI konzentriert sich auf die Interaktion zwischen Softwareentwicklern, während UX die Interaktion zwischen Projektmanagern fokussiert.

# Aufgabe 2 - Entity Relationship Model (10 Pkt.)

## ER-Modell für eine Bibliothek

**Aufgabenstellung:**

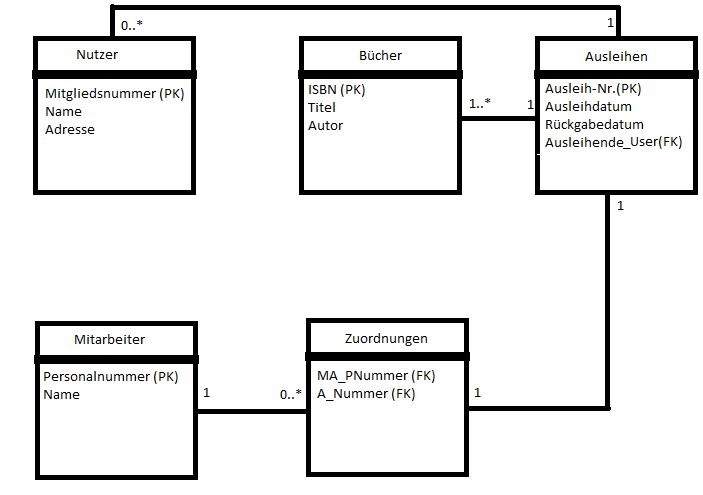
Sie sind als Fachinformatiker in einer kleinen Bibliothek tätig. Die Bibliotheksleitung hat Sie beauftragt, ein ER-Modell für die Verwaltung der Bibliotheksbestände und -ausleihen zu erstellen, da das System bisher komplett auf Papier verwaltet wird. Nun soll es digitalisiert werden.

Folgende Anforderungen sollen in Ihrem ER-Modell berücksichtigt werden:

1. **Nutzer**: Jeder Nutzer hat eine eindeutige Mitgliedsnummer, einen Namen und eine Adresse.
2. **Bücher**: Jedes Buch hat eine ISBN, einen Titel und einen Autor.
3. **Ausleihen**: Für jede Ausleihe eines Buches wird ein Ausleihvorgang erstellt. Zu einem Ausleihvorgang gehören die Ausleih-Nr., das Ausleih- und Rückgabedatum sowie der ausleihende Nutzer.
4. **Mitarbeiter**: In der Bibliothek arbeiten Bibliothekare. Jeder Mitarbeiter hat eine Personalnummer und einen Namen.
5. **Zuordnungen**: Jede Ausleihe wird von einem Mitarbeiter bearbeitet.

Erstellen Sie ein ER-Modell, das die oben genannten Anforderungen abbildet. Achten Sie dabei auf die korrekten Kardinalitäten zwischen den Entitäten.

*Hinweis: Verwenden Sie für die grafische Darstellung des ER-Modells die bekannten Konventionen, achten Sie auch darauf, starke und schwache Entitäten zu unterscheiden.*

Lösung als Bild hier einfügen: 

# Aufgabe 3 - Datenbanken und SQL (10 Pkt.)

## Analyse von Verkaufsdaten

Sie haben Zugriff auf eine Datenbank eines Einzelhandelsunternehmens mit folgenden Tabellen:

**Tabelle: Produkte**

- produkt\_id

- produktname

- kategorie

- preis

**Tabelle: Verkäufe**

- verkaufs\_id

- produkt\_id

- kunde\_id

- verkaufsdatum

- menge

- umsatz

**Tabelle: Kunden**

- kunde\_id

- name

- adresse

- email

- telefon

**Ihre Aufgabe ist es, verschiedene Analysen zu den Verkaufsdaten durchzuführen und die Ergebnisse in SQL-Abfragen darzustellen.**

**1. Umsatz pro Produktkategorie**

- Erstellen Sie eine Abfrage, die den Gesamtumsatz10 für jede Produktkategorie ausgibt.

- Sortieren Sie das Ergebnis absteigend nach dem Gesamtumsatz.

BEGIN

SELECT sum(V.Umsatz) as Umsatz

FROM Verkäufe V, Produkte P

WHERE P.produkt\_id = V.produkt\_id

GROUP BY P.Kategorie

END

**2. Topprodukte nach Umsatz**

- Schreiben Sie eine Abfrage, die die 5 umsatzstärksten Produkte ausgibt.

- Zeigen Sie dafür den Produktnamen, den Preis und den Gesamtumsatz an.

BEGIN

SELECT TOP(5) P.produktname AS Produkt, P.preis AS Preis, SUM(V.umsatz) AS Gesamtumsatz

FROM Produkte P, Verkäufe V

WHERE P.produkt\_id = V.produkt\_id

GROUP BY P.produktname

ORDER BY V.umsatz

END

*Hinweise:*

* *Verwenden Sie geeignete JOIN-Befehle, um die Tabellen miteinander zu verknüpfen.*
* *Nutzen Sie Aggregatfunktionen wie SUM(), AVG(), COUNT() etc., um die gewünschten Berechnungen durchzuführen.*
* *Verwenden Sie Aliasnamen für eine übersichtliche Darstellung der Ergebnisse.*

# Aufgabe 4 – Programmierung (15 +3 Pkt.)

## Kundenverwaltungssystem

**Hintergrund**:

Sie arbeiten an einem Kundenverwaltungssystem für ein kleines Unternehmen. Das System soll in der Lage sein, Kundeninformationen zu speichern und eine einfache Suche nach Kunden durchzuführen.

**Aufgabenstellung**:

Implementieren Sie einen Pseudocode (oder Code in einer bekannten Programmiersprache) für die folgenden Anforderungen:

1. **Datenstruktur**:

Erstellen Sie eine Datenstruktur `Kunde`, die die folgenden Attribute enthält:

- `KundenID` (eindeutige Identifikationsnummer)

- `Name` (Name des Kunden)

- `Email` (E-Mail-Adresse des Kunden)

public struct Kunde

{

public int KundenID;

public string Name;

public string EMail;

}

2. **Kundenliste**:

Erstellen Sie eine Liste, um mehrere Kundenobjekte zu speichern.

public List<Kunde> Kundenliste = new List<Kunde>();

3. **Kunde hinzufügen:**

Schreiben Sie eine Funktion `KundeHinzufuegen(KundenListe, KundenID, Name, Email)`, die einen neuen Kunden zur Liste hinzufügt. Überprüfen Sie, ob die `KundenID` bereits existiert. Wenn ja, geben Sie eine Fehlermeldung aus.

public List<Kunde> KundeHinzufuegen(List<Kunde> Kundenliste, int KundenID, string Name, string EMail)

{

bool istNeu = true;

foreach (Kunde K in Kundenliste)

{

if (K.KundenID == KundenID)

{

Console.Write("Kunde bereits vorhanden");

istNeu = false;

break;

}

else { /\* Skip \*/ }

}

if (istNeu)

{

Kunde neuerKunde = new Kunde();

neuerKunde.KundenID = KundenID;

neuerKunde.Name = Name;

neuerKunde.EMail = EMail;

Kundenliste.Add(neuerKunde);

}

else { /\* Kunde existiert bereits \*/ }

return Kundeliste;

}

4. **Kunde suchen:**

Schreiben Sie eine Funktion `KundeSuchen(KundenListe, KundenID)`, die einen Kunden anhand seiner `KundenID` sucht und die Informationen des Kunden zurückgibt. Wenn der Kunde nicht gefunden wird, geben Sie eine entsprechende Nachricht aus.

public Kunde? KundeSuchen (List<Kunde> Kundenliste, int KundenID)

{

Kunde gefundenerKunde;

foreach (Kunde K in Kundenliste)

{

if (K.KundenID == KundenID)

{

gefundenerKunde = K;

}

else { /\* Continue \*/ }

}

if (gefundenerKunde == null)

{

Console.WriteLine("Kein Kunde mit KundenID '{0}' wurde gefunden.", KundenID);

}

return gefundenerKunde;

}

5. **Kunde entfernen:**

Schreiben Sie eine Funktion `KundeEntfernen(KundenListe, KundenID)`, die einen Kunden anhand seiner `KundenID` aus der Liste entfernt. Geben Sie eine Bestätigung aus, wenn der Kunde erfolgreich entfernt wurde, oder eine Fehlermeldung, wenn der Kunde nicht gefunden wurde.

public List<Kunde> KundeEntfernen (List<Kunde> Kundenliste, int KundenID)

{

bool kundeGefunden = false;

foreach (Kunde K in Kundenliste)

{

if (K.KundenID == KundenID)

{

List.Remove(K);

}

else { /\* Continue \*/ }

}

if (kundeGefunden == false)

{

Console.WriteLine("Es wurde kein Kunde mit der KundenID '{0}' zum löschen gefunden.", KundenID);

}

else

{

Console.WriteLine("Kunde mit der KundenID '{0}' wurde erfolgreich gelöscht.", KundenID);

}

return Kundenliste;

}

6. **Zusatzfrage (+ 3 Pkt.)**:

Welche bekannte Datenstruktur ist besonders gut geeignet für die Realisierung dieser Aufgabe und warum?

*Hier ist ein einfaches Beispiel, wie die Funktionen verwendet werden können sollen:*

*// Hauptprogramm*

*KundenListe = []*

*KundeHinzufuegen(KundenListe, 1, "Max Mustermann", "max@beispiel.de")*

*KundeHinzufuegen(KundenListe, 2, "Erika Mustermann", "erika@beispiel.de")*

*KundeSuchen(KundenListe, 1) // Sollte Max Mustermann finden*

*KundeEntfernen(KundenListe, 1) // Sollte Max Mustermann entfernen*

*KundeSuchen(KundenListe, 1) // Sollte "Kunde nicht gefunden" ausgeben*

Antwort:

Besonders gut geeignet wäre ein Dictionary. Kunden könne über den Key schnell gesucht werden und das Entfernen über den Key ist ebenfalls schneller, als erst über eine Liste oder ein Array iterieren zu müssen