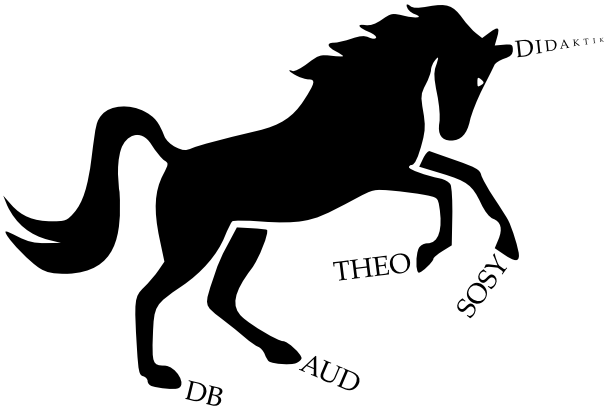
66116
Frühjahr
2016

Datenbanksysteme / Softwaretechnologie (vertieft)

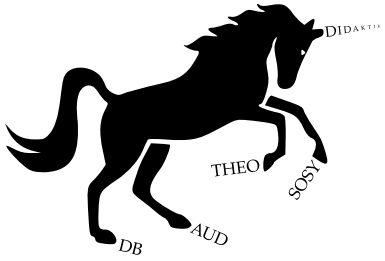
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Teilaufgabe Nr. 1

Forstverwaltung [Forstverwaltung]

Für die bayerische Forstverwaltung wird eine Datenbank zur Erschließung einer Jagd-Statistik benötigt. Gehen Sie dabei von folgendem Szenario aus:

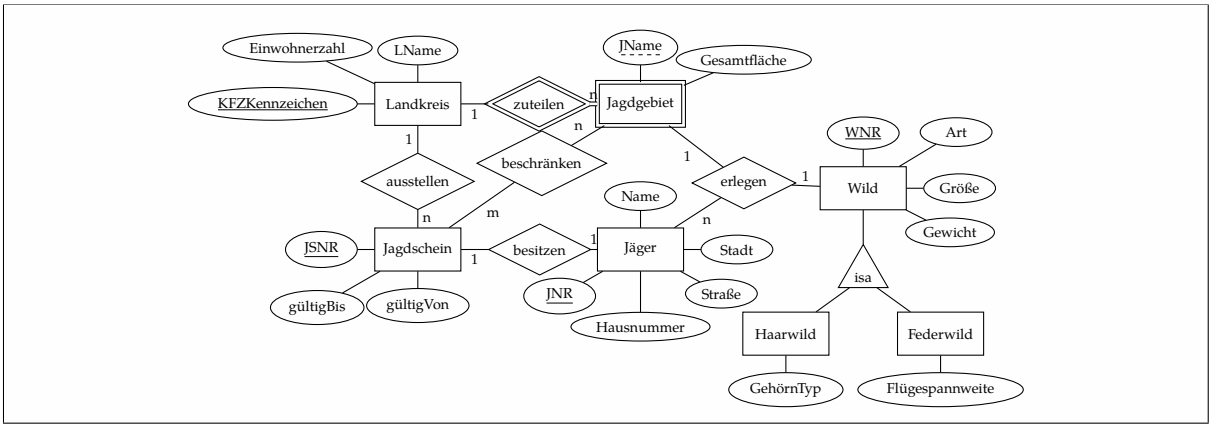
- Die Administration von Jagdgebieten obliegt den Landkreisen. Jeder **Landkreis** besitzt, neben seinem *Namen* (LName) und der *Einwohnerzahl*, ein eindeutiges *KFZ-Kennzeichen* (`KFZKennzeichen`).
- Die Jagd findet in Jagdgebieten statt. Ein **Jagdgebiet** soll dem Landkreis zugeteilt werden, indem es liegt. Gehen Sie davon aus, dass Jagdgebiete nicht in mehreren Landkreisen liegen können. Zusätzlich ist für jedes Jagdgebiet der *Name* (`JName`) und die *Gesamtfläche* zu speichern. Dabei ist zu beachten, dass die Namen nur innerhalb eines einzelnen Landkreises eindeutig sind.
- Die Erlaubnis zum Jagen wird durch einen **Jagdschein** erteilt. Dieser kann nur von einem Landkreis ausgestellt werden und beschränkt sich auf ein oder mehrere Jagdgebiete. Er wird durch eine *Jagdschein-Nummer* (`JSNR`) identifiziert und ist in einem bestimmtem Zeitintervall gültig. Dieses soll über zwei Zeitpunkte festgelegt werden (*gültig von* (`gültigVon`), *gültig bis* (`gültigBis`)).
- Ein **Jäger** besitzt genau einen Jagdschein. Zu einem Jäger sollen *Name*, *Stadt*, *Straße* und *Hausnummer*, gespeichert werden. Da die Jagdtradition innerhalb einer Familie häufig von einer zur nächsten Generation weitergegeben wird, kann es vorkommen, dass Name und Adresse von zwei unterschiedlichen Jägern gleich ist (z. B. Vater und Sohn). Aus diesem Grund ist eine eindeutige *Identifikationsnummer* (`JNR`) notwendig.
- Um Statistiken erheben zu können, muss berücksichtigt werden, welches **Wild** von welchen Jägern zu welchem Zeitpunkt in welchem Jagdgebiet erlegt worden ist. Gehen Sie davon aus, dass es mehrere Jäger geben kann, die gemeinsam ein Wild erlegen (z. B. in einer Jagdgesellschaft). Zu einem Wild gehört die *Art* (z. B. Reh), die *Größe*, das *Gewicht*, sowie eine eindeutige *Identifikationsnummer* (`WNR`). Zusätzlich unterscheidet man zwischen **Haarwild** und **Federwild**, wobei beim Haarwild der *Typ des Gehörns* (`GehörnTyp`) (z. B. Hirschgeweih) und beim Federwild die *Flügelspannweite* betrachtet werden soll.

(a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell in Chen-Notation. Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- die Funktionalitäten der Relationship-Typen.

□ **E: Landkreis**
○ *A: Namen*
○ *A: Einwohnerzahl*
○ *A: KFZ-Kennzeichen*
□ **E: Jagdgebiet**
◇ R: zugeteilt
○ *A: Name*
○ *A: Gesamtfläche*
□ **E: Jagdschein**
◇ R: ausgestellt
◇ R: beschränkt
○ *A: Jagdschein-Nummer*
○ *A: gültig von*
○ *A: gültig bis*
□ **E: Jäger**
◇ R: besitzt
○ *A: Name*
○ *A: Stadt*
○ *A: Straße*
○ *A: Hausnummer*
○ *A: Identifikationsnummer*
□ **E: Wild**
◇ R: erlegt
○ *A: Art*
○ *A: Größe*
○ *A: Gewicht*
○ *A: Identifikationsnummer*
□ **E: Haarwild**
□ **E: Federwild**
○ *A: Typ des*

Lösungsvorschlag



(b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Lösungsvorschlag

Landkreis(KFZKennzeichen, LName, Einwohnerzahl)

Jagdgebiet(JName, KFZKennzeichen[Landkreis], Gesamtfläche)

Jagdschein(JSNR, KFZKennzeichen[Landkreis], gültigVon, gültigBis)

Jäger(JNR, JSNR, Name, Stadt, Straße, Hausnummer)

Wild(WNR, Art, Größe, Gewicht)

Haarwild(WNR, GehörnTyp)

Federwild(WNR, Flügelspannweite)

erlegen(JNR[Jäger], WNR[Wild], JName[Jagdgebiet], KFZKennzeichen[Landkreis])

beschränken(JSNR[Jagdschein], JName[Jagdgebiet], KFZKennzeichen[Landkreis])

Aufgabe 2 [Polizei]

Gehen Sie dabei von dem dazugehörigen relationalen Schema aus:

Polizist : {[PersNr, DSID, Vorname, Nachname, Dienstgrad, Gehalt]}

Dienststelle : {[DSID, Name, Strasse, HausNr, Stadt]}

Fall : {[AkZ, Titel, Beschreibung, Status]}

Arbeitet_An : {[PersNr, AkZ, Von, Bis]}

`Vorgesetzte : {[ PersNr, PersNr, Vorgesetzter ]}`

Gegeben sei folgendes ER-Modell, welches Polizisten, deren Dienststelle und Fälle, an denen sie arbeiten, speichert:

(a) Formulieren Sie eine Anfrage in relationaler Algebra, welche den *Vornamen* und *Nachnamen* von Polizisten zurückgibt, deren Dienstgrad *„Polizeikommissar“* ist und die mehr als 1500 Euro verdienen.

Lösungsvorschlag

$$\pi_{\text{Vorname,Nachname}}(\sigma_{\text{Dienstgrad}=\text{'Polizeikommissar'} \wedge \text{Gehalt}>1500}(\text{Polizist}))$$

(b) Formulieren Sie eine Anfrage in relationaler Algebra, welche die *Titel* der *Fälle* ausgibt, die von *Polizisten* mit dem *Nachnamen „Mayer“* bearbeitet wurden.

Lösungsvorschlag

$$\pi_{\text{Titel}}(\sigma_{\text{Nachname}=\text{'Mayer'}}(\text{Polizist}) \bowtie_{\text{PersNr}} \text{Arbeitet\_An} \bowtie_{\text{AkZ}} \text{Fall})$$

(c) Formulieren Sie eine SQL-Anfrage, welche die Anzahl der Polizisten ausgibt, die in der Stadt *„München“* arbeiten und mit Nachnamen *„Schmidt“* heißen.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl_Polizisten
FROM Polizist p, Dienststelle d
WHERE
  p.DSID = d.DSID AND
  d.Stadt = 'München' AND
  p.Nachname = 'Schmidt';


 anzahl_polizisten
-------------------
                   1
(1 row)
```

(d) Formulieren Sie eine SQL-Anfrage, welche die *Namen* der *Dienststellen* ausgibt, die am *14.02.2012* an dem Fall mit dem *XZ1508* beteiligt waren. Ordnen Sie die Ergebnismenge alphabetisch (aufsteigend) und achten Sie darauf, dass keine Duplikate enthalten sind.

Lösungsvorschlag

```
SELECT DISTINCT d.Name
FROM Dienststelle d, Polizist p, Arbeitet_An a
WHERE
  a.AkZ = 'XZ1508' AND
  p.PersNr = a.PersNr AND
  p.DSID = d.DSID AND
  a.Von <= '2012-02-14' AND
  a.Bis >= '2012-02-14'
ORDER BY d.Name ASC;


              name
-------------------------------
```

```
 Dienststelle Nürnberg (Mitte)
(1 row)
```

(e) Definieren Sie die View *„Erstrebenswerte Dienstgrade“*, welche Dienstgrade enthalten soll, die in *München* mit durchschnittlich mehr als *2500* Euro besoldet werden.

Lösungsvorschlag

```
CREATE VIEW ErstrebenswerteDienstgrade AS (
  SELECT DISTINCT p.Dienstgrad
  FROM Polizist p, Dienststelle d
  WHERE
    p.DSID = d.DSID AND
    d.Stadt = 'München'
  GROUP BY Dienstgrad
  HAVING (AVG(Gehalt) > 2500)
);

SELECT * FROM ErstrebenswerteDienstgrade;


   dienstgrad
-----------------
 Polizeikommisar
 Polizeimeister
(2 rows)
```

(f) Formulieren Sie eine SQL-Anfrage, welche *Vorname*, *Nachname* und *Dienstgrad* von *Polizisten* mit *Vorname*, *Nachname* und *Dienstgrad* ihrer *Vorgesetzten* als ein Ergebnis-Tupel ausgibt (siehe Beispiel-Tabelle). Dabei sind nur *Polizisten* zu selektieren, die an Fällen gearbeitet haben, deren Titel den Ausdruck „Fussball“ beinhalten. An *Vorgesetzte* sind keine Bedingungen gebunden. Achten Sie darauf, dass Sie nicht nur direkte Vorgesetzte, sondern alle Vorgesetzte innerhalb der Vorgesetzten-Hierarchie betrachten. Ordnen Sie ihre Ergebnismenge alphabetisch (absteigend) nach Nachnamen des Polizisten.

Hinweis: Sie dürfen Views verwenden, um Teilergebnisse auszudrücken.

Lösungsvorschlag

Vorarbeiten:

```
  SELECT p.Vorname, p.Nachname
  FROM Polizist p, Arbeitet_An a, Fall f
  WHERE
    p.PersNr = a.PersNr AND
    a.AkZ = f.Akz AND
    f.Titel LIKE '%Fussball%';


 vorname | nachname
---------+----------
 Hans    | Müller
 Josef   | Fischer
(2 rows)
```

Lösungsansatz 1

```
WITH RECURSIVE Fussball_Vorgesetzte (PersNr, VN, NN, DG, PN_VG, VN_VG, NN_VG,
↪  DG_VG) AS
(
  SELECT
    p1.PersNr,
    p1.Vorname AS VN,
    p1.Nachname AS NN,
    p1.Dienstgrad AS DG,
    p2.PersNr AS PN_VG,
    p2.Vorname AS VN_VG,
    p2.Nachname AS NN_VG,
    p2.Dienstgrad AS DG_VG
  FROM Polizist p1, Fall f, Arbeitet_An a, Vorgesetzte v
  LEFT JOIN Polizist p2 ON v.PersNr_Vorgesetzter = p2.PersNr
  WHERE
    p1.PersNr = a.PersNr AND
    a.AkZ = f.Akz AND
    f.Titel LIKE '%Fussball%' AND
    p1.PersNr = v.PersNr

  UNION ALL

  SELECT
    m.PersNr,
    m.VN AS VN,
    m.NN AS NN,
    m.DG AS DG,
    p.PersNr AS PN_VG,
    p.Vorname AS VN_VG,
    p.Nachname AS NN_VG,
    p.Dienstgrad AS DG_VG
  FROM Fussball_Vorgesetzte m, Vorgesetzte v
  LEFT JOIN Polizist p ON v.PersNr_Vorgesetzter = p.PersNr
  WHERE m.PN_VG = v.PersNr
)

SELECT VN, NN, DG, VN_VG, NN_VG, DG_VG
FROM Fussball_Vorgesetzte
ORDER BY NN DESC;
```

```
  vn   |   nn    |         dg          |   vn_vg   |  nn_vg   |        dg_vg
-------+---------+---------------------+-----------+----------+-------------------
 Hans  | Müller  | Polizeimeister      | Andreas   | Schmidt  | Polizeikommisar
 Hans  | Müller  | Polizeimeister      | Stefan    | Hoffmann | Polizeidirektor
 Josef | Fischer | Polizeihauptmeister | Stefan    | Hoffmann | Polizeidirektor
 Josef | Fischer | Polizeihauptmeister | Sebastian | Wagner   | Polizeioberkommis
(4 rows)
```

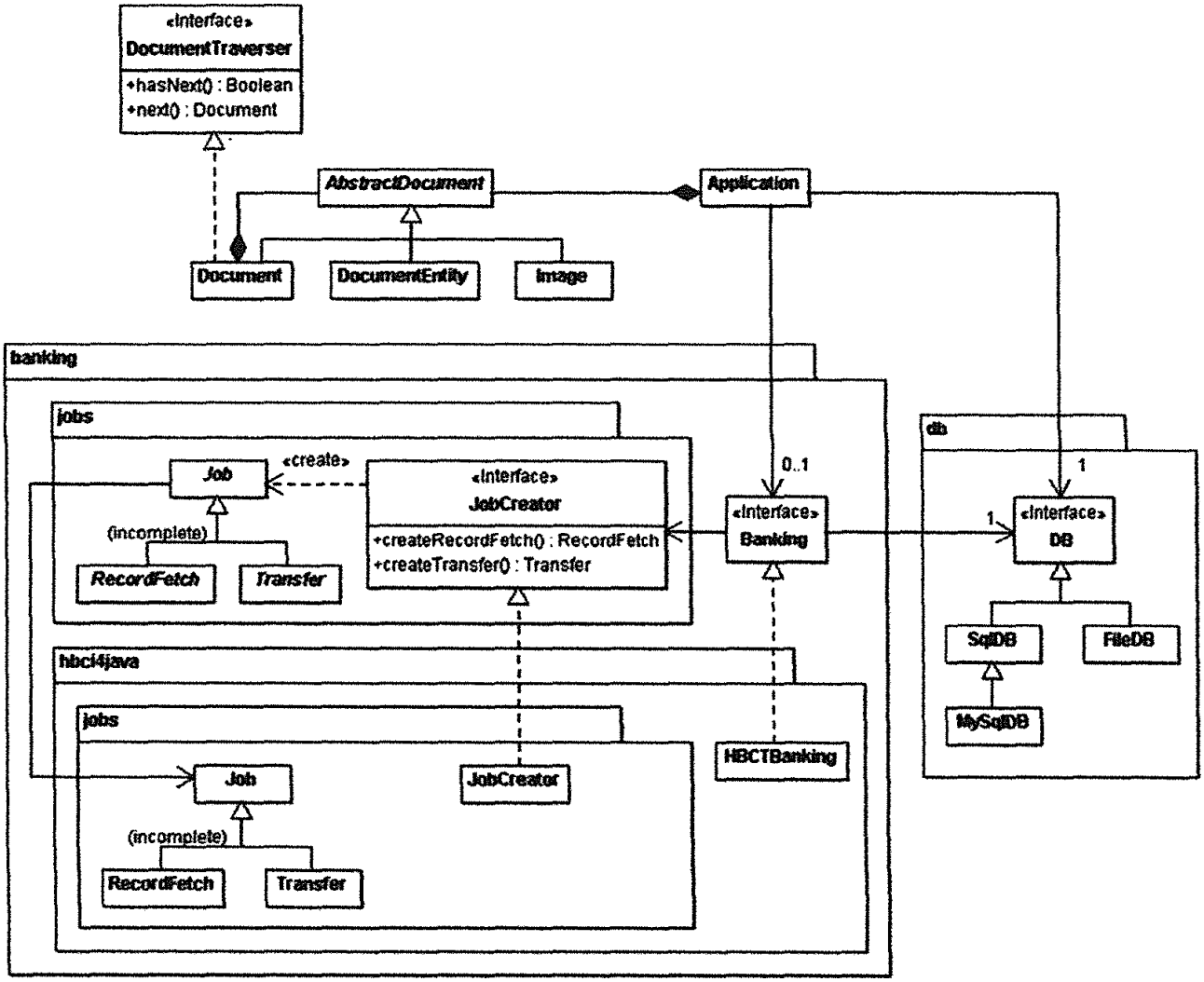
Lösungsansatz 2

```sql
CREATE VIEW naechste_Vorgesetzte AS
  SELECT
    p.PersNR,
    p.Vorname,
    p.Nachname,
    p.Dienstgrad,
    v.PersNr_Vorgesetzter AS Vorgesetzter
  FROM Polizist p LEFT JOIN Vorgesetzte v
  ON p.PersNr = v.PersNr;

WITH RECURSIVE Fussball_Vorgesetzte (VN, NN, DG, VN_VG, NN_VG, DG_VG) AS (
  SELECT
    x.Vorname AS VN,
    x.Nachname AS NN,
    x.Dienstgrad AS DG,
    y.Vorname AS VN_VG,
    y.Nachname AS NN_VG,
    y.Dienstgrad AS DG_VG
  FROM naechste_Vorgesetzte x, Fall f, Arbeitet_An a,
  naechste_Vorgesetzte y
  WHERE
    f.Titel LIKE '%Fussball%' AND
    f.AkZ = a.AkZ AND
    x.PersNr = a.PersNr AND
    x.Vorgesetzter = y.PersNr
  UNION ALL
  SELECT
    a.Vorname AS VN,
    a.Nachname AS NN,
    a.Dienstgrad AS DB,
    Vorname AS VN_VG,
    Nachname AS NN_VG,
    Dienstgrad AS DG_VG
  FROM naechste_Vorgesetzte a INNER JOIN Fussball_Vorgesetzte
  ON a.Vorgesetzter = PersNr
)

SELECT *
FROM Fussball_Vorgesetzte;
```

```
  vn   |   nn    |         dg          |   vn_vg   |  nn_vg   |        dg_vg
-------+---------+---------------------+-----------+----------+-------------------
 Hans  | Müller  | Polizeimeister      | Andreas   | Schmidt  | Polizeikommisar
 Hans  | Müller  | Polizeimeister      | Stefan    | Hoffmann | Polizeidirektor
 Josef | Fischer | Polizeihauptmeister | Stefan    | Hoffmann | Polizeidirektor
 Josef | Fischer | Polizeihauptmeister | Sebastian | Wagner   | Polizeioberkommis
(4 rows)
```

Teilaufgabe Nr. 2

Aufgabe 2 [Entwurfsmuster in UML-Diagramm erkennen]



(a) Kennzeichnen Sie im folgenden Klassendiagramm die Entwurfsmuster *„Abstrakte Fabrik“*, *„Iterator“*, *„Adapter “* und *„Kompositum“*. Geben Sie die jeweils beteiligten Klassen und deren Zuständigkeit im entsprechenden Muster an.

Lösungsvorschlag

Iterator

DocumentTraverser (interface) Schnittstelle zur Traversierung und zum Zugriff auf Dokumente

Document implementiert die Schnittstelle

Kompositum

AbstractDocument abstrakte Basisklasse, die gemeinsames Verhalten der beteiligten Klassen definiert

Document enthält wiederum weitere Documente bzw. DocumentEntities und

Images

DocumentEntity, Image primitive Unterklassen, besitzen keine Kindobjekte

Adapter (Objektadapter)

Banking (interface) vom Client (hier Application) verwendete Schnittstelle

HBCTBanking passt Schnittstelle der unpassenden Klasse an Zielschnittstelle (Banking) an

DB (interface) anzupassende Schnittstelle

abstrakte Fabrik

JobCreator (interface) abstrakte Fabrik

Job (abstrakt) mit Unterklassen RecordFetch und Transfer abstraktes Produkt

Job (konkret) mit Unterklassen konkretes Produkt

(b) (i) Beschreiben Sie die Funktionsweise der folgenden Entwurfsmuster und geben Sie ein passendes UML-Diagramm an.

- Dekorierer
- Klassenadapter
- Objektadapter

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(ii) Erklären Sie mit maximal zwei Sätzen den Unterschied zwischen Klassenadapter und Objektadapter.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.

(c) Implementieren Sie einen Stapel in der Programmiersprache Java. Nutzen Sie dazu ein Array mit fester Größe. Auf eine Überlaufprüfung darf verzichtet werden. Implementieren Sie in der Klasse das Iterator Entwurfsmuster, um auf die Inhalte zuzugreifen, sowie eine Funktion zum Hinzufügen von Elementen. Als Typ für den Stapel kann zur Vereinfachung ein Integertyp verwendet werden.

Lösungsvorschlag

Diese Aufgabe hat noch keine Lösung. Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per

E-Mail an hermine.bschlangaul@gmx.net.

Thema Nr. 2

Teilaufgabe Nr. 2