LU02.A04 - Klassendiagramm zeichnen

Zeichnen Sie ein UML Klassendiagramm anhand eines vorgegebenen Sourcecodes

Auftrag

1. Wählen Sie 2 Codesequenzen aus (siehe unten, z.B. `stack.py`). Übersetzen Sie diese dann in das entsprechende Klassendiagramm.
2. Halten Sie sich an die UML-Notation. Verwenden Sie für das Erstellen der Diagramme ein Tool wie z.B. PlantUML oder MS-Visio.
3. Ergänzen Sie das Diagramm um Initialisierung und/oder Zusicherung. Angaben dazu finden Sie im Merkblatt Klasse.

Dauer

20 Minuten

Abgabe

Exportieren Sie Ihre Diagramme in ein PDF-Dokument und geben Sie dieses dann im Moodle-Kurs ab.

Codesequenzen

stack.py

```
class Stack:

    def __int__(self):
        self._items = []

    def is_empty(self) -> bool:
        pass

    def push(self, item: object) -> None:
        pass

    def pop(self) -> object:
        pass
```

sudoku.py

```
import math


class Board:

    def __init__(self, data: list):
        self._data = data
        self._rows = len(data)
        self._cols = len(data[0])
        self._size = self.rows
        self._block_size = int(math.sqrt(self.size))

    @property
    def rows(self) -> list:
        return self._rows

    @property
    def cols(self) -> list:
        return self._cols

    @property
    def size(self) -> int:
        return self._size

    @property
    def block_size(self) -> int:
        return self._block_size

    def get_col(self, x: int) -> list:
        pass

    def get_row(self, y: int) -> list:
        pass

    def get_box(self, b: int) -> list:
        pass

    def rows_have_correct_size(self) -> bool:
        pass
```

bicycle.py

```
class Bicycle:
    """
    Die Klasse stellt eine vereinfachte Beschreibung eines Fahrrads dar.
    Es werden nur die Attribute color, seats, gear,
```

```
    """

    def __init__(self, color, seats = 1):
        """
        Initialisiert das Fahrrad mit seiner Farbe und der Anzahl der sitze.
        Per default ist der Wert auf 1 festgelegt.
        :param color:
        :return:
        """
        self.seats = seats
        self._color = color

    def accelerate(self):
        """
        diese Methode beschleunigt das Fahrrad
        """
        pass

    def slow_down(self):
        """
        Diese Methode bremst das Fahrrad
        """
        pass

    def drive(self):
        """
        Diese Methode bewegt das Fahrrad mit der aktuellen Geschwindigkeit
        """
        pass

    @property
    def color(self):
        """
        liefert die Farbe des Fahrrades
        :return:
        """
        return self._color

    @property
    def seats(self):
        """
        liefert die Anzahl der Sitze des Fahrrades
        :return:
        """
        return self._seats

    @seats.setter
    def seats(self, value):
        """
        Setzt die Anzahl der Sitze des Fahrrades. Der Wert kann nur 1 oder 2
(Tandem) sein.
```

```
        :return:
        """
        if value >= 1 and value <= 2:
            self._seats = value
```

grade.py

```
class Grade:
    """
    Eine Zeugnisnote beschreibt eine Leistung mit Werten von 1 (schlecht)
bis 6 (sehr gut).
    Die Werte werden i.d.R. als Ganz- und Halbnoten festgehalten, also z.B.
4.0 bzw. 4.5
    Der Notenwert wird im Konstruktor gesetzt, kann aber mittels der
setMethode angepasst
    werden. Dabei muss der Notenbereich (1..6) zugesichert werden.
    Über getNote kann der Notenwert ausgelesen werden.
    """

    def __init__(self, new_score):
        """
        Die Note wird mit einem Wert initialisiert
        :param score: ein Wert im Bereich 1.0 .... 6.0
        """
        self.score(new_score)

    @property
    def score(self):
        """
        Liefert den Notenwert
        :return: Notenwert
        """
        return self._score

    @score.setter
    def score(self, score):
        """
        Legt einen neuen Notenwert fest.
        Der Wert muss im Bereich 1.0 .... 6.0 liegen. Ist dies nicht der
Fall,
        liefert die Methoden den Wert False (Zuweisung nicht erfolgreich).
        :param score:  ein Wert im Bereich 1.0 .... 6.0
        :return: True bei erfolgreicher Zuweisung, sonst False
        """
        if (score >= 1.0) and (score <= 6.0):
            self._score = score
            return True
        else:
```

```
            return False

    def print(self):
        print("Grade : " + str(self._score))
```

---

m320-LU02

© René Probst

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m320_2024/learningunits/lu02/aufgaben/zeichnen?rev=1788091536

Last update: **2026/08/30 14:05**