

**T.C**

**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ**

**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**

**BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ**

**PROGRAM DERS İLİŞKİ MATRİSİ PROJESİ**

**HAZIRLAYANLAR:**

**Müge ÖZLER Yavuz Selim GÜRSOY 210501004 220501003**

**Emrah ŞAHİN Talha TUNA**

**220502025 220502015**

**DERS SORUMLUSU:**

**DR. ÖĞR. ÜYESİ**

**Elif Pınar HACIBEYOĞLU**

**TARİH:22.11.2024**

**1 GİRİŞ**

**1.1 Projenin Amacı**

Bu projenin amacı, eğitim kurumları için öğrenci ve ders yönetimi yazılımı geliştirmektir. Yazılım, öğretmenlerin ve yöneticilerin ders bilgilerini, öğrenci kayıtlarını ve notları etkili bir şekilde yönetmelerine olanak tanıyacaktır. Kullanıcılar, ders bilgilerini girebilir, güncelleyebilir, notları kaydedebilir ve öğrenci başarılarını hesaplayabilir. Proje, eğitim kurumlarının yönetim süreçlerini kolaylaştırmayı hedeflemektedir.

**2 GEREKSİNİM ANALİZİ**

**2.1 Arayüz Gereksinimleri**

**Kullanıcı Dostu Arayüz:** Sistem, öğrencilerin ve öğretim görevlilerin kolayca veri yüklemesine ve analizleri incelemesine olanak tanıyan, anlaşılabilir bir arayüz sunmalıdır.

**Tablo Görünümü:** Öğrenci notları, program çıktıları ve ders çıktıları arasındaki ilişkilerin tablo olarak görüntülemesi sağlanmalıdır.

**Dosya Yükleme ve Kaydetme:** Excel dosyalarını yüklemek ve analiz sonuçlarını tekrar Excel formatında dışa aktarabilmek için dosa yükleme ve kaydetme seçenekleri sunulmalıdır.

**Raporlama Modülü:** Sistem, Öğrencilerin başarı oranlarını grafik ve raporlarla göstermelidir.

**Çoklu Kullanıcı Erişimi:** Farklı derslere ait verilerin tek bir sistem üzerinde ayrı olarak yönetilmesine olanak tanıyan bir yapı.

**2.2 Fonksiyonel Gereksinimler**

**Veri Yükleme:**

* Kullanıcıların ders çıktıları, program çıktıları ve öğrenci notları için Excel dosyaları yükleyebilmesi.

**Veri Doğrulanma:**

* Yüklenen tablolarda eksik veya hatalı veri olup olmadığını kontrol etme.
* Değerlerin uygun aralıkta olup olmadığını doğrulama (0-1 arası veya 0-100 arası gibi)

**Analiz:**

* Ders çıktıları ile değerlendirme kriterleri arasındaki ağırlıklı ilişkiyi hesaplama
* Program çıktıları üzerinden öğrencilerin başarı oranlarını hesaplama

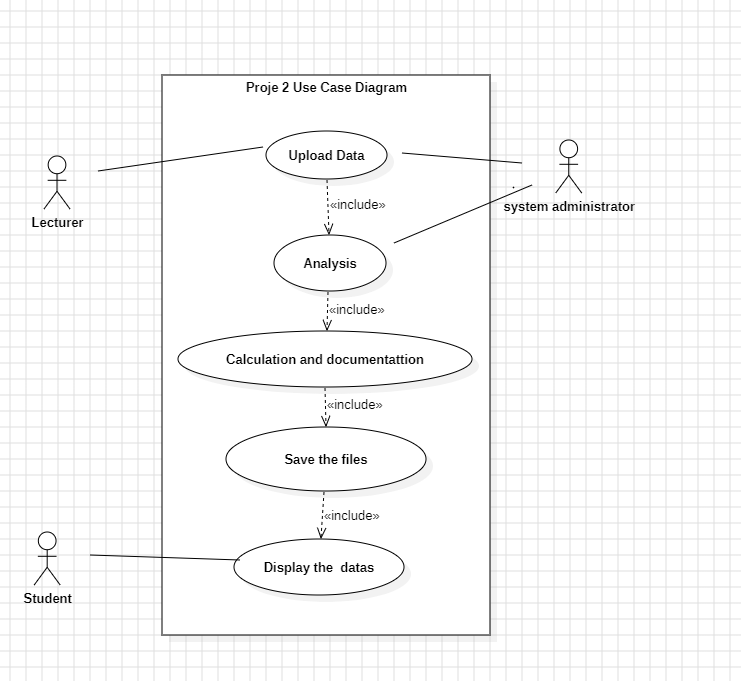
**Tablo Oluşturma:**

* Ders çıktıları ve değerlendirme kriterleri arasındaki ilişki matrisinden kaynaklanan Tablo3, Tablo4, Tablo5 oluşturma.

**Raporlama ve Dışa Aktarım:**

* + Her öğrenci için detaylı rapor oluşturma
  + Oluşan raporları ve Excel dosyası olarak kaydetme

**2.3 Use-Case Diyagramı**



**3 TASARIM**

**3.1 Mimari Tasarım**

Sistem, üç ana katmandan oluşacak şekilde tasarlanacaktır:

* **Veri Katmanı:**
  + Excel dosyalarından gelen verilerin okunması, doğrulanması ve analiz sonuçlarının kaydedilmesini yönetir.
  + Kullanılacak Teknolojiler: Pandas, OpenPyXL.
* **Uygulama Katmanı:**
  + Veri işleme, hesaplama ve raporlama işlemlerinin gerçekleştirildiği katmandır.
  + Kullanılacak Teknolojiler: Python, Flask/Django (API için).
* **Sunum Katmanı:**
  + Kullanıcı arayüzü, grafiksel raporlar ve veri yükleme/sunma işlemlerinin gerçekleştirildiği katmandır.
  + Kullanılacak Teknolojiler: HTML, CSS, JavaScript (React veya Vue.js).

**3.2 Kullanılacak Teknolojiler**

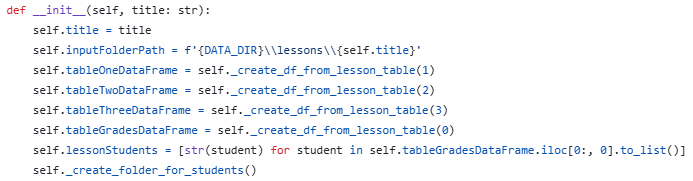
1. **Programlama Dili:** Python (veri işleme ve analiz için).
2. **Veritabanı:** SQLite veya PostgreSQL (eğer sistem, kalıcı veritabanı desteği gerektirirse).
3. **Kütüphaneler:**
   * Veri işleme: Pandas, NumPy
   * Dosya işlemleri: OpenPyXL, PyPDF2
   * Grafik ve Raporlama: Matplotlib, Seaborn, Plotly
4. **Frontend Teknolojileri:**
   * HTML, CSS (Bootstrap ile)
   * JavaScript Frameworkü: React veya Vue.js
5. **Backend Teknolojileri:** Flask veya Django (API ve veri işleme için).
6. **Test Araçları:**
   * Unit Test: Pytest
   * Entegrasyon Testleri: Postman
7. **Versiyon Kontrolü:** Git ve GitHub.

**4 UYGULAMA**

**4.1 Kodlanan Bileşenlerin Açıklamaları**

## 4.1.1 Class LESSON:

**def \_\_init\_\_(self, title: str):**

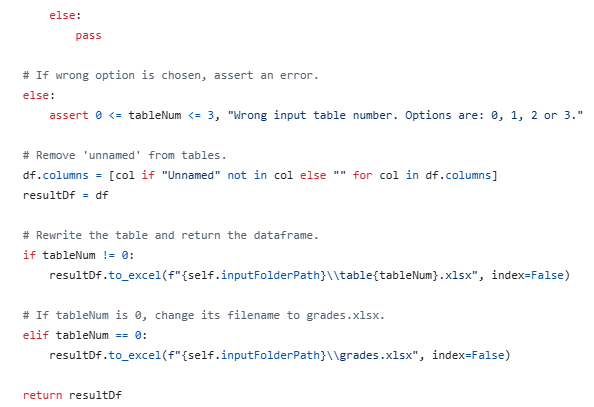
 Bu kod, Lesson sınıfının bir örneği oluşturulduğunda çalışan kurucu fonksiyondur. **self.title = title**: Dersin başlığını sınıfın title özelliğine atar. **self.inputFolderPath = f'{DATA\_DIR}\\lessons\\{self.title}'**: Dersle ilgili verilerin bulunduğu klasör yolunu oluşturur ve inputFolderPath değişkenine atar. **self.tableOneDataFrame = self.\_create\_df\_from\_lesson\_table(1)**, **self.tableTwoDataFrame = self.\_create\_df\_from\_lesson\_table(2)**, **self.tableThreeDataFrame = self.\_create\_df\_from\_lesson\_table(3)**, **self.tableGradesDataFrame = self.\_create\_df\_from\_lesson\_table(0)**: Dersle ilgili farklı veri tablolarını oluşturur ve sınıfın ilgili özelliklerine atar. **self.lessonStudents = [str(student) for student in self.tableGradesDataFrame.iloc[0:, 0].to\_list()]**: Dersin öğrenci listesini, notlar tablosundaki ilk sütundan alarak bir listeye dönüştürür ve lessonStudents özelliğine atar. **self.\_create\_folder\_for\_students()**: Dersin öğrencileri için birer klasör oluşturur.

**def \_create\_df\_from\_lesson\_table(self, tableNum: int) -> pd.DataFrame:**

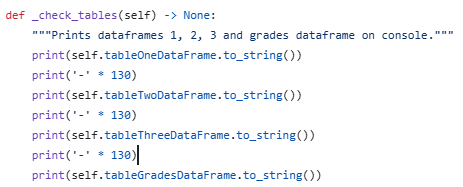
Bu kod, dersle ilişkili belirli bir tablodan bir DataFrame oluşturur ve işlemler yapar. tableNum parametresi, hangi tablonun işleneceğini belirler. tableNum == 1 olduğunda, table1.xlsx okunur, değerlerin 0 ile 1 arasında olup olmadığı kontrol edilir, ardından her satırın toplam ilişkisi İlişki Değeri sütununda hesaplanır. tableNum == 2 olduğunda, table2.xlsx okunur, sütun sayısının en az 3 olması ve ağırlıkların toplamının 100 olması kontrol edilir, ardından TOPLAM sütunu eklenir. tableNum == 3 olduğunda, table2.xlsx verileri yeniden hesaplanarak table3 oluşturulur, gerekli sütunlar eklenir ve TOPLAM sütunu hesaplanır. tableNum == 0 olduğunda, grades.xlsx okunur, ağırlıklı ortalamalar hesaplanır ve ORT sütunu eklenir. Yanlış bir tableNum değeri verilirse hata oluşturulur. unnamed sütun adları temizlenir, tablo uygun adla tekrar yazılır ve bir DataFrame olarak döndürülür.

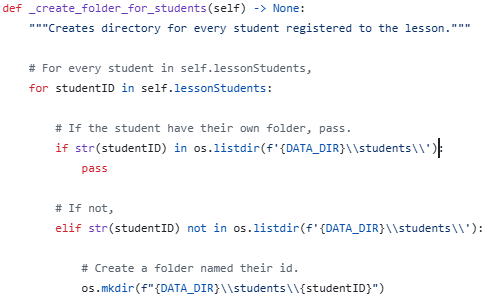




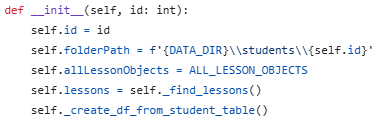


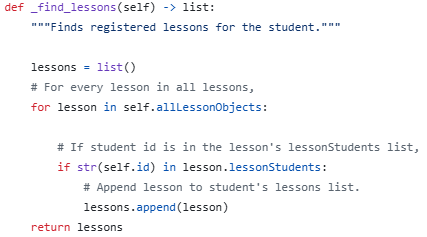
**def \_check\_tables(self) -> None:**

Bu fonksiyon, Lesson sınıfındaki tablo verilerini konsola yazdırır. **print(self.tableOneDataFrame.to\_string())** ile birinci tablo (table1) detaylı bir şekilde ekrana yazdırılır. **print('-' \* 130)** tablo yazımından sonra 130 karakter uzunluğunda bir ayırıcı çizgi çizer. Bu işlem sırasıyla ikinci tablo (table2), üçüncü tablo (table3) ve notlar tablosu (grades) için tekrarlanır. Amaç, tüm tabloların düzenli bir şekilde konsolda görüntülenmesini sağlamaktır.

**def \_create\_folder\_for\_students(self) -> None:**Bu fonksiyon, derse kayıtlı her öğrenci için bir klasör oluşturur. for studentID in self.lessonStudents ile dersin öğrenci listesi döngüye alınır. if str(studentID) in os.listdir(f'{DATA\_DIR}\\students\\') ifadesi, öğrencinin klasörünün mevcut olup olmadığını kontrol eder. Eğer klasör varsa hiçbir işlem yapılmaz (pass). elif str(studentID) not in os.listdir(f'{DATA\_DIR}\\students\\') kontrolü ile klasör mevcut değilse, os.mkdir(f"{DATA\_DIR}\\students\\{studentID}") kullanılarak öğrencinin kimlik numarasıyla bir klasör oluşturulur. Amaç, her öğrencinin dersle ilgili dosyalarını saklayabileceği bir dizin yapısını otomatik olarak oluşturmaktır.

## Class Student:

**def \_\_init\_\_(self, id: int):**Bu kod, Student sınıfının bir örneği oluşturulduğunda çalışan kurucu fonksiyondur. **self.id = id** ifadesi, öğrencinin kimlik numarasını id özelliğine atar. **self.folderPath = f'{DATA\_DIR}\\students\\{self.id}'** ile öğrencinin dosyalarının bulunacağı klasör yolunu oluşturur. **self.allLessonObjects = ALL\_LESSON\_OBJECTS** ifadesi, tüm ders nesnelerini allLessonObjects özelliğine atar. **self.lessons = self.\_find\_lessons()** ile öğrencinin kayıtlı olduğu dersleri bulur ve lessons özelliğine atar. Son olarak, **self.\_create\_df\_from\_student\_table()** çağrısı, öğrencinin dersleriyle ilgili tablo dosyalarını oluşturur. Amaç, öğrenciye ait tüm bilgileri ve ilgili ders verilerini kurulum sırasında hazır hale getirmektir.

**def \_find\_lessons(self) -> list:**Bu fonksiyon, öğrencinin kayıtlı olduğu dersleri bulur. lessons = list() ile boş bir liste oluşturulur. for lesson in self.allLessonObjects döngüsü, tüm ders nesnelerini (tüm dersler ALL\_LESSON\_OBJECTS'da bulunur) teker teker inceler. if str(self.id) in lesson.lessonStudents kontrolü, öğrencinin kimlik numarasının ilgili dersin öğrenci listesinde olup olmadığını kontrol eder. Eğer öğrenci derse kayıtlıysa, lessons.append(lesson) ile bu ders nesnesi öğrencinin dersler listesine eklenir. Döngü tamamlandığında, öğrenciye ait kayıtlı derslerin bir listesini döndürür. Amaç, öğrencinin kayıtlı olduğu dersleri hızlı bir şekilde belirlemek ve işlemekte kullanmaktır.

**def \_create\_df\_from\_student\_table(self) -> None:**

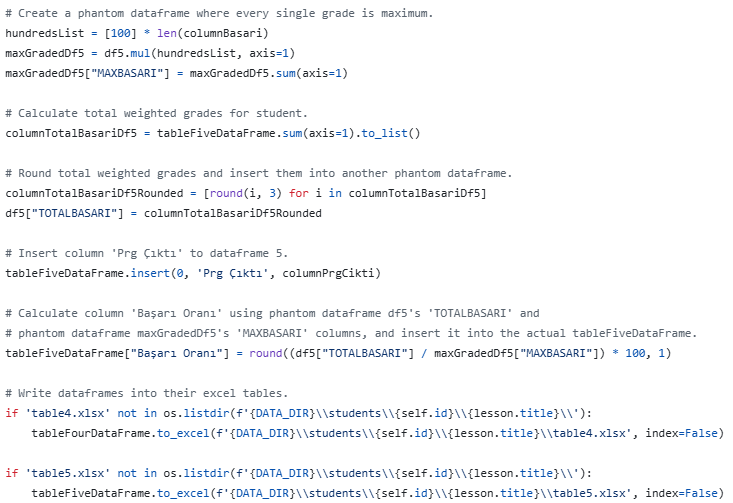
Bu fonksiyon, bir öğrenci için ilgili derslerin veri tablolarını oluşturur ve işlemler yapar. **for lesson in self.lessons** döngüsü ile öğrencinin kayıtlı olduğu her ders üzerinde işlem yapılır. Eğer dersin klasörü öğrencinin klasöründe yoksa, **os.mkdir(f"{DATA\_DIR}\\students\\{self.id}\\{lesson.title}")** ile o ders için bir klasör oluşturulur. **df3 ve dfGrades** değişkenleri sırasıyla dersin üçüncü tablosu ve notlar tablosu olarak alınır. **recalculatedDf3** ve **recalculatedDfGrades**, bu tablolardan sadece veri değerlerini almak için kırpılmış hallerdir. **columnDersCikti** ve **columnToplamDf3**, üçüncü tablodan sırasıyla "Ders Çıktı" ve "TOPLAM" sütunlarının değerlerini alır. **columnMaxDf4**, bu toplam değerlerin 100 ile çarpılmasıyla elde edilir.

**df4**, üçüncü tablonun değerleriyle oluşturulan bir yeni veri çerçevesidir. Döngü içinde, öğrencinin kimlik numarasına uygun satır bulunur ve **studentGrades** listesine öğrencinin notları alınır. **tableFourDataFrame**, üçüncü tablodaki değerlerin öğrenci notlarıyla çarpılmasıyla oluşturulur ve bu tabloya "TOPLAM", "MAX", "Başarı" ve "Ders Çıktı" sütunları eklenir.

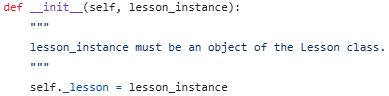
Daha sonra birinci tablo kırpılarak **df1**, öğrenci başarısını içeren sütunla birleştirilir ve **df5** oluşturulur. **tableFiveDataFrame**, bu tablonun öğrenci başarısıyla çarpılmasıyla elde edilir. Maksimum başarı değerleri hesaplanır ve "Başarı Oranı" sütunu eklenir. Son olarak, **table4.xlsx** ve **table5.xlsx** dosyaları oluşturularak öğrenci klasörüne kaydedilir. Bu fonksiyon, öğrencinin performansını derslere göre analiz eden ve ilgili tabloları dosya sistemine yazan bir yapı sağlar.





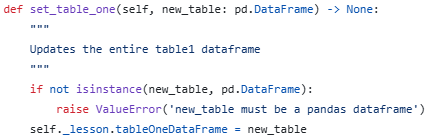


**Class LessonDataMutator:**

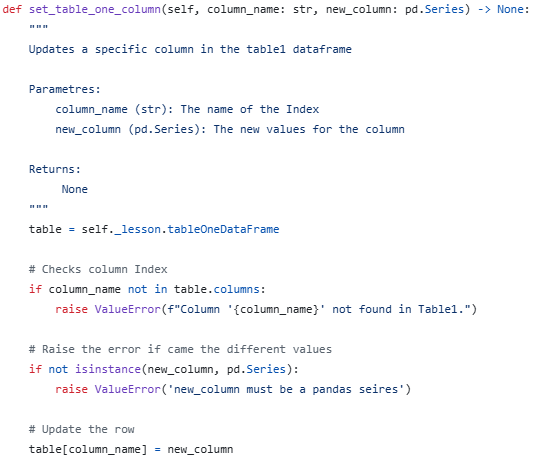
**def \_\_init\_\_(self, lesson\_instance):**

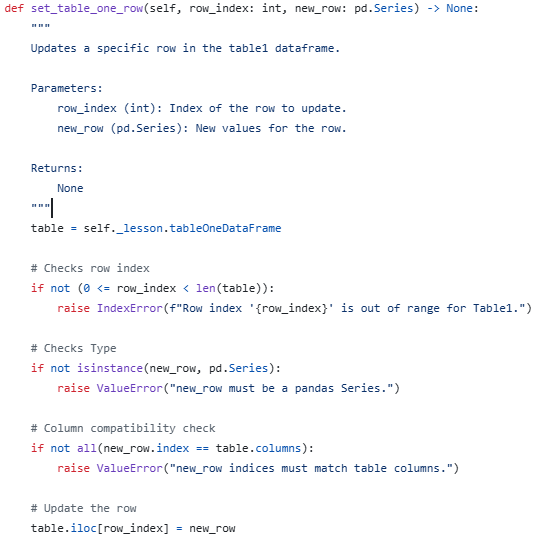
Bu kod, LessonDataMutator veya LessonDataAccessor gibi bir sınıfın kurucu fonksiyonudur ve bir Lesson nesnesini alıp, sınıfın bir özelliği olarak saklar. **self.\_lesson = lesson\_instance** ifadesi, lesson\_instance parametresiyle verilen Lesson sınıfı nesnesini \_lesson adlı bir özel özelliğe atar. Bu sayede, bu sınıf içerisindeki diğer metotlar, ilgili Lesson nesnesine erişebilir ve onunla işlem yapabilir. Bu yapı, Lesson nesnesi üzerindeki veri manipülasyonlarını veya erişimleri düzenli bir şekilde yapmak için bir temel oluşturur.

**def set\_table\_one(self, new\_table: pd.DataFrame) -> None:**

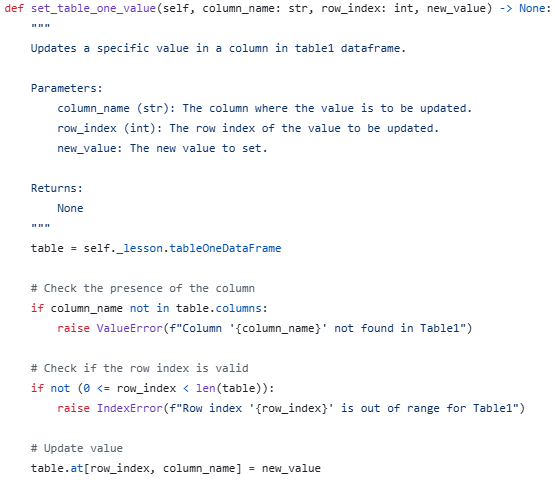
Bu fonksiyon, Lesson nesnesinin birinci tablosunu (tableOneDataFrame) günceller. if not isinstance(new\_table, pd.DataFrame) ifadesi, verilen new\_table parametresinin bir pandas DataFrame olup olmadığını kontrol eder. Eğer değilse, raise ValueError('new\_table must be a pandas dataframe') ile hata fırlatır. Eğer doğru formatta bir DataFrame verilmişse, self.\_lesson.tableOneDataFrame = new\_table ifadesi ile bu yeni tablo, Lesson nesnesindeki tableOneDataFrame özelliğine atanır. Bu fonksiyon, table1 verilerinin tamamen yenilenmesini sağlar.

**def set\_table\_one\_column(self, column\_name: str, new\_column: pd.Series) -> None:**

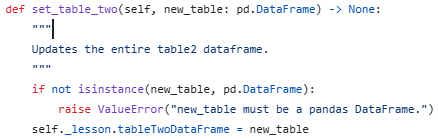
Bu fonksiyon, Lesson nesnesinin birinci tablosundaki (tableOneDataFrame) belirli bir sütunu günceller. **table = self.\_lesson.tableOneDataFrame** ifadesi, birinci tabloyu table değişkenine atar. **if column\_name not in table.columns** kontrolü, güncellenmek istenen sütunun tablodaki mevcut sütunlar arasında olup olmadığını kontrol eder; eğer sütun yoksa **raise ValueError(f"Column '{column\_name}' not found in Table1.")** ile hata fırlatır. **if not isinstance(new\_column, pd.Series)** kontrolü, yeni sütunun bir pandas Series olup olmadığını kontrol eder; değilse **raise ValueError('new\_column must be a pandas series')** ile hata oluşturur. Eğer tüm kontroller geçilirse, **table[column\_name] = new\_column** ifadesiyle belirtilen sütun, yeni sütun değerleriyle güncellenir. Bu fonksiyon, tablo üzerinde seçili bir sütunu güvenli bir şekilde güncellemek için tasarlanmıştır.

**def set\_table\_one\_row(self, row\_index: int, new\_row: pd.Series) -> None:**Bu fonksiyon, Lesson nesnesinin birinci tablosundaki (tableOneDataFrame) belirli bir satırı günceller. table = self.\_lesson.tableOneDataFrame ifadesiyle mevcut tablo alınır. if not (0 <= row\_index < len(table)) kontrolü, güncellenmek istenen satırın tablo içinde geçerli bir indeks olup olmadığını kontrol eder; eğer değilse raise IndexError(f"Row index '{row\_index}' is out of range for Table1.") ile hata fırlatır. if not isinstance(new\_row, pd.Series) kontrolü, yeni satırın bir pandas Series olup olmadığını doğrular; değilse raise ValueError("new\_row must be a pandas Series.") ile hata oluşturur. if not all(new\_row.index == table.columns) kontrolü, yeni satırın sütun başlıklarıyla uyumlu olup olmadığını kontrol eder; değilse raise ValueError("new\_row indices must match table columns.") ile hata verir. Tüm kontroller geçilirse, table.iloc[row\_index] = new\_row ifadesiyle belirtilen satır, yeni değerlerle güncellenir. Bu fonksiyon, tablo üzerindeki belirli bir satırı güvenli ve doğru bir şekilde değiştirmek için tasarlanmıştır.

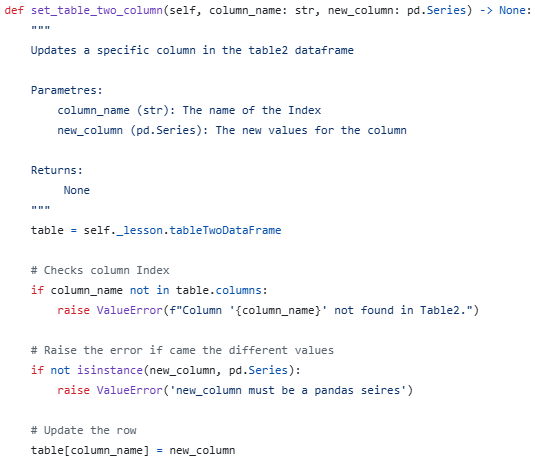
**def set\_table\_one\_value(self, column\_name: str, row\_index: int, new\_value) -> None:**

Bu fonksiyon, Lesson nesnesinin birinci tablosundaki (tableOneDataFrame) belirli bir sütunun belirli bir satırındaki değeri günceller. table = self.\_lesson.tableOneDataFrame ifadesiyle tablo referansı alınır. if column\_name not in table.columns kontrolü, güncellenmek istenen sütunun tablodaki mevcut sütunlar arasında olup olmadığını kontrol eder; eğer değilse raise ValueError(f"Column '{column\_name}' not found in Table1") ile hata fırlatır. if not (0 <= row\_index < len(table)) kontrolü, satır indeksinin geçerli olup olmadığını doğrular; eğer değilse raise IndexError(f"Row index '{row\_index}' is out of range for Table1") ile hata oluşturur. Eğer tüm kontroller geçilirse, table.at[row\_index, column\_name] = new\_value ifadesiyle belirtilen satır ve sütundaki hücre, verilen yeni değerle güncellenir. Bu fonksiyon, bir hücredeki değeri hızlı ve güvenli bir şekilde değiştirmek için tasarlanmıştır.

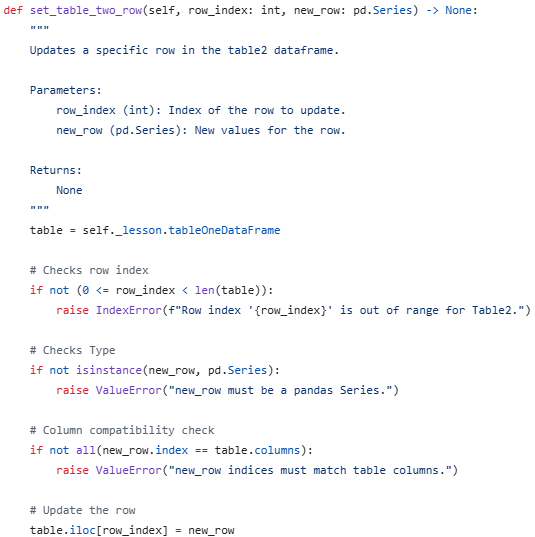
**def set\_table\_two(self, new\_table: pd.DataFrame) -> None:**

Bu fonksiyon, Lesson nesnesinin ikinci tablosunu (tableTwoDataFrame) günceller. if not isinstance(new\_table, pd.DataFrame) ifadesi, verilen new\_table parametresinin bir pandas DataFrame olup olmadığını kontrol eder; eğer değilse, raise ValueError("new\_table must be a pandas DataFrame.") ile hata fırlatır. Eğer uygun bir DataFrame verilmişse, self.\_lesson.tableTwoDataFrame = new\_table ifadesiyle yeni tablo, Lesson nesnesinin ikinci tablo özelliğine atanır. Bu fonksiyon, tablo 2’nin tüm içeriğini güvenli bir şekilde güncellemeyi sağlar.

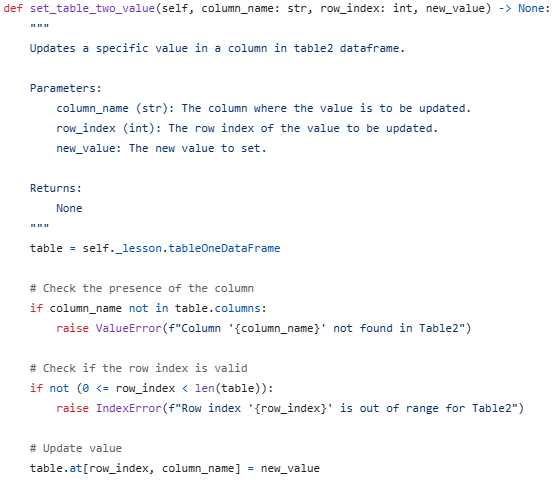
**def set\_table\_two\_column(self, column\_name: str, new\_column: pd.Series) -> None:**

Bu fonksiyon, Lesson nesnesinin ikinci tablosundaki (tableTwoDataFrame) belirli bir sütunu günceller. table = self.\_lesson.tableTwoDataFrame ifadesiyle ikinci tablo referansı alınır. if column\_name not in table.columns kontrolü, güncellenmek istenen sütunun mevcut olup olmadığını doğrular; eğer sütun yoksa, raise ValueError(f"Column '{column\_name}' not found in Table2.") ile hata fırlatır. if not isinstance(new\_column, pd.Series) kontrolü, yeni sütunun bir pandas Series olup olmadığını kontrol eder; değilse, raise ValueError('new\_column must be a pandas series') ile hata oluşturur. Eğer kontroller geçilirse, table[column\_name] = new\_column ifadesiyle belirtilen sütun, new\_column ile güncellenir. Bu fonksiyon, tablo 2'de belirli bir sütunu güvenli ve doğru bir şekilde değiştirmek için tasarlanmıştır.

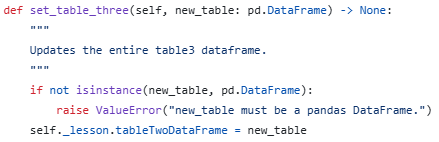
**def set\_table\_two\_row(self, row\_index: int, new\_row: pd.Series) -> None:**

Bu fonksiyon, Lesson nesnesinin ikinci tablosundaki (tableTwoDataFrame) belirli bir satırı günceller. table = self.\_lesson.tableOneDataFrame ifadesi yanlışlıkla birinci tabloyu (tableOneDataFrame) referans almış, doğru tablo referansı tableTwoDataFrame olmalıdır. if not (0 <= row\_index < len(table)) kontrolü, güncellenmek istenen satırın geçerli bir indeks olup olmadığını doğrular; eğer değilse, raise IndexError(f"Row index '{row\_index}' is out of range for Table2.") ile hata fırlatır. if not isinstance(new\_row, pd.Series) kontrolü, yeni satırın bir pandas Series olup olmadığını kontrol eder; değilse, raise ValueError("new\_row must be a pandas Series.") ile hata oluşturur. if not all(new\_row.index == table.columns) kontrolü, yeni satırın sütun başlıklarıyla uyumlu olup olmadığını kontrol eder; değilse, raise ValueError("new\_row indices must match table columns.") ile hata verir. Tüm kontroller geçilirse, table.iloc[row\_index] = new\_row ifadesiyle belirtilen satır, yeni değerlerle güncellenir. Bu fonksiyon, tablo 2'de belirli bir satırı güvenli ve doğru bir şekilde güncellemek için tasarlanmıştır.

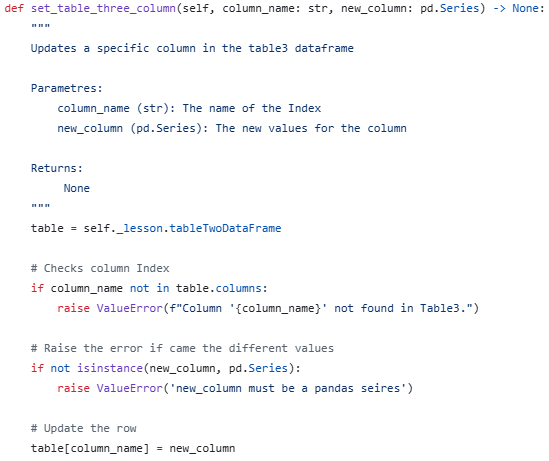
**def set\_table\_two\_value(self, column\_name: str, row\_index: int, new\_value) -> None:**

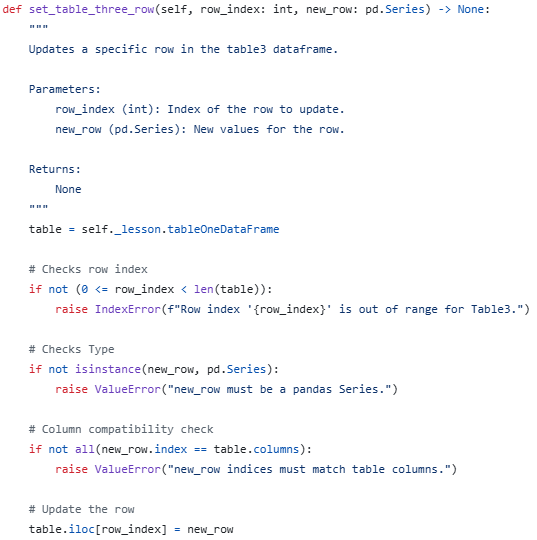
Bu fonksiyon, Lesson nesnesinin ikinci tablosundaki (tableTwoDataFrame) belirli bir sütunun, belirli bir satırındaki değeri günceller. table = self.\_lesson.tableOneDataFrame ifadesi yanlışlıkla birinci tabloyu referans almış, doğru tablo tableTwoDataFrame olmalıdır. if column\_name not in table.columns kontrolü, güncellenmek istenen sütunun mevcut olup olmadığını kontrol eder; eğer yoksa, raise ValueError(f"Column '{column\_name}' not found in Table2") ile hata oluşturur. if not (0 <= row\_index < len(table)) kontrolü, satır indeksinin geçerli olupolmadığını doğrular; eğer geçersizse, raise IndexError(f"Row index '{row\_index}' is out of range for Table2") ile hata fırlatır. Eğer tüm kontroller geçilirse, table.at[row\_index, column\_name] = new\_value ifadesi ile belirtilen hücredeki değer, verilen yeni değerle güncellenir. Bu fonksiyon, tablo 2’de belirli bir hücre değerini güvenli bir şekilde değiştirmek için tasarlanmıştır.

**def set\_table\_three(self, new\_table: pd.DataFrame) -> None:**

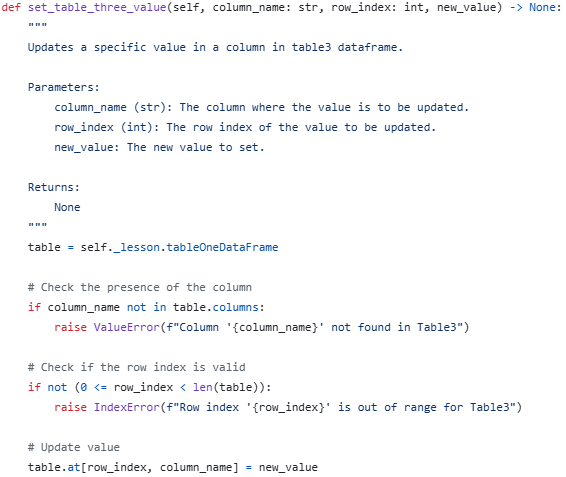
Bu fonksiyon, Lesson nesnesinin üçüncü tablosunu (tableThreeDataFrame) günceller. if not isinstance(new\_table, pd.DataFrame) kontrolü, verilen new\_table parametresinin bir pandas DataFrame olup olmadığını kontrol eder; eğer değilse, raise ValueError("new\_table must be a pandas DataFrame.") ile hata fırlatır. Eğer uygun bir DataFrame verilmişse, self.\_lesson.tableTwoDataFrame = new\_table ifadesiyle yeni tablo, yanlışlıkla ikinci tablo özelliğine atanmıştır. Doğru atama self.\_lesson.tableThreeDataFrame = new\_table olmalıdır. Bu fonksiyon, tablo 3'ün tüm içeriğini güvenli bir şekilde değiştirmek için tasarlanmıştır ancak koddaki hata düzeltilmelidir.

**def set\_table\_three\_column(self, column\_name: str, new\_column: pd.Series) -> None:**

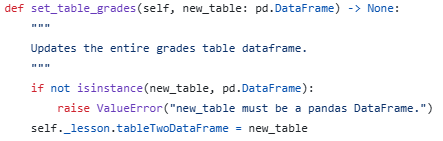
Bu fonksiyon, Lesson nesnesinin üçüncü tablosundaki (tableThreeDataFrame) belirli bir sütunu günceller. table = self.\_lesson.tableTwoDataFrame ifadesi yanlışlıkla ikinci tabloyu (tableTwoDataFrame) referans almış, doğru referans tableThreeDataFrame olmalıdır. if column\_name not in table.columns kontrolü, güncellenmek istenen sütunun mevcut olup olmadığını doğrular; eğer yoksa, raise ValueError(f"Column '{column\_name}' not found in Table3.") ile hata fırlatır. if not isinstance(new\_column, pd.Series) kontrolü, yeni sütunun bir pandas Series olup olmadığını kontrol eder; değilse, raise ValueError('new\_column must be a pandas series') ile hata oluşturur. Eğer kontroller geçilirse, table[column\_name] = new\_column ifadesiyle belirtilen sütun, new\_column ile güncellenir. Bu fonksiyon, tablo 3’te belirli bir sütunu güvenli ve doğru bir şekilde değiştirmek için tasarlanmıştır, ancak koddaki tablo referans hatası düzeltilmelidir. **def set\_table\_three\_row(self, row\_index: int, new\_row: pd.Series) -> None:**

Bu fonksiyon, Lesson nesnesinin üçüncü tablosundaki (tableThreeDataFrame) belirli bir satırı günceller. table = self.\_lesson.tableOneDataFrame ifadesi yanlışlıkla birinci tabloyu referans almış, doğru referans tableThreeDataFrame olmalıdır. if not (0 <= row\_index < len(table)) kontrolü, güncellenmek istenen satırın geçerli bir indeks olup olmadığını doğrular; eğer geçersizse, raise IndexError(f"Row index '{row\_index}' is out of range for Table3.") ile hata oluşturur. if not isinstance(new\_row, pd.Series) kontrolü, yeni satırın bir pandas Series olup olmadığını kontrol eder; değilse, raise ValueError("new\_row must be a pandas Series.") ile hata fırlatır. if not all(new\_row.index == table.columns) kontrolü, yeni satırın sütun başlıklarıyla uyumlu olup olmadığını doğrular; uyumsuzsa, raise ValueError("new\_row indices must match table columns.") ile hata oluşturur. Tüm kontroller geçilirse, table.iloc[row\_index] = new\_row ifadesiyle belirtilen satır, yeni değerlerle güncellenir. Bu fonksiyon, tablo 3’te belirli bir satırı güvenli ve doğru bir şekilde güncellemek için tasarlanmıştır, ancak koddaki tablo referans hatası düzeltilmelidir.  
**def set\_table\_three\_value(self, column\_name: str, row\_index: int,**

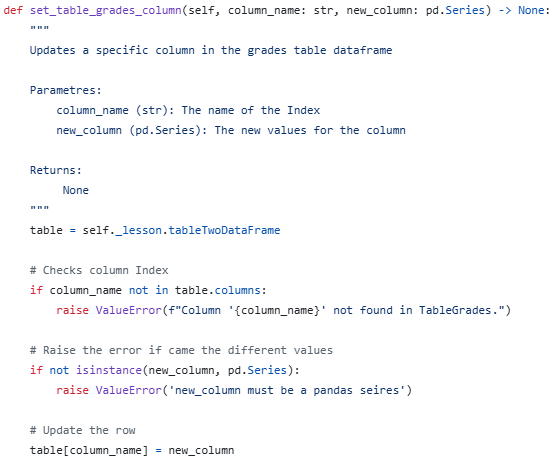
**new\_value) -> None:**

Bu fonksiyon, Lesson nesnesinin üçüncü tablosundaki (tableThreeDataFrame) belirli bir sütunun, belirli bir satırındaki değeri günceller. table = self.\_lesson.tableOneDataFrame ifadesi yanlışlıkla birinci tabloyu referans almış, doğru referans tableThreeDataFrame olmalıdır. if column\_name not in table.columns kontrolü, güncellenmek istenen sütunun mevcut olup olmadığını kontrol eder; eğer sütun yoksa, raise ValueError(f"Column '{column\_name}' not found in Table3") ile hata fırlatır. if not (0 <= row\_index < len(table)) kontrolü, satır indeksinin geçerli olup olmadığını kontrol eder; eğer geçersizse, raise IndexError(f"Row index '{row\_index}' is out of range for Table3") ile hata oluşturur. Eğer tüm kontroller geçilirse, table.at[row\_index, column\_name] = new\_value ifadesiyle belirtilen satır ve sütundaki hücre, verilen yeni değerle güncellenir. Bu fonksiyon, tablo 3’te belirli bir hücre değerini güvenli bir şekilde değiştirmek için tasarlanmıştır, ancak koddaki tablo referans hatası düzeltilmelidir.

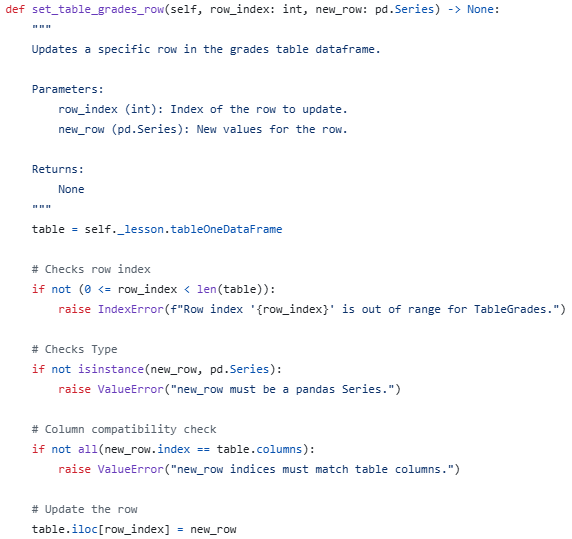
**def set\_table\_grades(self, new\_table: pd.DataFrame) -> None:**

Bu fonksiyon, Lesson nesnesinin notlar tablosunu (tableGradesDataFrame) günceller. if not isinstance(new\_table, pd.DataFrame) ifadesi, verilen new\_table parametresinin bir pandas DataFrame olup olmadığını kontrol eder; eğer değilse, raise ValueError("new\_table must be a pandas DataFrame.") ile hata oluşturur. Eğer uygun bir DataFrame verilmişse, self.\_lesson.tableTwoDataFrame = new\_table ifadesiyle tablo güncellenir, ancak bu ifade yanlışlıkla ikinci tabloyu (tableTwoDataFrame) referans almıştır. Doğru ifade self.\_lesson.tableGradesDataFrame = new\_table olmalıdır. Bu fonksiyon, notlar tablosunu tamamen güncelleyerek doğru veri ile değiştirmek için tasarlanmıştır ancak koddaki referans hatası düzeltilmelidir.

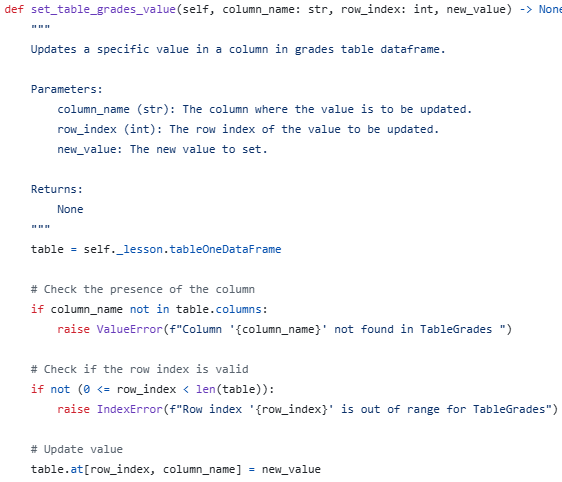
**def set\_table\_grades\_column(self, column\_name: str, new\_column: pd.Series) -> None:**

Bu fonksiyon, Lesson nesnesinin notlar tablosundaki (tableGradesDataFrame) belirli bir sütunu günceller. table = self.\_lesson.tableTwoDataFrame ifadesi yanlışlıkla ikinci tabloyu (tableTwoDataFrame) referans almış, doğru referans tableGradesDataFrame olmalıdır. if column\_name not in table.columns kontrolü, güncellenmek istenen sütunun mevcut olup olmadığını kontrol eder; eğer sütun yoksa, raise ValueError(f"Column '{column\_name}' not found in TableGrades.") ile hata oluşturur. if not isinstance(new\_column, pd.Series) kontrolü, yeni sütunun bir pandas Series olup olmadığını kontrol eder; değilse, raise ValueError('new\_column must be a pandas series') ile hata fırlatır. Eğer kontroller geçilirse, table[column\_name] = new\_column ifadesiyle belirtilen sütun, new\_column ile güncellenir. Bu fonksiyon, notlar tablosunda belirli bir sütunu güvenli ve doğru bir şekilde değiştirmek için tasarlanmıştır, ancak koddaki tablo referans hatası düzeltilmelidir.

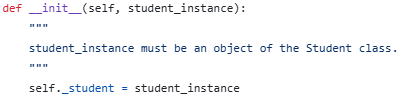
**def set\_table\_grades\_row(self, row\_index: int, new\_row: pd.Series) -> None:**

Bu fonksiyon, Lesson nesnesinin notlar tablosundaki (tableGradesDataFrame) belirli bir satırı günceller. table = self.\_lesson.tableOneDataFrame ifadesi yanlışlıkla birinci tabloyu (tableOneDataFrame) referans almış, doğru referans tableGradesDataFrame olmalıdır. if not (0 <= row\_index < len(table)) kontrolü, belirtilen satır indeksinin tablodaki geçerli bir satır olup olmadığını doğrular; eğer geçersizse, raise IndexError(f"Row index '{row\_index}' is out of range for TableGrades.") ile hata fırlatır. if not isinstance(new\_row, pd.Series) kontrolü, yeni satırın bir pandas Series olup olmadığını kontrol eder; eğer değilse, raise ValueError("new\_row must be a pandas Series.") ile hata oluşturur. if not all(new\_row.index == table.columns) kontrolü, yeni satırın sütun başlıklarıyla uyumlu olup olmadığını doğrular; uyumsuzsa, raise ValueError("new\_row indices must match table columns.") ile hata oluşturur. Eğer tüm kontroller geçilirse, table.iloc[row\_index] = new\_row ifadesiyle belirtilen satır, yeni değerlerle güncellenir. Bu fonksiyon, notlar tablosunda belirli bir satırı güvenli ve doğru bir şekilde değiştirmek için tasarlanmıştır, ancak koddaki tablo referans hatası düzeltilmelidir.

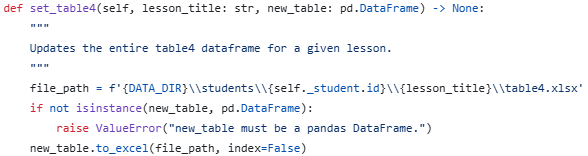
**def set\_table\_grades\_value(self, column\_name: str, row\_index: int, new\_value) -> None:**

Bu fonksiyon, Lesson nesnesinin notlar tablosundaki (tableGradesDataFrame) belirli bir sütunun, belirli bir satırındaki değeri günceller. table = self.\_lesson.tableOneDataFrame ifadesi yanlışlıkla birinci tabloyu (tableOneDataFrame) referans almış, doğru referans tableGradesDataFrame olmalıdır. if column\_name not in table.columns kontrolü, güncellenmek istenen sütunun tablodaki mevcut sütunlar arasında olup olmadığını kontrol eder; eğer sütun yoksa, raise ValueError(f"Column '{column\_name}' not found in TableGrades") ile hata oluşturur. if not (0 <= row\_index < len(table)) kontrolü, belirtilen satır indeksinin geçerli olup olmadığını kontrol eder; eğer geçersizse, raise IndexError(f"Row index '{row\_index}' is out of range for TableGrades") ile hata oluşturur. Eğer tüm kontroller geçilirse, table.at[row\_index, column\_name] = new\_value ifadesiyle belirtilen hücre, yeni değerle güncellenir. Bu fonksiyon, notlar tablosunda belirli bir hücre değerini güvenli ve doğru bir şekilde değiştirmek için tasarlanmıştır, ancak koddaki tablo referans hatası düzeltilmelidir.

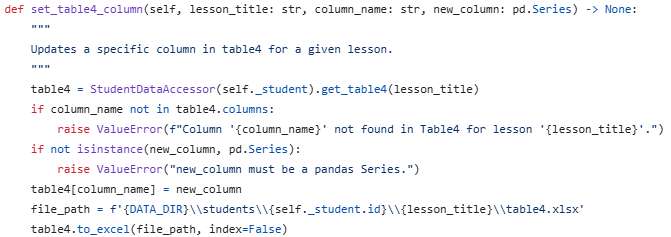
**Class StudentDataMutator:**

**def \_\_init\_\_(self, student\_instance):**Bu fonksiyon, StudentDataMutator veya StudentDataAccessor gibi bir sınıfın kurucu fonksiyonudur ve bir Student nesnesini alıp, sınıfın bir özelliği olarak saklar. self.\_student = student\_instance ifadesi, fonksiyona parametre olarak verilen student\_instance nesnesini \_student adlı bir özel özelliğe atar. Bu, sınıf içerisindeki diğer metotların ilgili Student nesnesine kolayca erişmesini sağlar. Amaç, belirli bir öğrenci üzerinde veri manipülasyonu veya erişim işlemleri yapabilmek için bu nesneyi merkezi bir şekilde kullanılabilir hale getirmektir.

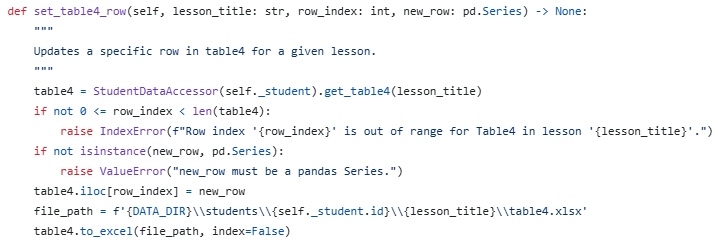
**def set\_table4(self, lesson\_title: str, new\_table: pd.DataFrame) -> None:**

Bu fonksiyon, belirli bir dersin table4.xlsx dosyasını günceller. file\_path = f'{DATA\_DIR}\\students\\{self.\_student.id}\\{lesson\_title}\\table4.xlsx' ifadesi, güncellenmek istenen table4.xlsx dosyasının tam dosya yolunu oluşturur. if not isinstance(new\_table, pd.DataFrame) kontrolü, verilen new\_table parametresinin bir pandas DataFrame olup olmadığını kontrol eder; eğer değilse, raise ValueError("new\_table must be a pandas DataFrame.") ile hata oluşturur. Eğer uygun bir DataFrame verilmişse, new\_table.to\_excel(file\_path, index=False) ifadesiyle yeni tablo, belirtilen dosya yolunda bir Excel dosyası olarak kaydedilir ve varsa eski dosyanın üzerine yazılır. Bu fonksiyon, bir öğrencinin belirli bir dersi için table4 tablosunu tamamen güncellemek amacıyla tasarlanmıştır.

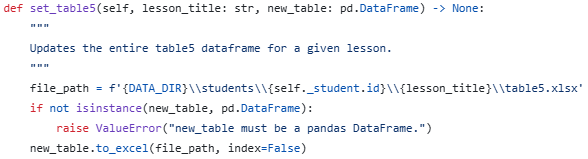
**def set\_table4\_column(self, lesson\_title: str, column\_name: str, new\_column: pd.Series) -> None:**

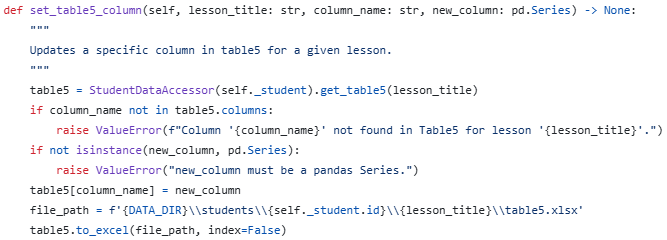
Bu fonksiyon, belirli bir dersin table4.xlsx dosyasındaki belirli bir sütunu günceller. table4 = StudentDataAccessor(self.\_student).get\_table4(lesson\_title) ifadesi, StudentDataAccessor sınıfını kullanarak öğrencinin belirtilen dersine ait table4.xlsx dosyasını bir pandas DataFrame olarak alır. if column\_name not in table4.columns kontrolü, güncellenmek istenen sütunun table4 tablosunda mevcut olup olmadığını kontrol eder; eğer sütun yoksa, raise ValueError(f"Column '{column\_name}' not found in Table4 for lesson '{lesson\_title}'.") ile hata oluşturur. if not isinstance(new\_column, pd.Series) kontrolü, yeni sütunun bir pandas Series olup olmadığını kontrol eder; eğer değilse, raise ValueError("new\_column must be a pandas Series.") ile hata fırlatır. Kontroller geçilirse, table4[column\_name] = new\_column ifadesiyle belirtilen sütun, yeni değerlerle güncellenir. Son olarak, file\_path = f'{DATA\_DIR}\\students\\{self.\_student.id}\\{lesson\_title}\\table4.xlsx' ifadesiyle dosya yolu belirlenir ve table4.to\_excel(file\_path, index=False) ile güncellenmiş tablo Excel dosyasına kaydedilir. Bu fonksiyon, bir öğrencinin belirli bir dersi için table4 tablosundaki sütun verilerini güvenli bir şekilde değiştirmek için tasarlanmıştır.

**def set\_table4\_row(self, lesson\_title: str, row\_index: int, new\_row: pd.Series) -> None:**

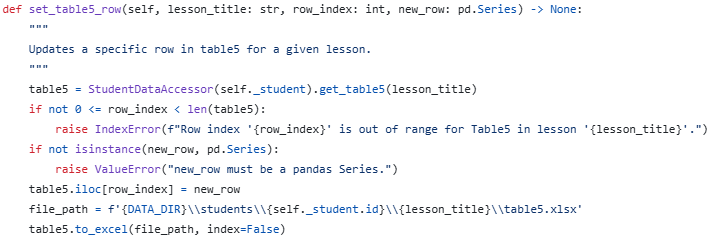
Bu fonksiyon, belirli bir dersin table4.xlsx dosyasındaki belirli bir satırı günceller. table4 = StudentDataAccessor(self.\_student).get\_table4(lesson\_title) ifadesi, StudentDataAccessor sınıfını kullanarak öğrencinin belirtilen dersine ait table4 tablosunu bir pandas DataFrame olarak alır. if not 0 <= row\_index < len(table4) kontrolü, güncellenmek istenen satırın tablo içinde geçerli bir indeks olup olmadığını kontrol eder; eğer geçersizse, raise IndexError(f"Row index '{row\_index}' is out of range for Table4 in lesson '{lesson\_title}'.") ile hata fırlatır. if not isinstance(new\_row, pd.Series) kontrolü, yeni satırın bir pandas Series olup olmadığını doğrular; değilse, raise ValueError("new\_row must be a pandas Series.") ile hata oluşturur. Kontroller geçilirse, table4.iloc[row\_index] = new\_row ifadesiyle belirtilen satır, new\_row ile güncellenir. file\_path = f'{DATA\_DIR}\\students\\{self.\_student.id}\\{lesson\_title}\\table4.xlsx' ifadesiyle dosya yolu belirlenir ve table4.to\_excel(file\_path, index=False) ile güncellenmiş tablo Excel dosyasına kaydedilir. Bu fonksiyon, bir öğrencinin belirli bir dersi için table4 tablosundaki satır verilerini güvenli bir şekilde değiştirmek için tasarlanmıştır.

**def set\_table5(self, lesson\_title: str, new\_table: pd.DataFrame) -> None:**

Bu fonksiyon, belirli bir dersin table5.xlsx dosyasını günceller. file\_path = f'{DATA\_DIR}\\students\\{self.\_student.id}\\{lesson\_title}\\table5.xlsx' ifadesi, öğrencinin belirtilen dersine ait table5.xlsx dosyasının tam dosya yolunu oluşturur. if not isinstance(new\_table, pd.DataFrame) kontrolü, verilen new\_table parametresinin bir pandas DataFrame olup olmadığını kontrol eder; eğer değilse, raise ValueError("new\_table must be a pandas DataFrame.") ile hata oluşturur. Eğer uygun bir DataFrame verilmişse, new\_table.to\_excel(file\_path, index=False) ifadesiyle yeni tablo, belirtilen dosya yolunda bir Excel dosyası olarak kaydedilir ve varsa eski dosyanın üzerine yazılır. Bu fonksiyon, bir öğrencinin belirli bir dersi için table5 tablosunu tamamen güncellemek amacıyla tasarlanmıştır.

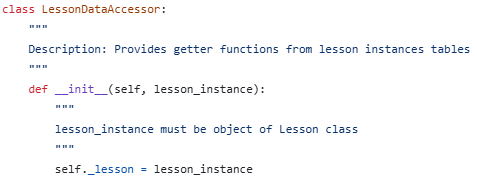
**def set\_table5\_column(self, lesson\_title: str, column\_name: strnew\_column: pd.Series) -> None:**  
Bu fonksiyon, belirli bir dersin table5.xlsx dosyasındaki belirli bir sütunu günceller. table5 = StudentDataAccessor(self.\_student).get\_table5(lesson\_title) ifadesi, StudentDataAccessor sınıfını kullanarak öğrencinin belirtilen dersine ait table5 tablosunu bir pandas DataFrame olarak alır. if column\_name not in table5.columns kontrolü, güncellenmek istenen sütunun tablo içinde mevcut olup olmadığını kontrol eder; eğer sütun yoksa, raise ValueError(f"Column '{column\_name}' not found in Table5 for lesson '{lesson\_title}'.") ile hata oluşturur. if not isinstance(new\_column, pd.Series) kontrolü, yeni sütunun bir pandas Series olup olmadığını doğrular; değilse, raise ValueError("new\_column must be a pandas Series.") ile hata oluşturur. Kontroller geçilirse, table5[column\_name] = new\_column ifadesiyle belirtilen sütun, new\_column ile güncellenir. file\_path = f'{DATA\_DIR}\\students\\{self.\_student.id}\\{lesson\_title}\\table5.xlsx' ifadesiyle dosya yolu belirlenir ve table5.to\_excel(file\_path, index=False) ifadesiyle güncellenmiş tablo Excel dosyasına kaydedilir. Bu fonksiyon, bir öğrencinin belirli bir dersi için table5 tablosundaki sütun verilerini güvenli bir şekilde değiştirmek için tasarlanmıştır.

**def set\_table5\_row(self, lesson\_title: str, row\_index: int, new\_row: pd.Series) -> None:**

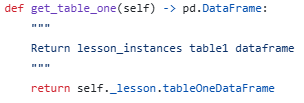
Bu fonksiyon, belirli bir dersin table5.xlsx dosyasındaki belirli bir satırı günceller. table5 = StudentDataAccessor(self.\_student).get\_table5(lesson\_title) ifadesi, StudentDataAccessor sınıfını kullanarak öğrencinin belirtilen dersine ait table5 tablosunu bir pandas DataFrame olarak alır. if not 0 <= row\_index < len(table5) kontrolü, güncellenmek istenen satırın tablodaki geçerli bir indeks olup olmadığını kontrol eder; eğer geçersizse, raise IndexError(f"Row index '{row\_index}' is out of range for Table5 in lesson '{lesson\_title}'.") ile hata fırlatır. if not isinstance(new\_row, pd.Series) kontrolü, yeni satırın bir pandas Series olup olmadığını kontrol eder; eğer değilse, raise ValueError("new\_row must be a pandas Series.") ile hata oluşturur. Kontroller geçilirse, table5.iloc[row\_index] = new\_row ifadesiyle belirtilen satır, new\_row ile güncellenir. file\_path = f'{DATA\_DIR}\\students\\{self.\_student.id}\\{lesson\_title}\\table5.xlsx' ifadesiyle dosya yolu belirlenir ve table5.to\_excel(file\_path, index=False) ifadesiyle güncellenmiş tablo Excel dosyasına kaydedilir. Bu fonksiyon, bir öğrencinin belirli bir dersi için table5 tablosundaki satır verilerini güvenli bir şekilde değiştirmek için tasarlanmıştır.

**Class LessonDataAccessor:**

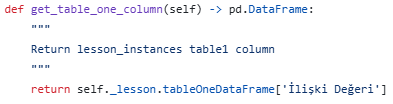
**def \_\_init\_\_(self, lesson\_instance):**

Bu fonksiyon, LessonDataAccessor veya LessonDataMutator gibi bir sınıfın kurucu fonksiyonudur ve bir Lesson nesnesini alıp, bu nesneyi sınıfın bir özelliği olarak saklar. self.\_lesson = lesson\_instance ifadesi, parametre olarak verilen lesson\_instance nesnesini \_lesson adlı özel bir özelliğe atar. Bu sayede, sınıf içerisindeki diğer metotlar ilgili Lesson nesnesine kolayca erişebilir ve üzerinde işlem yapabilir. Bu yapı, belirli bir dersle ilişkili verilere erişim veya bu verilerin manipülasyonu için bir temel oluşturur.

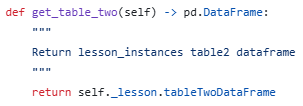
**def get\_table\_one(self) -> pd.DataFrame:**

Bu fonksiyon, Lesson nesnesine ait birinci tabloyu (tableOneDataFrame) döndürür. return self.\_lesson.tableOneDataFrame ifadesi, kurucu fonksiyonda \_lesson özelliği olarak saklanan Lesson nesnesinin tableOneDataFrame özelliğini çağırır ve bir pandas DataFrame olarak döner. Bu fonksiyon, Lesson nesnesindeki birinci tabloya hızlı ve doğrudan erişim sağlamak için tasarlanmıştır.

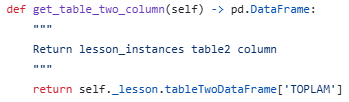
**def get\_table\_one\_column(self) -> pd.DataFrame:**

Bu fonksiyon, Lesson nesnesine ait birinci tablodaki (tableOneDataFrame) "İlişki Değeri" adlı sütunu döndürür. return self.\_lesson.tableOneDataFrame['İlişki Değeri'] ifadesi, Lesson nesnesindeki tableOneDataFrame tablosundan "İlişki Değeri" sütununu seçer ve bir pandas Series olarak döner. Bu fonksiyon, birinci tablodaki "İlişki Değeri" sütununa hızlı ve doğrudan erişim sağlamak için tasarlanmıştır. Eğer sütun bulunamazsa, kod çalıştırıldığında hata fırlatacaktır.

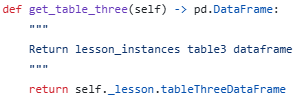
**def get\_table\_two(self) -> pd.DataFrame:**

Bu fonksiyon, Lesson nesnesine ait ikinci tabloyu (tableTwoDataFrame) döndürür. return self.\_lesson.tableTwoDataFrame ifadesi, kurucu fonksiyon tarafından \_lesson özelliği olarak saklanan Lesson nesnesinin tableTwoDataFrame özelliğini çağırır ve bir pandas DataFrame olarak döner. Bu fonksiyon, Lesson nesnesindeki ikinci tabloya hızlı ve doğrudan erişim sağlamak için tasarlanmıştır.

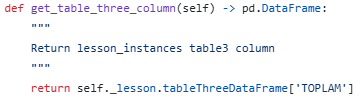
**def get\_table\_two\_column(self) -> pd.DataFrame:**

Bu fonksiyon, Lesson nesnesine ait ikinci tablodaki (tableTwoDataFrame) "TOPLAM" adlı sütunu döndürür. return self.\_lesson.tableTwoDataFrame['TOPLAM'] ifadesi, Lesson nesnesindeki tableTwoDataFrame tablosundan "TOPLAM" sütununu seçer ve bir pandas Series olarak döner. Bu fonksiyon, ikinci tablodaki "TOPLAM" sütununa hızlı ve doğrudan erişim sağlamak için tasarlanmıştır. Eğer "TOPLAM" sütunu mevcut değilse, kod çalıştırıldığında hata fırlatacaktır.

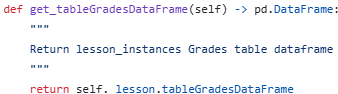
**def get\_table\_three(self) -> pd.DataFrame:**

Bu fonksiyon, Lesson nesnesine ait üçüncü tabloyu (tableThreeDataFrame) döndürür. return self.\_lesson.tableThreeDataFrame ifadesi, Lesson nesnesindeki tableThreeDataFrame özelliğini çağırır ve bir pandas DataFrame olarak döner. Bu fonksiyon, Lesson nesnesindeki üçüncü tabloya hızlı ve doğrudan erişim sağlamak için tasarlanmıştır.

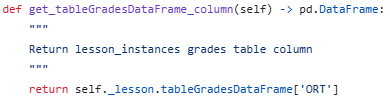
**def get\_table\_three\_column(self) -> pd.DataFrame:**

Bu fonksiyon, Lesson nesnesine ait üçüncü tablodaki (tableThreeDataFrame) **"TOPLAM"** adlı sütunu döndürür. **return self.\_lesson.tableThreeDataFrame['TOPLAM']** ifadesi, Lesson nesnesindeki tableThreeDataFrame tablosundan **"TOPLAM"** sütununu seçer ve bir pandas Series olarak döner. Bu fonksiyon, üçüncü tablodaki **"TOPLAM"** sütununa hızlı ve doğrudan erişim sağlamak için tasarlanmıştır. Eğer **"TOPLAM"** sütunu mevcut değilse, kod çalıştırıldığında hata oluşturacaktır.

**def get\_tableGradesDataFrame(self) -> pd.DataFrame:**

Bu fonksiyon, Lesson nesnesine ait notlar tablosunu (tableGradesDataFrame) döndürür. return self.\_lesson.tableGradesDataFrame ifadesi, Lesson nesnesindeki tableGradesDataFrame özelliğini çağırır ve bir pandas DataFrame olarak döner. Bu fonksiyon, Lesson nesnesindeki notlar tablosuna hızlı ve doğrudan erişim sağlamak için tasarlanmıştır.

**def get\_tableGradesDataFrame\_column(self) -> pd.DataFrame:**

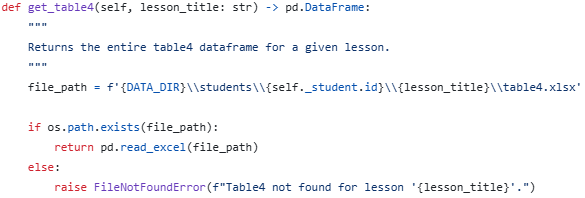
Bu fonksiyon, Lesson nesnesine ait notlar tablosundaki (tableGradesDataFrame) "ORT" adlı sütunu döndürür. return self.\_lesson.tableGradesDataFrame['ORT'] ifadesi, Lesson nesnesindeki tableGradesDataFrame tablosundan "ORT" sütununu seçer ve bir pandas Series olarak döner. Bu fonksiyon, notlar tablosundaki "ORT" sütununa hızlı ve doğrudan erişim sağlamak için tasarlanmıştır. Eğer "ORT" sütunu mevcut değilse, kod çalıştırıldığında hata oluşturur.

**Class StudentDataAccessor:**

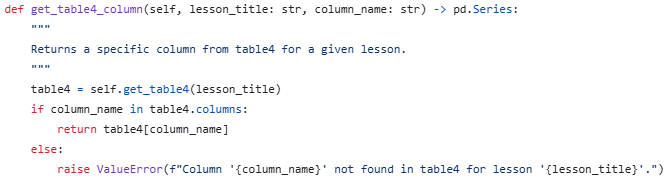
**def \_\_init\_\_(self, student\_instance):**

Bu fonksiyon, StudentDataAccessor veya StudentDataMutator gibi bir sınıfın kurucu fonksiyonudur ve bir Student nesnesini alıp bu nesneyi sınıfın bir özelliği olarak saklar. self.\_student = student\_instance ifadesi, parametre olarak verilen student\_instance nesnesini \_student adlı özel bir özelliğe atar. Bu sayede, sınıf içerisindeki diğer metotlar ilgili Student nesnesine kolayca erişebilir ve üzerinde işlem yapabilir. Amaç, belirli bir öğrenciyle ilişkili verilere erişim veya bu verilerin manipülasyonu için bir temel oluşturmaktır.

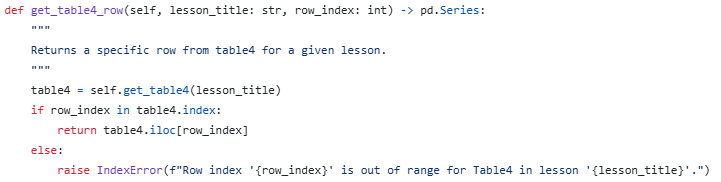
**def get\_table4(self, lesson\_title: str) -> pd.DataFrame:**

Bu fonksiyon, belirli bir dersin table4.xlsx dosyasını okuyarak bir pandas DataFrame olarak döndürür. **file\_path = f'{DATA\_DIR}\\students\\{self.\_student.id}\\{lesson\_title}\\table4.xlsx'** ifadesi, öğrencinin belirtilen dersine ait table4.xlsx dosyasının tam yolunu oluşturur. **if os.path.exists(file\_path)** kontrolü, belirtilen dosyanın mevcut olup olmadığını kontrol eder. Eğer dosya mevcutsa, **return pd.read\_excel(file\_path)** ifadesiyle dosya okunur ve bir pandas DataFrame olarak döndürülür. Eğer dosya bulunamazsa, **raise FileNotFoundError(f"Table4 not found for lesson '{lesson\_title}'.")** ifadesiyle bir hata fırlatılır. Bu fonksiyon, bir öğrencinin belirli bir dersi için table4 tablosuna güvenli ve hızlı bir şekilde erişim sağlamak amacıyla tasarlanmıştır.

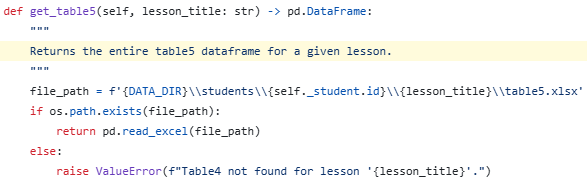
**def get\_table4\_column(self, lesson\_title: str, column\_name: str) -> pd.Series:**

Bu fonksiyon, belirli bir dersin table4.xlsx dosyasından belirtilen sütunu döndürür. table4 = self.get\_table4(lesson\_title) ifadesi, get\_table4 fonksiyonunu kullanarak dersin table4 tablosunu bir pandas DataFrame olarak alır. if column\_name in table4.columns kontrolü, istenen sütunun table4 tablosunda mevcut olup olmadığını kontrol eder. Eğer sütun mevcutsa, return table4[column\_name] ifadesiyle bu sütun bir pandas Series olarak döndürülür. Eğer sütun bulunamazsa, raise ValueError(f"Column '{column\_name}' not found in table4 for lesson '{lesson\_title}'.") ifadesiyle bir hata fırlatılır. Bu fonksiyon, bir öğrencinin belirli bir dersi için table4 tablosundaki belirli bir sütuna güvenli ve hızlı bir şekilde erişim sağlamak amacıyla tasarlanmıştır.

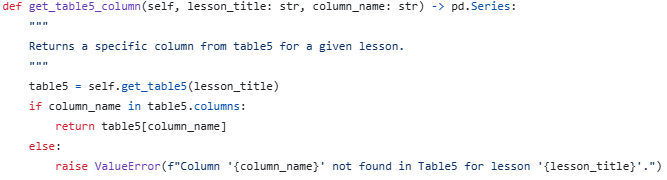
**def get\_table4\_row(self, lesson\_title: str, row\_index: int) -> pd.Series:**

Bu fonksiyon, belirli bir dersin table4.xlsx dosyasından belirtilen satırı döndürür. table4 = self.get\_table4(lesson\_title) ifadesi, get\_table4 fonksiyonunu kullanarak dersin table4 tablosunu bir pandas DataFrame olarak alır. if row\_index in table4.index kontrolü, istenen satırın belirtilen tablo içinde geçerli bir indeks olup olmadığını kontrol eder. Eğer geçerli bir indeksse, return table4.iloc[row\_index] ifadesiyle bu satır bir pandas Series olarak döndürülür. Eğer satır indeksi geçerli değilse, raise IndexError(f"Row index '{row\_index}' is out of range for Table4 in lesson '{lesson\_title}'.") ifadesiyle bir hata fırlatılır. Bu fonksiyon, bir öğrencinin belirli bir dersi için table4 tablosundaki belirli bir satıra güvenli ve hızlı bir şekilde erişim sağlamak amacıyla tasarlanmıştır.

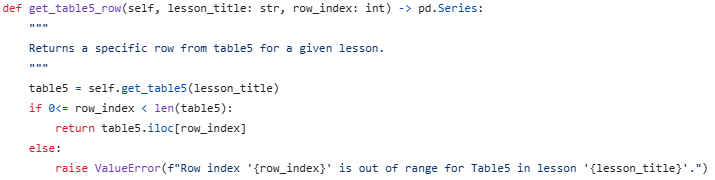
**def get\_table5(self, lesson\_title: str) -> pd.DataFrame:**

Bu fonksiyon, belirli bir dersin table5.xlsx dosyasını okuyarak bir pandas DataFrame olarak döndürür. file\_path = f'{DATA\_DIR}\\students\\{self.\_student.id}\\{lesson\_title}\\table5.xlsx' ifadesi, öğrencinin belirtilen dersine ait table5.xlsx dosyasının tam yolunu oluşturur. if os.path.exists(file\_path) kontrolü, belirtilen dosyanın mevcut olup olmadığını kontrol eder. Eğer dosya mevcutsa, return pd.read\_excel(file\_path) ifadesiyle dosya okunur ve bir pandas DataFrame olarak döndürülür. Eğer dosya bulunamazsa, raise ValueError(f"Table4 not found for lesson '{lesson\_title}'.") ifadesiyle hata fırlatılır. (Hata mesajında bir yazım hatası var, "Table4" yerine "Table5" yazılması gerekirdi.) Bu fonksiyon, bir öğrencinin belirli bir dersi için table5 tablosuna güvenli ve hızlı bir şekilde erişim sağlamak amacıyla tasarlanmıştır.

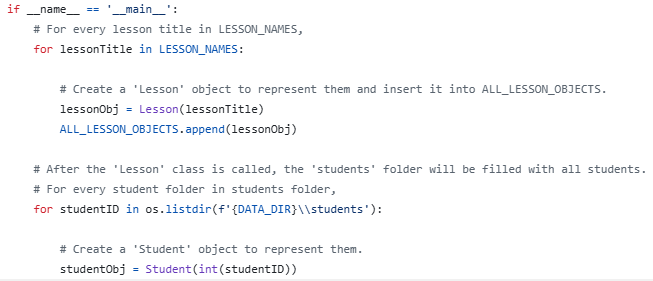
**def get\_table5\_column(self, lesson\_title: str, column\_name: str) -> pd.Series:**

Bu fonksiyon, belirli bir dersin table5.xlsx dosyasından belirtilen sütunu döndürür. table5 = self.get\_table5(lesson\_title) ifadesi, get\_table5 fonksiyonunu kullanarak dersin table5 tablosunu bir pandas DataFrame olarak alır. if column\_name in table5.columns kontrolü, istenen sütunun tablo içinde mevcut olup olmadığını kontrol eder. Eğer sütun mevcutsa, return table5[column\_name] ifadesiyle bu sütun bir pandas Series olarak döndürülür. Eğer sütun bulunamazsa, raise ValueError(f"Column '{column\_name}' not found in Table5 for lesson '{lesson\_title}'.") ifadesiyle bir hata fırlatılır. Bu fonksiyon, bir öğrencinin belirli bir dersi için table5 tablosundaki belirli bir sütuna güvenli ve hızlı bir şekilde erişim sağlamak amacıyla tasarlanmıştır.

**def get\_table5\_row(self, lesson\_title: str, row\_index: int) -> pd.Series:**

Bu fonksiyon, belirli bir dersin table5.xlsx dosyasından belirtilen satırı döndürür. table5 = self.get\_table5(lesson\_title) ifadesi, get\_table5 fonksiyonunu kullanarak dersin table5 tablosunu bir pandas DataFrame olarak alır. if 0 <= row\_index < len(table5) kontrolü, istenen satırın tablo içinde geçerli bir indeks olup olmadığını kontrol eder. Eğer geçerli bir indeksse, return table5.iloc[row\_index] ifadesiyle bu satır bir pandas Series olarak döndürülür. Eğer satır indeksi geçerli değilse, raise ValueError(f"Row index '{row\_index}' is out of range for Table5 in lesson '{lesson\_title}'.") ifadesiyle bir hata fırlatılır. Bu fonksiyon, bir öğrencinin belirli bir dersi için table5 tablosundaki belirli bir satıra güvenli ve hızlı bir şekilde erişim sağlamak amacıyla tasarlanmıştır.

**if \_\_name\_\_ == '\_\_main\_\_':**

Bu kod, programın başlangıcında çalıştırılarak Lesson ve Student nesnelerinin oluşturulmasını sağlar.

for lessonTitle in LESSON\_NAMES: döngüsü, LESSON\_NAMES listesindeki her bir ders adı için bir Lesson nesnesi oluşturur. lessonObj = Lesson(lessonTitle) ifadesiyle bir Lesson nesnesi oluşturulur ve ALL\_LESSON\_OBJECTS.append(lessonObj) ile bu nesne, tüm derslerin saklandığı ALL\_LESSON\_OBJECTS listesine eklenir. Bu işlem, programın tüm derslere kolayca erişebilmesini sağlar.

Ardından, for studentID in os.listdir(f'{DATA\_DIR}\\students'): döngüsü, students klasöründeki her bir öğrenci klasörü için bir Student nesnesi oluşturur. studentObj = Student(int(studentID)) ifadesiyle her öğrenciye karşılık bir Student nesnesi oluşturulur. Bu işlem, tüm öğrencilerin veri yapılarıyla temsil edilmesini ve derslerle ilişkilendirilmesini sağlar.

Kod, ders ve öğrenci nesnelerinin oluşturulup düzenli bir şekilde saklanmasını ve kullanılmasını sağlayan temel bir yapı oluşturur.

**4.2 Görev Dağılımı**

**Geliştirici Emrah Şahin:**

* Lesson ve Student sınıflarının geliştirilmesi.
* Tablo1 ve Tablo2 alımından sonra Tablo3 oluşturulması ve güncellenmesi.

**Geliştirici Yavuz Selim Gürsoy:**

* Tablo4 ve Tablo5 oluşturulması ve gerekli yerlerde tablo değişiklikleri yapması.
* Bir sonraki proje aşaması için kullanıcı arayüzü kolaylıklarını sağlama.

**Geliştirici Talha Tuna:**

* Excel okuma ve yazdırma

**Geliştirici Müge Özler:**

* Test senaryolarının yazılması ve test süreçlerinin yürütülmesi.
* Yazılımın dokümantasyonunun hazırlanması.

**5 TEST VE DOĞRULAMA**

**5.1 Yazılımın Test Süreci**

Yazılım, unittest modülü kullanılarak test edilmiştir. Her bir sınıf ve fonksiyon için aşağıdaki testler gerçekleştirilmiştir:

**Lesson Sınıfı Testleri:**

* Başlatma, tablo oluşturma, öğrenci klasörlerinin oluşturulması.
* Tabloların doğru bir şekilde güncellenip güncellenmediği.

**Student Sınıfı Testleri:**

* Başlatma, ders bulma, öğrenci tablolarının oluşturulması.
* Öğrenci bilgileri ile ders bilgileri arasındaki ilişki.

**Fonksiyonel Testler:**

* Kullanıcı etkileşimleri ve beklenen sonuçların doğrulanması.
* Not girişi ve başarı hesaplama işlemlerinin doğruluğu.
* Test süreci, yazılımın güvenilirliğini ve işlevselliğini sağlamak için düzenli olarak yapılacaktır. Her yeni özellik eklendiğinde veya değişiklik yapıldığında testlerin güncellenmesi ve çalıştırılması gerekecektir.

**5.2 Hata Yönetimi**

Yazılım, kullanıcı hatalarını ve sistem hatalarını yönetmek için uygun hata mesajları ve istisna yönetimi mekanizmaları içermektedir. Kullanıcıların karşılaşabileceği yaygın hatalar için bilgilendirici mesajlar sağlanacaktır.

**6 SONUÇ**

Bu rapor, yazılım projesinin kapsamını, gereksinimlerini, tasarımını, uygulamasını ve test sürecini detaylı bir şekilde açıklamaktadır. Proje, eğitim kurumlarının yönetim süreçlerini daha verimli hale getirmeyi hedeflemektedir. Geliştirilen yazılım, kullanıcıların ders ve öğrenci bilgilerini etkili bir şekilde yönetmelerine olanak tanırken, aynı zamanda başarı oranlarını hesaplayarak eğitim süreçlerine katkıda bulunmaktadır. Yazılımın gelecekteki sürümlerinde, kullanıcı geri bildirimleri doğrultusunda yeni özellikler eklenmesi ve mevcut işlevlerin iyileştirilmesi planlanmaktadır

**7 GitHub Bağlantıları**

* [Müge Özler](https://github.com/Yavuz-Selim-Gursoy/Ders-Programi)
* [Emrah Şahin](https://github.com/Yavuz-Selim-Gursoy/Ders-Programi)
* [Yavuz Selim Gürsoy](https://github.com/Yavuz-Selim-Gursoy/Ders-Programi)
* [Talha Tuna](https://github.com/Yavuz-Selim-Gursoy/Ders-Programi)