'==== Aufgabe 5 - Applikation nach Vorgabe erstellen ====

Ziel

Sie können Objekte nach Vorgabe eines Klassen- und Sequenzdiagramms erstellen und kommunizieren lassen.

Hinweise

- Lösen Sie die Aufgabe jeweils step by step und führen Sie dann den Code aus, um zu prüfen, ob Ihre Implementation erfolgreich ist.
- Die Klassen `Book` und `Library` werden nicht bearbeitet. Lesen Sie aber dennoch die Beschreibungen der Methoden, damit Sie wissen, wie diese anzuwenden sind.
- Eine OO-Anwendung startet damit, dass in der `main`-Methode alle wichtigen Klassen instanziiert und deren Beziehungen sichergestellt werden.
- Das Klassendiagramm gibt den statischen Aufbau der Anwendung wieder.

Auftrag

Teil 1 : ** Die Objekte werden erzeugt. - Akzeptieren Sie das Assignment in GitHub Classroom und klonen Sie das Repo in Ihre Entwicklungsumgebung. - Implementieren Sie den Konstruktor der Klassen `Customer`. **Achtung:** *Beim Erzeugen eines Customer-Objektes meldet sich dieses selber (proaktiv) bei der Library an. Sie können das im Ablauf des Konstruktors erkennen, da dort der Aufruf `add_customer` ausgeführt wird.*

1. Führen Sie nun in `test_library.py` die folgende Tests aus, welche fehlerfrei sein müssen:
   - `test_add_and_print_customers`
   - `test_search_customer`
   - `test_search_customer_failed`
2. Pushen Sie die aktuelle Lösung auf GitHub.
3. Implementieren Sie den Konstruktor der Klasse `Librarian`.
   - Hinweis: *Es gibt in diesem Zustand der Klasse keine Tests, die prüfen, ob der Konstruktor erfolgreich angelegt wurde. Wenn Sie unsicher sind, ob Sie die Aufgabe richtig gelöst haben, frage Sie bei Ihrer Lehrperson nach.*
4. Implementieren Sie nun in der `main`-Methode (`main.py`) die Instanziierung der Objekte sowie die Methodenaufrufe gemäss dem gezeigten Sequenzdiagramm.
   - Hinweis: *Halten Sie sich an die Anweisungen im Quellcode. //*
     - Führen Sie dann das Programm aus. Es muss fehlerfrei laufen.
     Die Ausgabe soll wie folgt aussehen:
     - Testen Sie das Programm mit der Methode `test_main_part1` in `test_main.py`. Wie immer muss auch dieser Test fehlerfrei ablaufen. - Pushen Sie den Teilauftrag 1 auf GitHub
     ***Teil 2 : ** Der Bibliothek werden 5 Bücher zugeführt. - Erzeugen Sie für die 2. Teilaufgabe einen neuen Branch. So ist sichergestellt, dass Sie - im Notfall - jederzeit auf eine lauffähige Version zurückgreifen können. - Ergänzen Sie den Code der Methode `buy_new_book` in der Klasse `Librarian`.*

//Hinweise:
- *Halten Sie sich an die Anweisungen im Code der Datei `librarian.py` // * //Beachten Sie bitte die Beschreibung der benötigten Methoden in den Klassen `Library` und `Book`. // * //Der Wert des Attributs `location` ist ein Zufallswert der in der Methode `add_book()` generiert wird. // * //Halten Sie sich an das gezeigte Sequenzdiagramm. // - Führen Sie den Test `test_buy_new_book` in der Datei `test_librarian.py` aus. Er muss fehlerfrei ablaufen. - Pushen Sie die aktuelle Lösung auf GitHub. - Ergänzen Sie den Code in der `main`-Methode. Halten Sie sich an die Anweisungen im Code der Datei `main.py`. //Hinweise:*
- *Übernehmen Sie die Buchtitel und ISBN-Nummern genau so, wie im Printout angegeben.*
- *Halten Sie sich an das gezeigte Sequenzdiagramm.*

5. Führen Sie das Programm aus. Es muss fehlerfrei laufen.
   Die Ausgabe soll wie folgt aussehen:
   Hinweis:
   · Der Ablageort wird durch die Library generiert und kann darum variieren!
6. Pushen Sie die lauffähige Teilaufgabe 2 auf GitHub.

Teil 3 : ** Ursula und Moritz leihen sich je ein Buch aus.
Ursula wählt das Buch „Das Omen“ und Moritz das Buch „Ich bin dann mal weg“.
- Erstellen Sie - basierend auf dem Branch aus Teilaufgabe 2 - einen neuen Branch für die Teilaufgabe 3. - Ergänzen Sie in der Klasse `Librarian` die Methode `borrow_a_book_by_title`.
Hinweise:
· Halten Sie sich an die Anweisungen im Code der Datei `librarian.py`
· Beachten Sie bitte die Beschreibung der benötigten Methoden in den Klassen `Library` und `Book`.
· Halten Sie sich an das gezeigte Sequenzdiagramm. Der Ablauf ist etwas komplex gehalten, dies aber bewusst, um möglichst viel „Kommunikation“ zu erzielen.
- Führen Sie den Test `test_borrow_a_book_by_title` in der Datei `test_librarian.py` aus. Er muss fehlerfrei ablaufen. - Pushen Sie diese Lösung auf GitHub. - Führen Sie nun den Test `test_borrow_a_book_by_unknown_title` in der Datei `test_librarian.py` aus. Er muss fehlerfrei ablaufen. - Pushen Sie diese Lösung auf GitHub. - Ergänzen Sie nun in der Klasse `Customer` die Methode `borrow_a_book_by_title`.
****Achtung!** Sie benötigen hier die Referenz `self._book` auf ein Book-Objekt, um den Rückgabewert aus dem Methodenaufruf zu speichern. Das lässt sich aus dem Sequenzdiagramm so nicht entnehmen. Dieses Wissen müssen Sie als Fachperson hier einbringen. Fehlt die Deklaration dieses Attributs im Konstruktor, müssen Sie das jetzt nachholen.
Hinweis:
· initialisieren Sie `self._book` mit dem Wert None //
- Führen Sie nun den Test `test_borrow_a_book_by_title` in der Datei `test_customer.py` aus. Er muss fehlerfrei ablaufen. - Pushen Sie die lauffähige Lösung auf GitHub. - Ergänzen Sie den Code in der `main`-Methode. Halten Sie sich an die Anweisungen im Code der Datei `main.py`.
//Hinweis :
· Halten Sie sich an das gezeigte Sequenzdiagramm. //
- Führen Sie das Programm aus. Es muss fehlerfrei laufen.
Die Ausgabe soll in etwa wie folgt aussehen:
- Pushen Sie die lauffähige Teilaufgabe 3 auf GitHub. //Hinweis:

· die main-Methode wird nicht mehr mit einem eigenen Test geprüft, da alle Methoden die benötigt werden, schon getestete sind. Sie müssen hier in Eigenverantwortung schauen, ob das angezeigte Ergebnis korrekt ist.

**Teil 4: ** Ursula bringt ihr Buch zurück.

1. Ergänzen Sie den Code der Methode `get_a_book_from_customer` in der Datei `librarian.py`.
2. Testen Sie den Code mit der Methode `test_get_a_book_from_customer` in der Datei `test_librarian.py`.
3. Pushen Sie die Lösung auf GitHub.
4. Ergänzen Sie den Code der Methode `bring_back_a_book` in der Datei `customer.py`.
5. Testen Sie den Code mit der Methode `test_bring_back_a_book` in der Datei `test_customer.py`.
6. Pushen Sie die Lösung auf GitHub.
7. Ergänzen Sie den Code in der `main`-Methode. //
   Hinweis:
   · Halten Sie sich an die Anweisungen im Code der Dateien `Librarian.py`, `Customer.py` und `main.py`.
   · Halten Sie sich an das gezeigte Sequenzdiagramm.//
8. Führen Sie das Programm nun aus. Es muss fehlerfrei laufen.
   Die Ausgabe soll in etwa wie folgt aussehen:
9. Pushen Sie den Teilauftrag 4 auf GitHub

**Teil 5 : ** Moritz wird gemahnt

1. Ergänzen Sie den Code der Methode `remind_customer` in der Datei `librarian.py`.
2. Testen Sie den Code mit der Methode `test_get_a_book_from_customer` in der Datei `test_librarian.py`.
3. Pushen Sie die Lösung auf GitHub.
4. Ergänzen Sie den Code in der `main`-Methode.
   //Hinweis: \\· Halten Sie sich an das Sequenzdiagramm. //
5. Führen Sie das Programm aus. Es muss fehlerfrei laufen.
   Die Ausgabe soll in etwa wie folgt aussehen:
6. Pushen Sie den Teilauftrag 5 auf GitHub.

**Teil 6 : ** Der Bibliothekar entfernt ein Buch aus der Bibliothek.
Entfernt wird das Buch „Harry Potter, die neue Welt“

1. Ergänzen Sie den Code in der Methode `remove_book` in der Datei `librarian.py`.
2. Testen Sie den Code mit der Methode `test_remove_book` in der Datei `test_librarian.py`.
3. Pushen Sie die Lösung auf GitHub.
4. Ergänzen Sie den Code in der `main`-Methode.
   //Hinweise:
   · Halten Sie sich an das Sequenzdiagramm. //

5. Führen Sie das Programm aus. Es muss fehlerfrei laufen.
   Die Ausgabe soll in etwa wie folgt aussehen:
   
6. Pushen Sie den Teilauftrag 6 auf GitHub

**Teil 7 : ** Ursula will ein Buch mit einem Titel, den es nicht gibt

1. Ergänzen Sie den Code in der `main`-Methode. (alles andere sollte schon implementiert sein.)
   Die Ausgabe des Programms muss in etwa wie folgt aussehen:
   

Dauer

3 - 5 Stunden

Abgabe

Pushen Sie - wie in der Anleitung erwähnt - jede lauffähige Version mit den jeweiligen Tests.
