Title: Building a Library Catalog System Once upon a time in a quaint college town, there was a diligent computer science student named Alex. Alex was passionate about Object-Oriented Programming (OOP) and was excited to apply the principles they had learned in class to a real-world project. Their latest challenge was to create a library catalog system. 1. Understanding the Problem Alex began their journey by carefully analyzing the problem. They needed to create a system to manage books in a library. The system should keep track of various aspects of each book, such as title, author, publication year, and available copies. To do this, they first decided to create a class diagram to represent the objects they would need. 2. Designing the Class Diagram Alex pulled out a blank sheet of paper and started sketching a class diagram. They identified three main classes: - Library: This class would represent the library itself. It would have attributes like name and location. - Book: This class would represent individual books. It would have attributes like title, author, publication year, and a list of copies. Alex also included methods for checking out and returning a book. - LibraryMember: To handle library members, Alex decided to create another class. It would have attributes like name, address, and a list of books currently checked out. Alex then established relationships between these classes. The library class would have a composition relationship with Book, meaning it would contain instances of the Book class. The library class would also have a relationship with the LibraryMember class for managing member information. 3. Implementing the Code With the class diagram in hand, Alex started writing code for their library catalog system. They decided to use Python for the project. Task. (Note: Screenshot your diagrams you have drawn and the code you have written and attach them in a Word document, plus your code. Both your diagrams and Code should be saved as an image after taking a screenshot and attached in a word document that must be uploaded to clanned.) Questions. 1. Based on Alex’s method of solving the above problem of developing a library catalog system. Draw the Object and Class diagrams to represent his solution. (10 Marks) 2. Write the Python code that implements the Object and Class diagrams you have designed above. (10 Marks) END

As a text-based AI model, I'm unable to create and attach images or Word documents. However, I can help you create a textual representation of the class and object diagrams based on the problem description and provide you with a Python code outline for a library catalog system. You can then use software to create the diagrams and implement the code.

Let's start with the textual representation of the class and object diagrams:

**Class Diagram:**

1. **Library Class**
   * Attributes: name, location
   * Associations: Contains multiple instances of the Book class
   * Aggregations: Manages multiple instances of the LibraryMember class
2. **Book Class**
   * Attributes: title, author, publication year, list of copies
   * Methods: check\_out(), return\_book()
3. **LibraryMember Class**
   * Attributes: name, address
   * Associations: Manages multiple instances of the Book class

**Object Diagram:**

Now, let's represent a sample instance of the classes:

* **Library Instance**
  + name: "Quaint College Library"
  + location: "College Town"
  + Contains multiple Book Instances
  + Manages multiple LibraryMember Instances
* **Book Instances**
  + Title: "Introduction to Programming"
    - Author: "John Smith"
    - Publication Year: 2020
    - List of Copies: [Copy 1, Copy 2, Copy 3]
  + Title: "Data Structures and Algorithms"
    - Author: "Alice Johnson"
    - Publication Year: 2019
    - List of Copies: [Copy 1, Copy 2]
  + Title: "Python for Beginners"
    - Author: "Eva Davis"
    - Publication Year: 2021
    - List of Copies: [Copy 1]
* **LibraryMember Instance**
  + Name: "Alex"
  + Address: "123 Main Street"
  + Manages a list of Book Instances (books checked out by the member)

**Python Code:**

Below is a Python code outline that implements the described class diagram:

class Book:

def \_\_init\_\_(self, title, author, publication\_year, copies=[]):

self.title = title

self.author = author

self.publication\_year = publication\_year

self.copies = copies

def check\_out(self):

if len(self.copies) > 0:

return self.copies.pop()

def return\_book(self, copy):

self.copies.append(copy)

class Library:

def \_\_init\_\_(self, name, location):

self.name = name

self.location = location

self.books = []

def add\_book(self, book):

self.books.append(book)

class LibraryMember:

def \_\_init\_\_(self, name, address):

self.name = name

self.address = address

self.checked\_out\_books = []

def check\_out\_book(self, book):

copy = book.check\_out()

if copy:

self.checked\_out\_books.append(copy)

def return\_book(self, book, copy):

book.return\_book(copy)

self.checked\_out\_books.remove(copy)

# Example Usage

book1 = Book("Introduction to Programming", "John Smith", 2020, ["Copy 1", "Copy 2", "Copy 3"])

book2 = Book("Data Structures and Algorithms", "Alice Johnson", 2019, ["Copy 1", "Copy 2"])

book3 = Book("Python for Beginners", "Eva Davis", 2021, ["Copy 1"])

library = Library("Quaint College Library", "College Town")

library.add\_book(book1)

library.add\_book(book2)

library.add\_book(book3)

member = LibraryMember("Alex", "123 Main Street")

member.check\_out\_book(book1)

print(f"{member.name} has checked out: {', '.join(member.checked\_out\_books)}")