Outline

1. Object-Oriented Design
   - unified modeling language
   - managing a library
2. Object-Oriented Programming in Python
   - class definitions and instantiations
   - data and functional attributes

MCS 260 Lecture 26
Introduction to Computer Science
Jan Verschelde, 12 July 2023

object-oriented design
OOP in Python

1. Object-Oriented Design
   - unified modeling language
   - managing a library
2. Object-Oriented Programming in Python
   - class definitions and instantiations
   - data and functional attributes

Object-Oriented Design

UML: Unified Modeling Language

Object-Oriented Programming (OOP) enables us to create our own high level data types, called abstract data types.

Real-world entities (such as books, people) are represented in the software by objects and classes.

UML is a graphical language to model, design and construct object-oriented software.

Two types of modeling diagrams:

1. structural ones define the static architecture;
2. behavioral ones captures interactions and states.

Running example: library management system.

object-oriented design
OOP in Python

1. Object-Oriented Design
   - unified modeling language
   - managing a library
2. Object-Oriented Programming in Python
   - class definitions and instantiations
   - data and functional attributes

Managing a Library

a case study

Goal: manage a library of books.

Two types of users: librarians and patrons.

Patrons when logged on may view the catalog,
check out books, and return books.

After logging in, in addition to what is available to all,
a librarian may

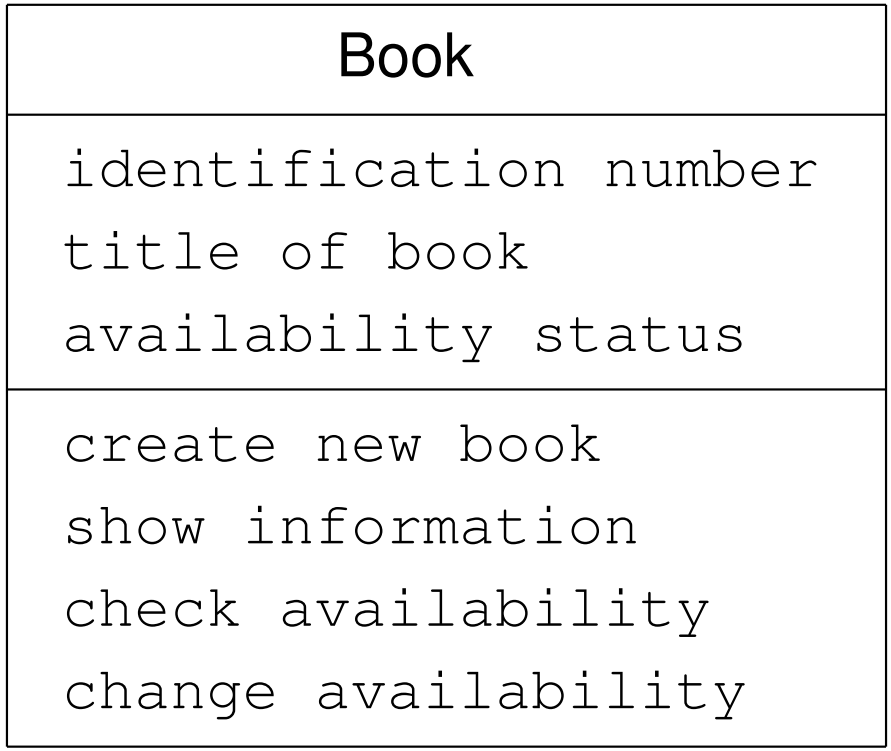
1. add and delete books;
2. add, search, and delete persons.

Still very simple management:
only one person uses the program at any given time.

the class Book

class diagram

An object of the class Book has three attributes:
`identification number`, `title`, `availability`.



Four methods: `__init__()`, `__str__()`, `check()`, `change()`.

the class Person

class diagram

An object of the class Person has three attributes:
`identification number`, `name`, `status`.

Person

`identification number`
`name of person`
`status: librarian?`

`create new person`
`show information`
`check status`
`change status`

Four methods: `__init__()`, `__str__()`, `check()`, `change()`.

the class Catalog

class diagram

The collection of books is an object of the class Catalog.
Its one attribute `collection` is a list of books.

Catalog
`list of books`

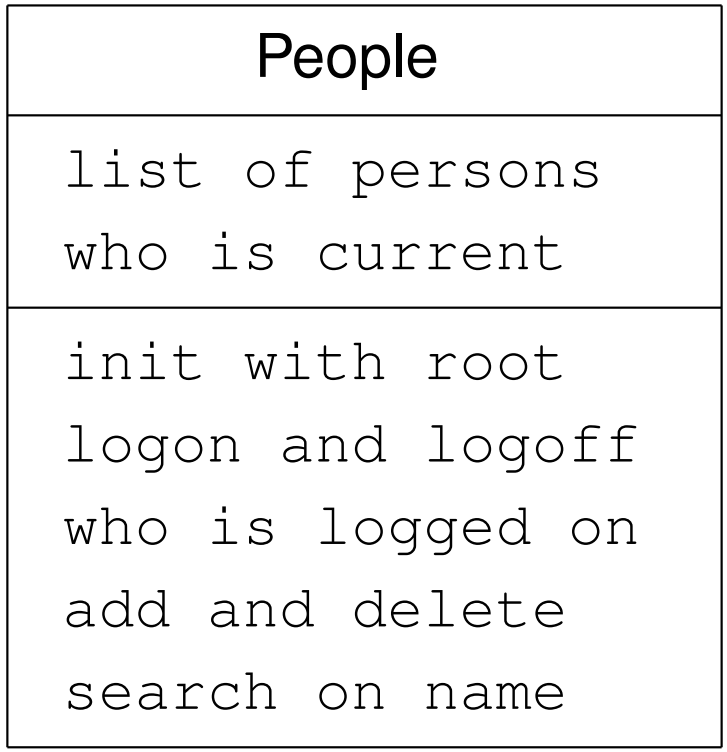
`add and delete`
`checkin and checkout`
`show the collection`
`search on key`

In addition to `__init__()` and `__str__()` we have five methods:
`add()`, `delete()`, `checkin()`, `checkout()`, and `search()`.
The class Catalog imports from the class Book.

the class People

class diagram

An object of the class People has a list as first attribute.
Its second attribute is who is currently logged on.

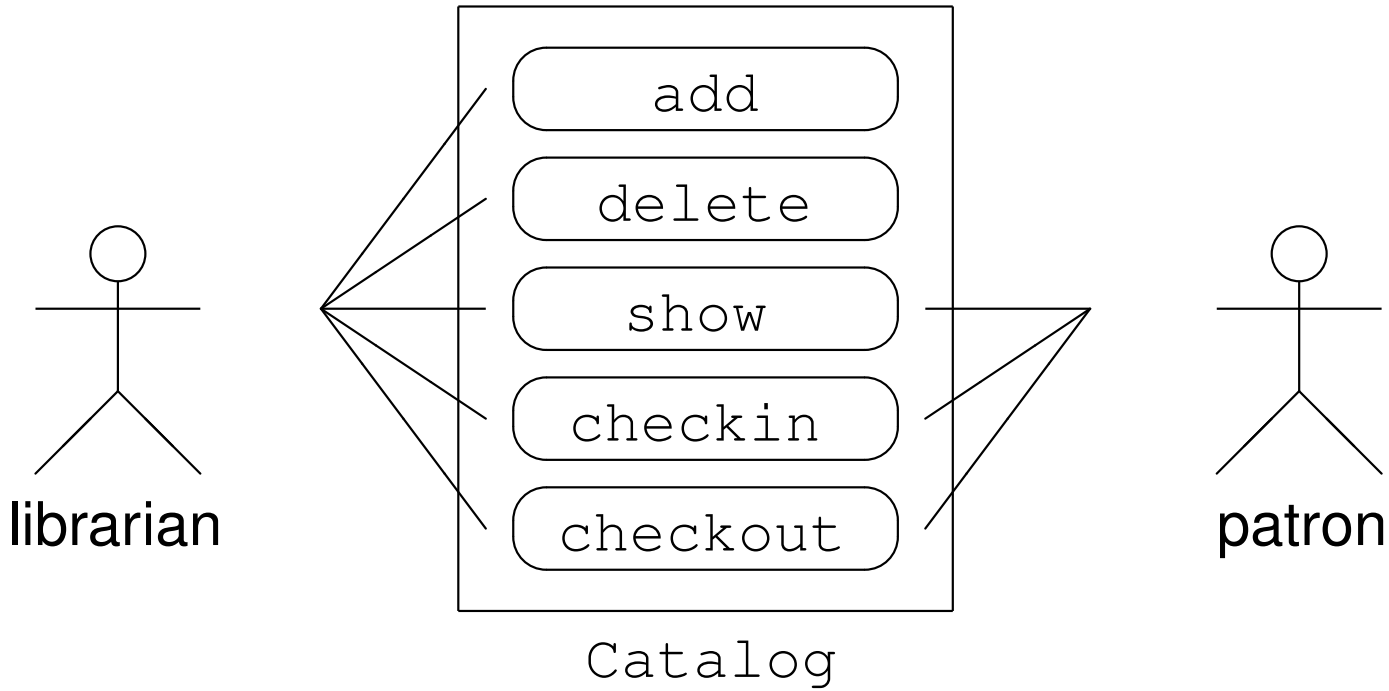


seven methods: `init()`, `logon()`, `logoff()`, `who()`, `add()`, `delete()` and `search()`.

Use Case Diagram for Catalog

a behavior modeling diagram

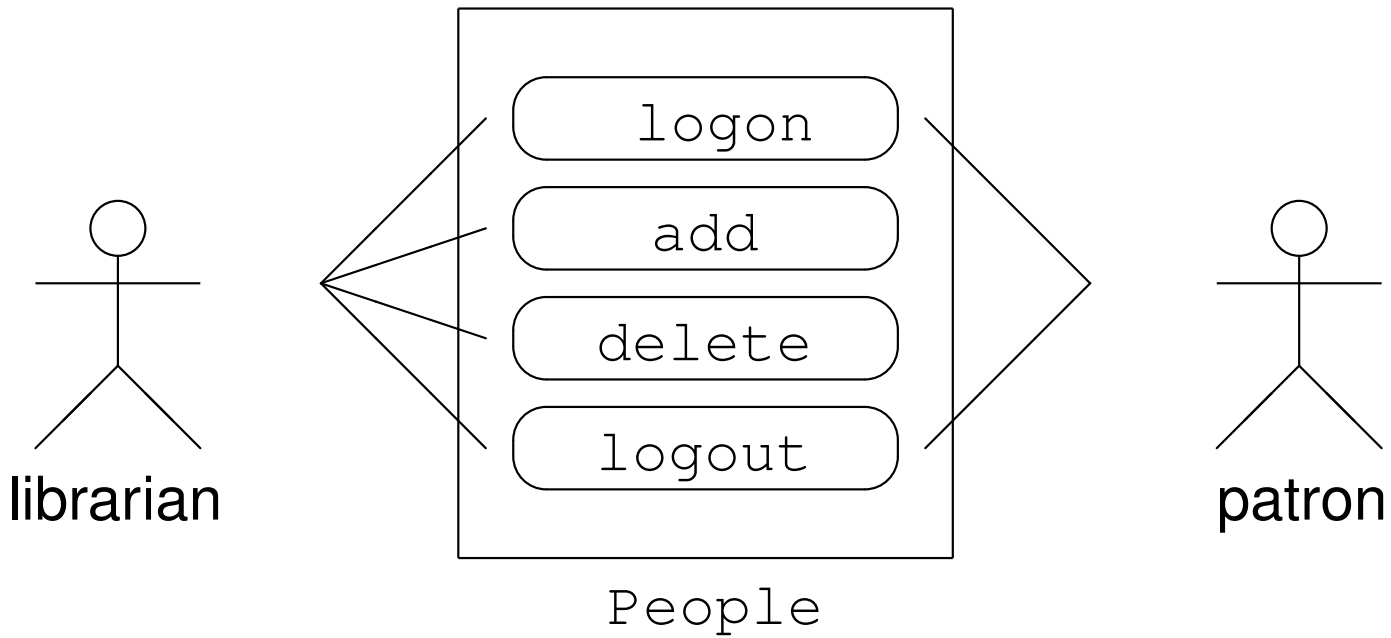
Librarians and patrons differ in their use of the Catalog:



Use Case Diagram for People

a behavior modeling diagram

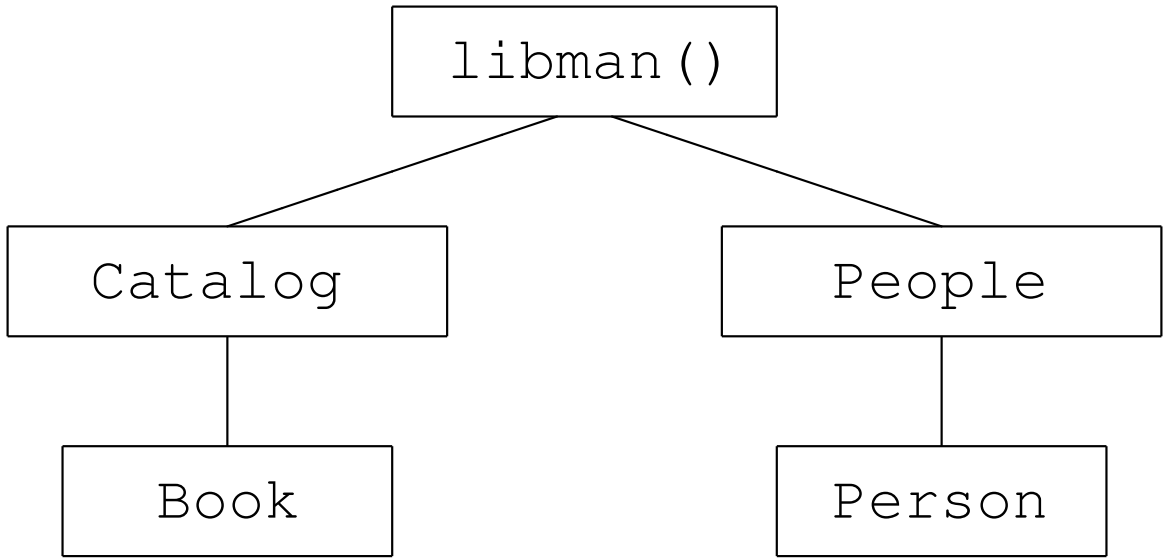
Librarians and patrons differ in their use of the People:



Design of a Library Manager

OOP follows bottom up design

Object-oriented design is typically bottom up, starting at the classes `Book` and `Person`.



The program libman() imports from `Catalog` and `People`.
The class `Catalog` imports from `Book`
and the class `People` imports from `Person`.

object-oriented design
OOP in Python

1. Object-Oriented Design
   - unified modeling language
   - managing a library

2. Object-Oriented Programming in Python
   - class definitions and instantiations
   - data and functional attributes

object-oriented programming

Definition (Grady Booch et al., 2007)

Object-oriented programming is a method of implementation in which

1. programs are organized as cooperative collections of ***objects***,
2. each of which represents an instance of some ***class***,
3. and whose classes are all members of a ***hierarchy*** of classes united via inheritance relationships.

Objects — not algorithms — are the building blocks.

Algorithms are central in procedure-oriented programming.

Definition from page 41 on *Object-Oriented Analysis and Design With Applications* by G. Booch et al., Addison-Wesley, 2007.

object-oriented design
OOP in Python

1. Object-Oriented Design
   - unified modeling language
   - managing a library

2. Object-Oriented Programming in Python
   - class definitions and instantiations
   - data and functional attributes

data and functional attributes

We distinguish between data and functional attributes:

1. data: information represented by the object; and
2. functional (often called methods):
   the operations defined on the objects.

A data attribute can be *class wide*,
that is: shared by every instance in the class.

Example: to give every object a unique identification number,
every object shares the same reference to the counter.

the `self` argument

Example: To sort the elements of the list `L`,
we apply the method `sort` as `L.sort()`.

The `self` is a reserved word in Python.

When defining any method, the `self` refers to the object to which the method applies.

Exercises

1. Make a class `Counter` which initializes to zero. The method `add` increments the counter by one. The string representation returns the value of the counter, that is: the value of the data attribute stored by the object instantiated from the class `Counter`.
2. Design a class `Rational` to compute with rational numbers. Ensure that a rational number is always normalized: numerator and denominator have 1 as their only common divisor.
3. Write Python code for the class `Rational`.
4. Describe how the design of our library manager would change if files would be used for the catalog and people.
   Which functions would change?