COMPSCI 230 Tutorial 2
Basic OOP/OOD –ArgoUML, Use cases and Class Diagrams

Welcome back. This tutorial will go over a quick introduction to ArgoUML and examples of how to use class diagrams and use-case diagrams.

Please ask questions if there is something on this sheet you want clarified.

Before we begin see if you can answer Question 1 below.

Q1: Remind ourselves of what the following keywords for OO in java mean:

public	Can be accessed outside the class (anywhere).
private	Can only be accessed inside the class it is declared in.
protected	Can be accessed by child classes.
static	Is shared by all instances of the declaring class

Design (Scenario): A new library has opened up on campus. They require a basic application to manage the books the library owns and the patrons who use the library. The library also houses valuable reference books that patrons can view in the library but are unable to borrow.

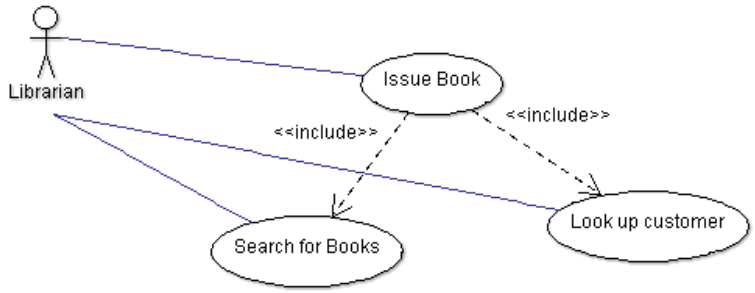
The system will have a database of all books that anybody is able to search, this should include author, title, call number and availability.

The Librarians will be able to maintain the database of users, and issue books.

Use-Case Diagrams

The library has commissioned you to finish their system. Their last employee was not very reliable and left everything unfinished. It is up to you to finish the job!

Examine the diagram below and answer the following questions.



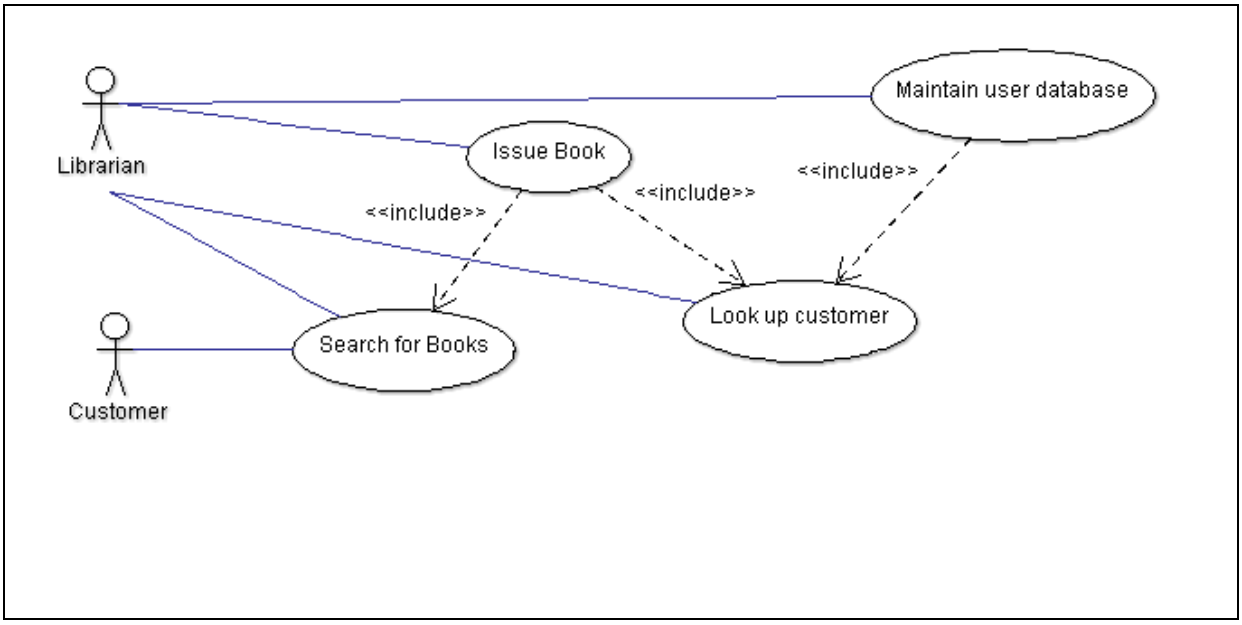
Q2: What does <<include>> mean? Give a brief explanation of why each <<include>> might be needed for the Issue Book use case.

<<Include>> means that when we are performing the use case, we must also perform the use case included.
1) To update the book record to show it has been borrowed, we must find the book record in the database.
2) To issue the book to a customer, we must look up the customer to add the book to their account.

Q3: The previous designer forgot completely about the library customers. Which of the above use cases should the customers be able to use. (**Hint:** In the scenario the customers are called patrons)

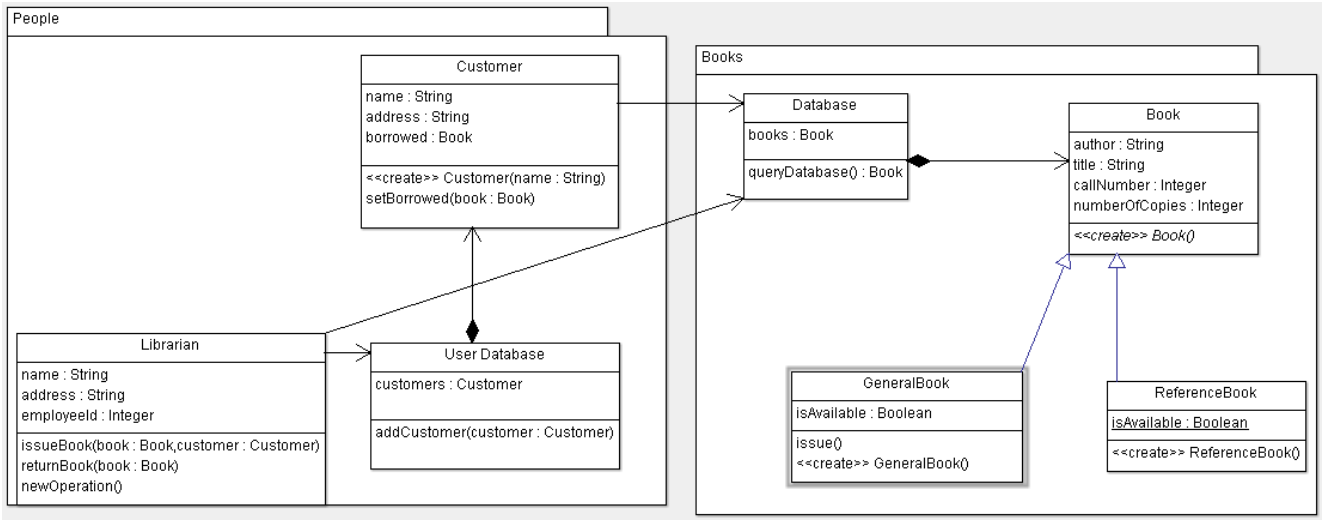
Search for Books

Q4: Currently the librarian is unable to maintain the user database. Add **ONE** Use-Case to this diagram to add that function. Be sure to add in any associations you feel is needed for this. Also include the changes described in your answer to **Q3.**

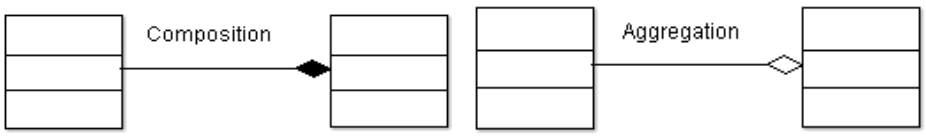


Class Diagrams

The library is happy with your new use case diagram and asks you to go ahead with the project. They ask you to make a class diagram to show how the system will work. Some of this has already been completed by the previous employee.



Q5: What is the difference between composition and aggregation?

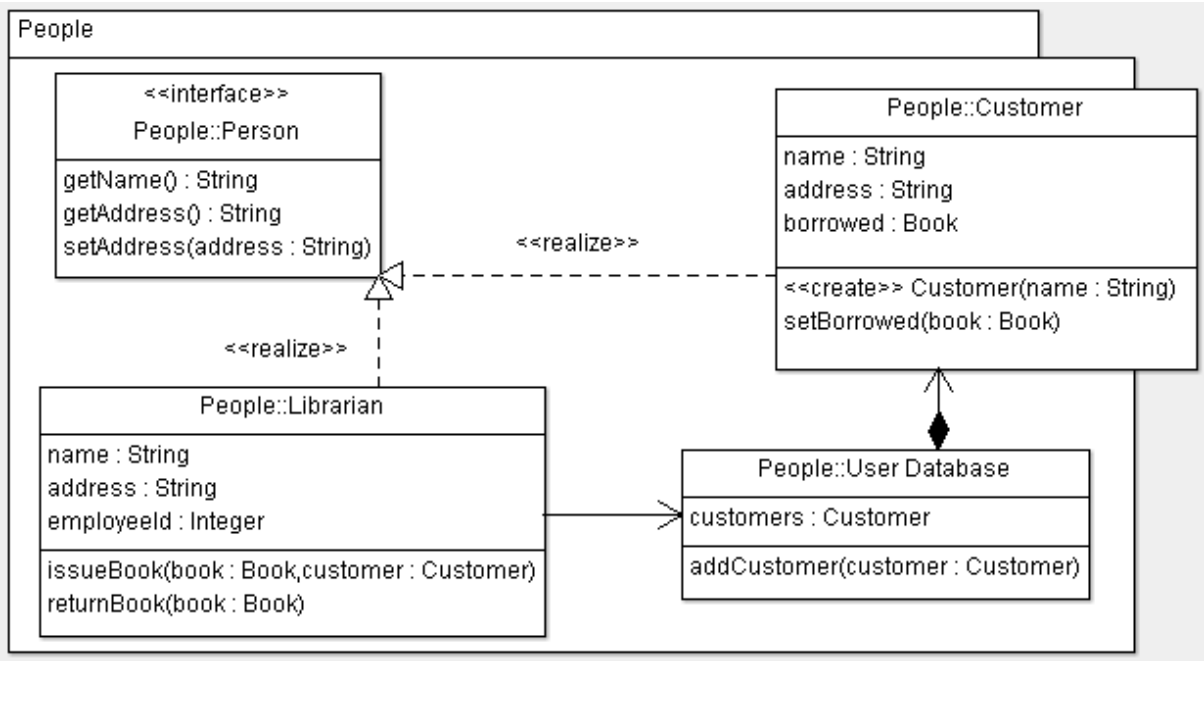


Composition means that classA owns-a classB. i.e. ClassB belongs to ONLY one instance of classA. (E.g. A sim card has a SIM)
Aggregation means that classA has-a classB. i.e. ClassB belongs to an instance of classA, but may also belong to another instance of classA or another class.
(E.g. One engine model may belong to many different car models.)

Q6: What is an interface?

An interface is a special class that defines the behaviour of any class that implements it. This is achieved by defining methods that must be created, including inputs and output for that method.

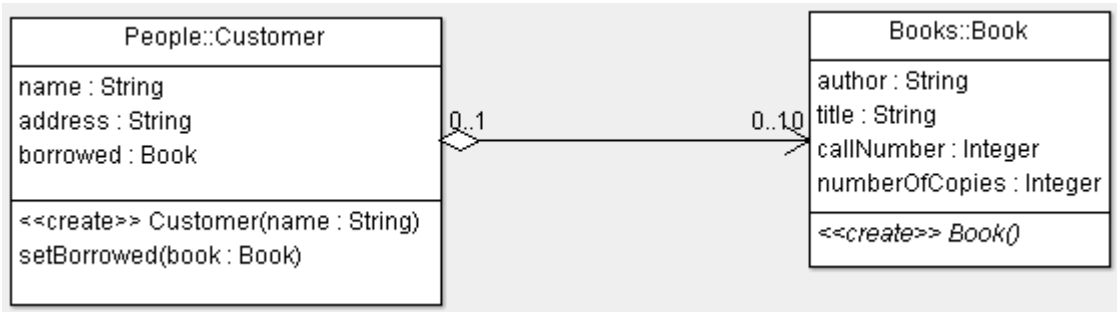
Q7: The library wants to be able to read the names and addresses, and update the addresses of all customers and librarians using the same interface. Add an interface to the class diagram part below to achieve this.



Q8: Complete the skeleton code (i.e the methods do nothing) of the Customer class. Pay close attention to the inheritance.

```
public class Customer implements Person {

    public Customer(String name){}


    public void setBorrowed(Book book){}

    public String getName(){}

    public String getAddress(){}

    public void setAddress(String address){}









}
```

Q9: Draw an aggregation between Customer and Book to indicate which books the Customer has borrowed. Indicate that the customer can borrow a maximum of 10 books.



Q10: Notice the relationship between GeneralBook and Book. What type of relationship is this? Explain briefly what the relationship means.

Inheritance.
An inhertiance relationship means that one class is a more specialised version of another. This specialised version has all the attributes of the general class and can be used in code as one of the general class objects.
(E.g. A car is a specialised version of a vehicle.)

Q11: Fill in the underlined spaces in the code to complete part of the GeneralBook class.

```
public class GeneralBook extends Book {

      public GeneralBook(String author, String title, int callNumber){

            super(author, title, callNumber);

      }

}
```