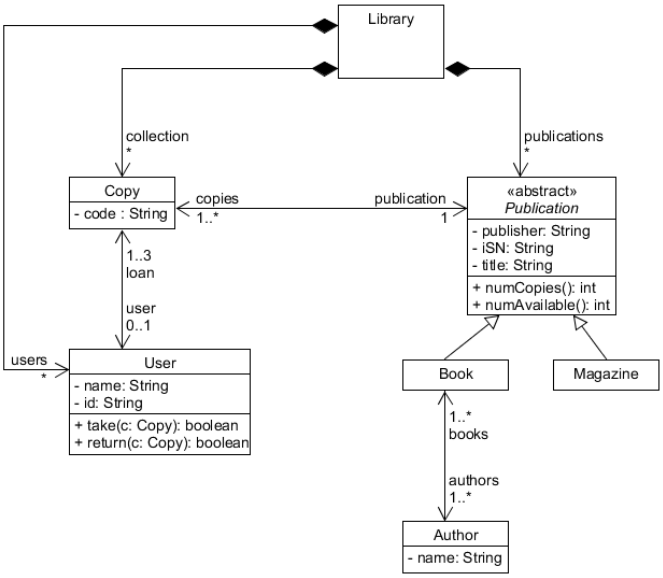
Lesson 2 Exercises

Object Orientation

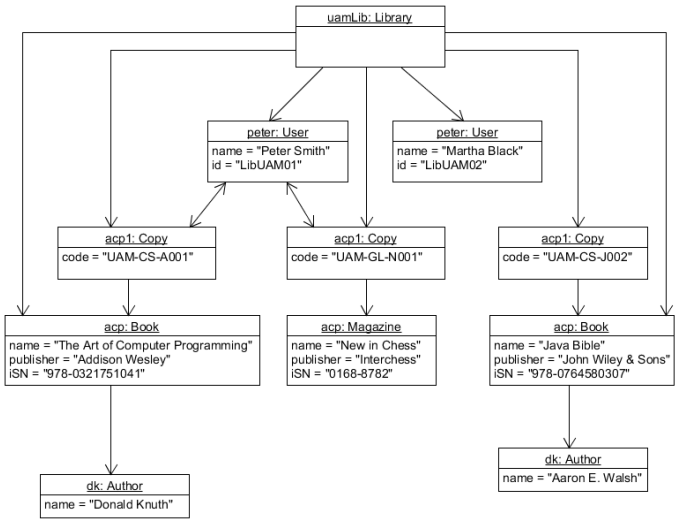
Start: Week of February 3rd.
Duration: 2 weeks.

1) Create the class diagram of the following application for a library. The library has two types of publications: magazines and books. Both are edited by some publisher. While the books have an ISBN, a title and (one or more) authors, the journals have an ISSN and a name. The library has one or more copies of the publications conforming its bibliographic collection. The copies are classified according to a code. Users of the library can take a maximum of three publications at a time.

   Add methods to the appropriate classes to: (i) allow users return or take publications, as well as to access the publications they have borrowed, (ii) given a publication, obtain the number of copies available to the library, and the number of them that are not borrowed.



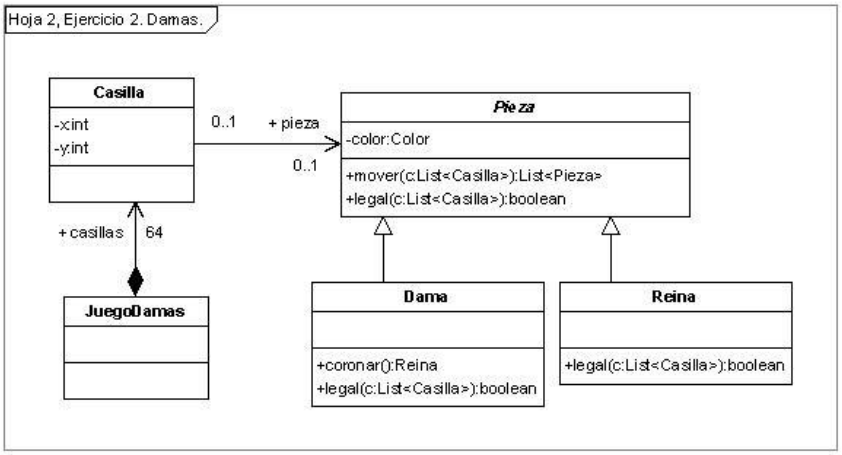
2) Based on the class diagram of exercise 1, create an object diagram that contains at least two users, two books, and one magazine; and assume one of the users has borrowed a book and a magazine.



3) Build a class diagram that models the *checkers* game. This is a game for two players that takes place on a board of 8x8 squares. The pieces have a color (black or red) and can be of two types: *men* and *kings*. While men move one square diagonally, kings can move several squares at once (always diagonally, and also backwards). Upon reaching the end of the board, a man promotes himself as king. A movement can also capture a piece of the adversary, and it is possible to chain several captures in a single movement of a piece.

   Add methods to the appropriate classes to: (i) move a piece, (ii) check if a move is legal, (iii) promote a man.

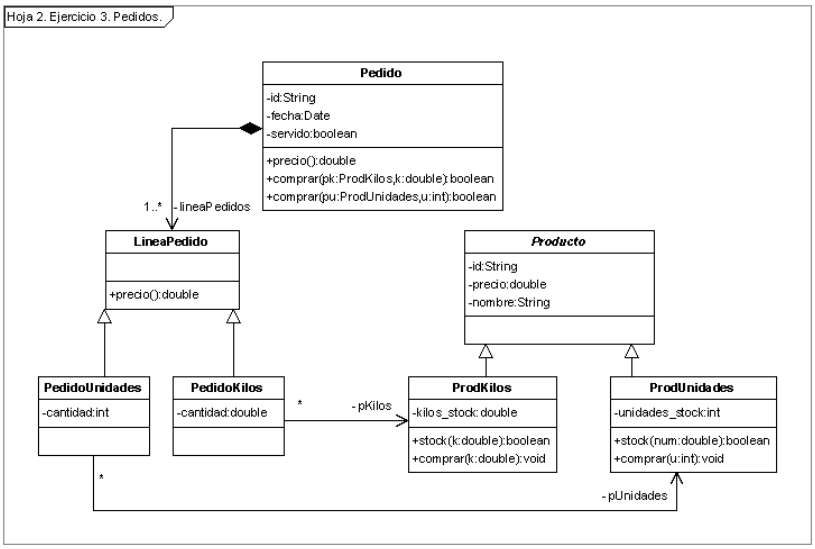
(In Spanish)



4) Build the class diagram for an information system to manage orders. There are two types of products that can be ordered: those that are sold by units, and those that are sold by weight. In either case, the products are described by an identifier, a name and a price (per unit or per kilo). An order consists of items, which contain the units or kilos that have been ordered for a particular product. Orders are described by an identifier, the date the order was placed, and an attribute indicating whether the order has been served or not.

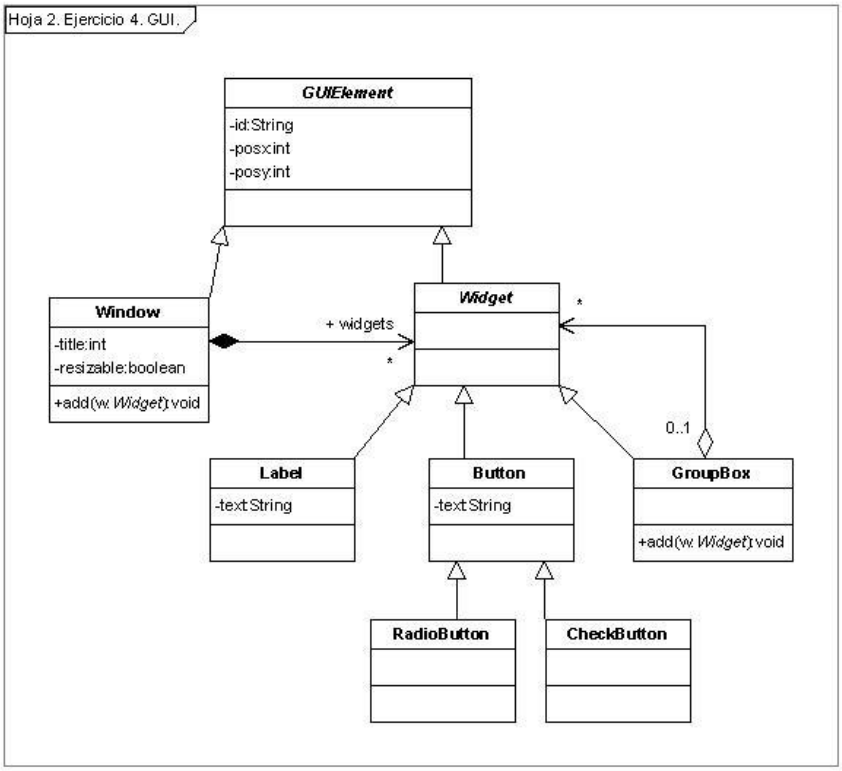
   Add methods to the appropriate classes to: (i) calculate the price of an order, (ii) add to an order the purchase of certain units or kilos of a given product. This method should check if there is stock of the product to be added, returning if the operation could be performed or not.

(In Spanish)

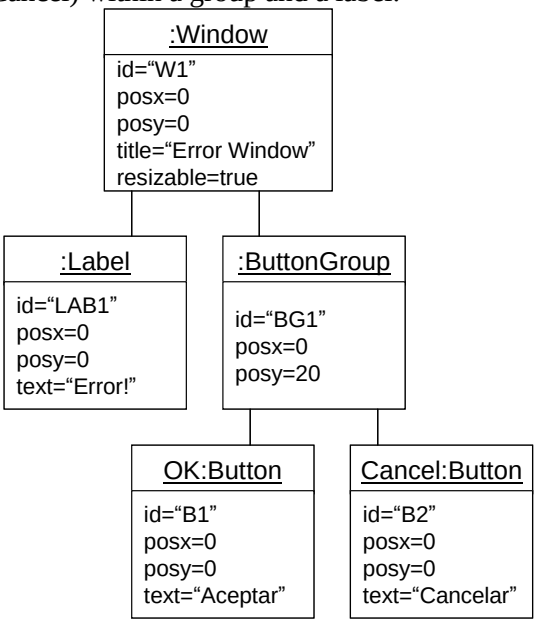


5) Build the class diagram for an application that allows the definition of simple graphical user interfaces (GUIs). All elements of an interface contain an identifier and a position (x and y coordinates). Windows have a title and may or may not be resizable. In addition, they can contain zero or more elements of type Label, Button, or group (GroupBox). Labels and buttons have a text. Buttons can also be normal, radio-buttons or allow selection (CheckButton). A GroupBox can contain labels, buttons and other GroupBoxes.

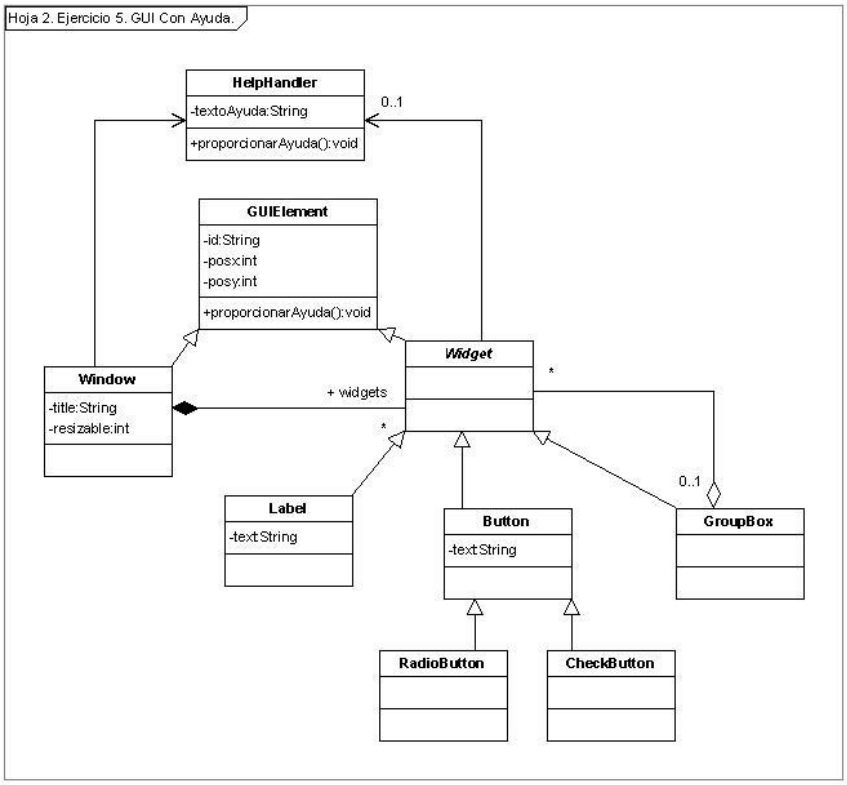
   Add methods to the appropriate classes to: (i) add an item to a window, and (ii) add an item to a GroupBox.



6) Based on the class diagram in Exercise 5, construct an object diagram that contains a window with two buttons (Accept and Cancel) within a group and a label.



7) Modify the class diagram of exercise 5 to add a contextual help system, which when activated, gives help on the specific element that has the focus at that time. If an element of the interface is not assigned any help, then the help of the container element must be displayed, if it exists. The window, being the highest level element, has always a contextual help assigned.

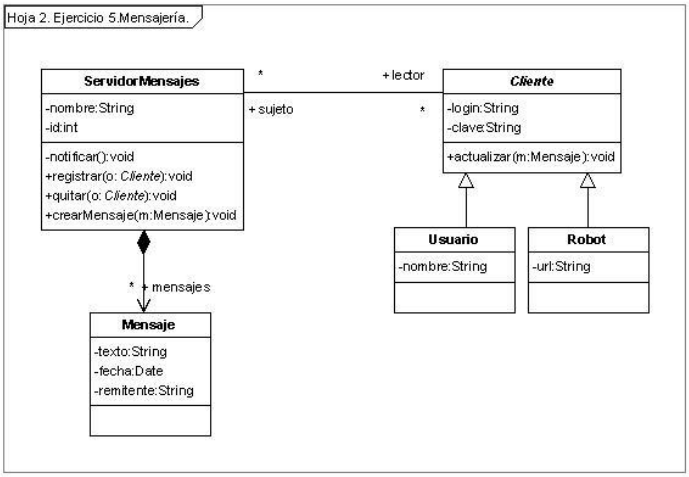


Remark: It is similar to the "chain of responsibility" design pattern. We must be able to navigate the composition links, which must be bidirectional.

8) Build the class diagram of a messaging application. The messages are made of a text, a sender and a date. Messages are added to a messaging server, which is described by a name and an identifier. Different clients can register on a message server, to read its content. There are two types of customers: users (people) and bots (applications). The former have a name and the latter a URL where they display the messages automatically. Both types of client have a login and password.

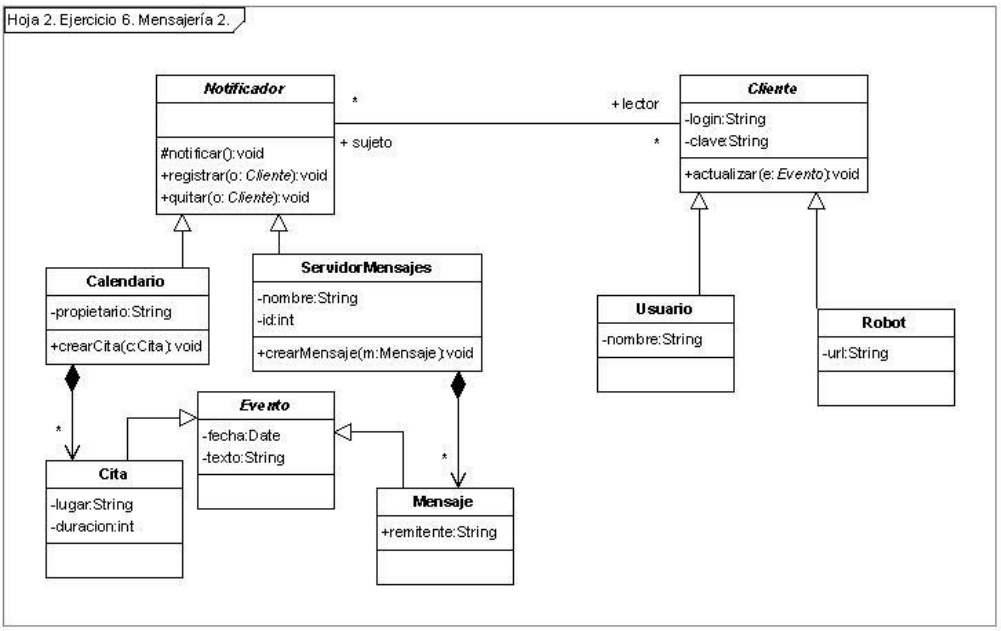
Add methods to the appropriate classes to: (i) allow customers register to and delete from a server, (ii) allow messages be created on a server, and permit the server notify all registered customers about the arrival of a new message.

(In Spanish)



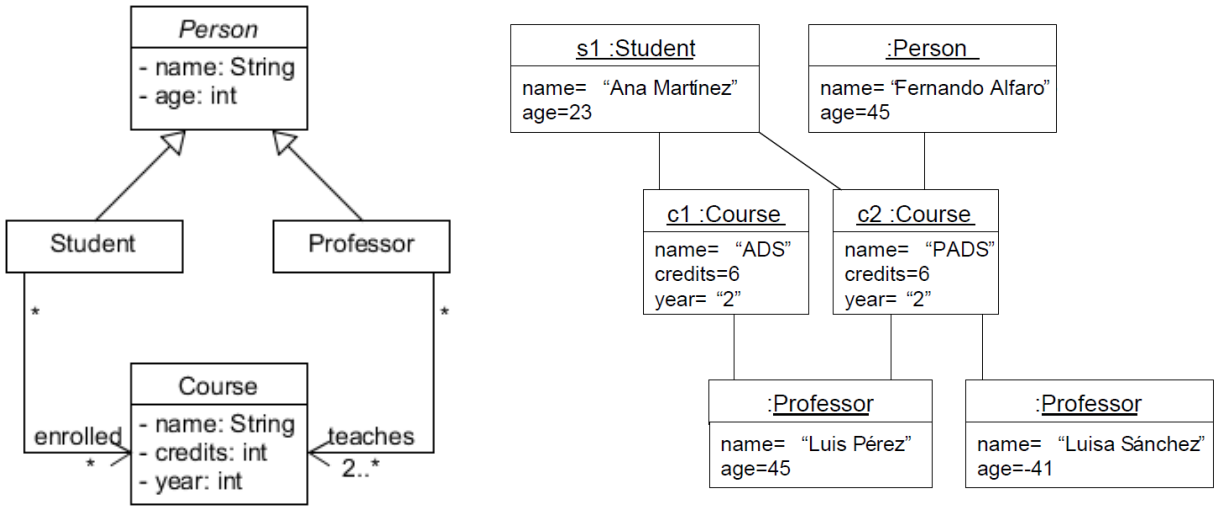
Remark: This solution is similar to the Observer design pattern.

9) Modify the previous design to also consider a calendar service. In this service, appointments can be added, which are formed by a date, place, duration and a descriptive text. We want customers be able to access the calendar service and the message service in a uniform way.



Remark: This is an Observer design pattern.

10) Given the class diagram to the left, find the errors (if any) of the object diagram to the right.



Class Person is abstract, and we cannot connect it with Course
Attribute year has integer type.
The minimum cardinality of "teaches" is 2.

11) We want to design an application for editing documents. The documents can be of two types: plain text or graphical. Any type of document must store the file name, the language the document is written in, and the author. A graphical document is made of pages, which have a header and one footer (which may be different for each page). The pages in turn can contain images, sounds, text fragments, or have embedded documents (of either type). An image is described by the file name, its type (JPEG, GIF, PNG or BMP), size and position. A sound is described by the file name and its type (WAV, MP3 or WMA), and a text fragment by a file name and the text it contains. A plain text document does not contain pages, but is directly made by text fragments.

a) Build the class diagram modelling the previous description.
b) Add methods to implement the following requirements:
   1. When a page is open in the editor, this should show all the components it has (images, sound, text fragments and other documents).
   2. By default, the documents inside a page are shown as an icon, displaying the file name. Documents can be opened, showing their contents.
   3. Individual fragments can be justified, as well as all fragments of the same page.