# Lesson 19 – Final Project

**10h**

* Use previous knowledge to create your final Project.

What students should know

**Dear Teacher**

The solution proposed relies on knowledge gained during the course. You can extend the questions as you want depending on your class level and available time.

To study how a school library can be created you can read the book "Setting Up and Running a School Library by Nicola Baird" here https[://files.Eric.Ed.](https://files.eric.ed.gov/fulltext/ED536911.pdf) [Gov/Fulltext/Ed536911.Pdf](https://files.eric.ed.gov/fulltext/ED536911.pdf)

The names of the books and authors come from the Gutenberg Project <https://www.gutenberg.org/ebooks/offline_catalogs.html>

Publishers' names and release dates are random as well as students' names.

*Prokopis Leon, pliroforikos[at]gmail.com*



## Create a School Library Application

A school library has a set of books which it lends to students in the three classes. The books are placed on shelves numbered from 1 to 60.

For each book, the following information is recorded:

* Code
* Title
* Author
* Release Year
* Publisher
* Shelf

For students, the following are recorded:

* Registry Number
* Name
* Surname
* Class
* Phone Number

Make the application described below:

## Files

Two text files named are given books.txt and students.txt (inside the materials folder).

Some of the file elements are:

**books.txt**

30;The Life of Abraham Lincoln;Henry Ketcham;1866;New Public publ.;54

31;Christopher Columbus; Mildred Stapley;1954;Cider publ.;43

32;The Adventures of Ferdinand Count Fathom; Tobias Smollett;1982;Orange punl.;32

33;Tales of the Jazz Age;F. Scott Fitzgerald;1944; Gutenberg publ.;5

34;The Old Stone House;Anne March;1904;Orange punl.;50

Each book is described in the following fields which we distinguish from each other with the character ";":

* Id
* Title
* Author
* Year
* Publisher
* Shelf

**students.** **txt**

1001;Jude;Segers;A;7900209

1002;Desire;Cid;A;7047635

1003;Madelyn;Pittard;A;9011036

1004;Lorita;Tomczak;A;6677490

1005;Lynwood;Posey;A;9014379

1006;Nella;Felps;A;8423818

Each student is described in the following fields which stand out from each other with the character ';':

* Id
* First Name
* Last Name
* Class
* Phone Number

Implement a function that inserts the above text files into Maps with the names mapBooks and mapStudents as the keys in the first column of each file where it represents the Id of the book and the student respectively. The values of each item will be a Type as described below.

**Teachers tip**

Before you can read the files you need to read all the content like a string and then check its characters one by one and place them in a table of items that you will later write into the Maps. A string is checked using the <string> method. **CharAt**(index).



**Example Of Student**  **Map**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Key** | **Type** | | | | |
| **Id** | **FirstName** | **LastName** | **Class** | **Phone** |
| 1001 | 1001 | Jude | Segers | A | 7900209 |
| 1002 | 1002 | Desire | Cid | A | 7047635 |
| 1003 | 1003 | Madelyn | Pittard | A | 9011036 |

## Data

### Types

1. Create Type Book

Type Book with elements:

* + - Id
    - Title
    - Author
    - Year
    - Publisher
    - Shelf

**Functions:**

* + Insert Book
  + Display/Select Book/s
  + Delete Book

1. Create Student Type

**Properties**:

* + Id
  + First Name
  + Last Name
  + Class
  + Phone Number

**Functions:**

* + Insert Student
  + Display / Select Student
  + Delete Student

Each student who has borrowed books has their own KVS file with name “ID number.dat”. For example, the student with ID 21 has a file named “21.dat”. The file writes the books borrowed by the student using a map structure. For example, if you have borrowed two books you should have a Map of the following structure:

(“11”, “03/27/2021”)

(“14”, “04/01/2021”)

Where the first number is the ID of the book borrowed and is also the key of the map while the borrowing date is saved as a value.

## Screen Design and Functions.

The program must have a central menu of buttons to call appropriate administrative screens. Specific:

1. **Book.**

Includes a CLV list (CustomListView) of library books to be loaded from the books.txt and book insert and delete buttons.

When the Insert button is pressed, create a dialog asking for new book items to be inserted into the file.

When the Delete button is pressed, delete the book selected from the list.

1. **Student**

Includes a CLV list of school students that will be loaded from the students.txt file and insert and delete buttons.

When the Insert button is pressed, create a dialog asking for new student information and store it into students.txt.

When the Delete button is pressed, delete that student from the list together the books they borrowed assuming they have returned them.

1. **Lending**

The lending screen includes a ComboBox containing the student's name and a CLV list of library books.   
There is also a button Lend when its’ click triggered:

**If there is a student's file**

It will load the entire map from the file

It will add a new key with the ID of the book borrowed and set the date variable for the borrowing date.

It will re-save the map

**If there is no student file**

It will create the file

It creates a Map with the borrowing information(ID and date)

It will save the file

**Consider that there are as many copies as you need from each book.**

1. **Return**

The lending screen includes a B4XcomboBox containing the student's names and a CLV list of books borrowed. Also, there is a button called “Return”.

If return\_click triggered then it will delete this book from the student's list as well as from the student's file.

If he hasn't borrowed another book, program will delete and lending record file too.

## Cheat Sheet

1. How to declare a type

**Sub** Class\_Globals

**Type** Student(LastName  **As String**, FirstName **As String**, \_

Address **As String**, PhoneNumber  **As String**)

**Private** Student1 **As Student**

**End Sub**

1. How to read a file on a string

stFile = File.**ReadString**(File.**DirAssets**, "students.txt")

1. Check character by character a string that contains a info and place it in a list of tables.

**Private** i **As Int** = 0

**For** j = 0 **To** stFile.**Length**-1

**If** stFile.**CharAt**(j) <> ";" And stFile.**CharAt**(j) <> CRLF **Then**

stud(i) = stud(i) & stFile.CharAt(j)

**Else if** stFile.CharAt(j) = ";" **Then**

i = i + 1

**else if** stFile.CharAt(j) = CRLF **Then**

i = 0

retList.**Add**(stud)

**Private** stud(5) **As String**

**End If**

**Next**

1. Convert a list of tables to a map

For i = 0 To lst.**Size**-1

**Private** stud(5) **As String**

stud = lst.**Get**(i) ‘ Create an array from list item

**Private** st  **As Student** ‘ Student is type declaration

st.**Initialize**

st.ID = stud(0) ‘Put every array item into type

St.FirstName = stud(1)

St.LastName = stud(2)

St.Cls = stud(3)

St.Phone = stud(4)

St.Borrowed = 0

mStudent.**Put**(stud(0), st) ‘ Insert type into map

**Next**

1. Place spaces In as string to increase its length (size) up to a number.

**Do While** s1.Length <= 5

s1 = s1 & " "

**Loop**

1. Select – Deselect an item from a CLV List.

Each item in a list is created by the language within a box called a Panel. You can set the color by accessing the box properties using the GetPanel(Index) method where Index the current value of the line that was clicked then:

* If an item in the list is clicked, the subroutine checks the value of selectedItem, and if that is -1 then sets a Blue background color in the clicked line and sets selectedItem to the Index value that gave the \_ItemClick event
* If selectedItem already has a value, a white color is set as the background in the box, and then there are two cases
* Click the already selected item, in which case the item stops being selected and selectedItem becomes -1
* Click another item in which case selectedItem gets the value of index.

**Private Sub** clvBooks\_ItemClick (Index **As Int**, Value **As Object**)

**If** selectedBook = -1 **Then**

**Private** p **As B4XView** = clvBooks. **GetPanel**(Index)

p.GetView(0). Color = xui. Color\_Blue

selectedBook = Index

**Else**

**Private** p **As B4XVie**w = clvBooks. **GetPanel**(selectedBook)

p.GetView(0). **Color** = xui. Color\_White

**If** selectedBook = Index  **Then**

selectedBook = -1

**Else**

**Private** p As **B4XView** = clvBooks. **GetPanel**(Index)

p.**GetView**(0). Color = xui. Color\_Blue

selectedBook = Index

**End If**

**End If**

**End Sub**

1. Load items from KVS File to map

The command must run within Wait For.

**Wait For**(StudentFIle.**GetMapAsync**(StudentFIle.**ListKeys**))**Complete**(mapSt **As Map**)

The GetMapAsync command returns a map with the items. First declare mapSt As Map and initialize it.

1. Save items to a map's KVS File

**Wait For** (StudentFIle.**PutMapAsync**(mapSt)) **Complete** (Succes  **As Boolean**)

1. Delete a file from a folder

File.**Delete**(File.**DirTemp**, <file name>)