Department of Computer Technology and Information Systems

**CTIS221 – Object Oriented Programming**

Fall 2019 - 2020

**Lab Guide 19 - Week 14-1**

| **OBJECTIVES:** General Review |
|---|
| **Instructor** : Burcu LİMAN<br>**Assistant** : Burcu ALPER, Leyla SEZER |

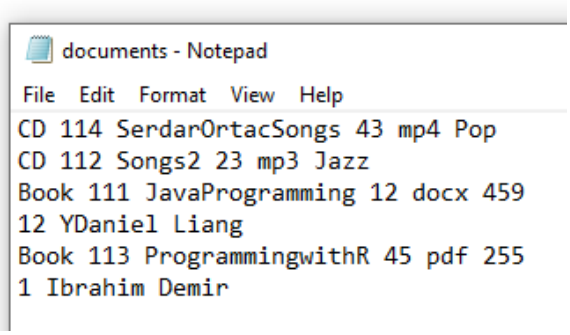
Write a Java program that reads Document information from a file named “documents.txt” and stores them into the related structures. The program, In part A, creates the related classes, in part B, creates a GUI for the related information and first gets the data from the file, then makes add and display operations.

**PART A: Implement your classes**

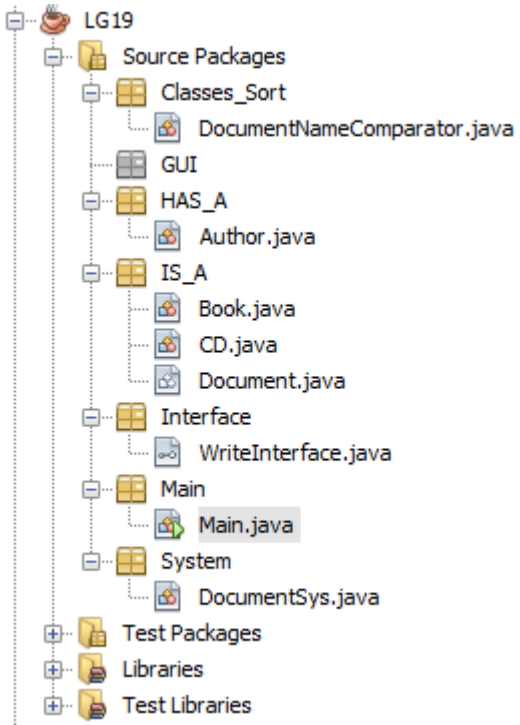
Your program will get input from 1 text file with the following structures;

The **documents.txt** file includes information about documents as; **document type**, **id, name, size, extension**

- If the type is **Book** ; get the total number of pages and an author object with his/her author id, author name and surname.
- If the type is **CD** ; get the genre,




You are going to create each class in separate java files. Packages and the file names are as follows;



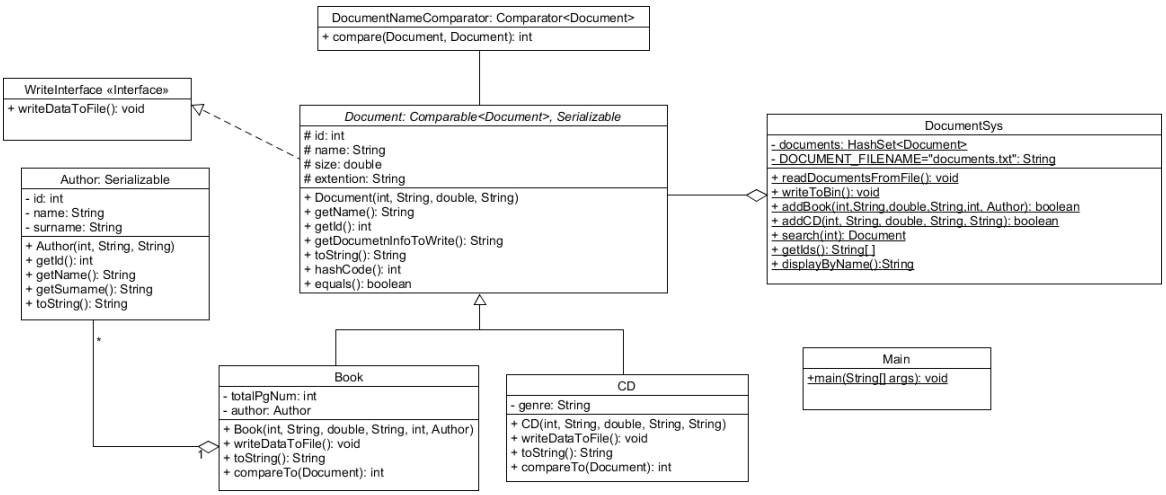
Check the UML class diagram, implement your classes according to it, **do not change** visibility modifiers.

➔ Write only the necessary accessor and mutator methods inside the classes!! You may add methods other than the given ones, if you are going to use them!!

➔ **toString()** method for all the classes: will return the necessary data field information **related with** the implemented class.

➔ There is a IS-A relationship between;
- Document - Book
- Document - CD

➔ There is a HAS-A relationship between;
- Book - Author
- Document - DocumentSys




***Information of the WriteInterface interface structure;***

- Write an abstract method **writeDataToFile ().**

***Information of the Author class structure;***

- Implement the given data members and methods according to the uml-class diagram.

***Information of the Document class structure;***

- Write a method named as **getDocumentInfoToWrite ()** that will return a string in the given format:
  `"id name size extension"`
- Write a **toString()** method.
- Write **hashCode()** and **equal()** methods for the id.

***Information of the Book class structure;***

- Write a **compareTo(..)** method that will take a Book object as a parameter and compares the book ids in ascending order.
- Write a method named **writeDataToFile()** that will append the book object to the "documents.txt" file with the given format. Use getDocumentInfoToWrite() method for super class members.
  `"Book id name size extension totalPgNum authorId authorName authorSurname"`

***Information of the CD class structure;***

- Write a **compareTo(..)** method that will take an CD object as a parameter and compares the cd ids in the ascending order.
- Write a method named **writeDataToFile ()** that will append the cd object to the "documents.txt" file with the given format. Use getDocumentInfoToWrite() method for super class members.
  `"CD id name size extension genre"`

***Information of the DocumentNameComparator class structure;***

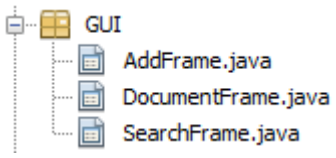
- Implement **compare(..)** method that compares document name of the objects.

**Hint:** There cannot be more than one object with the same document name .

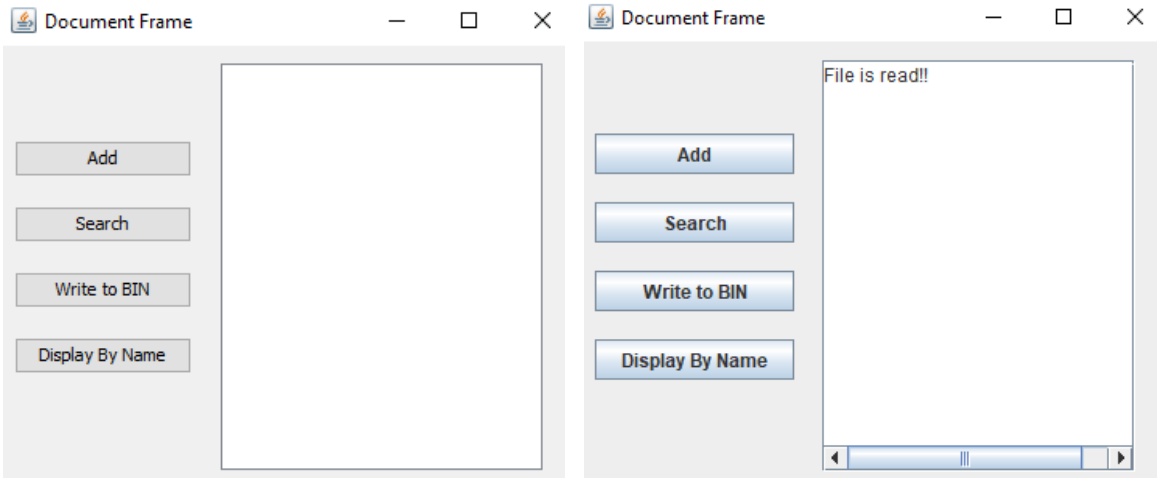
***Information of the DocumentSys class structure;***

- Create the data member;
  - A hashSet which stores Document objects
  - A constant variable to keep the file name.
- Create a **readDocumentsFromFile()** method that reads the content of the documents.txt into the documents HashSet.
- Create a **addBook(…)** method that gets a Book information as parameters. Then it creates an object and adds it to the documents HashSet. Finally write the object to the txt file by invoking the writeDataToFile(..) member method. Do not forget to check if the book object is exist or not.
- Create a **addCD(…)** method that gets a CD information as parameters. Then it creates an object and adds it to the documents HashSet. Finally write the object to the txt file by invoking the writeDataToFile (..) member method. Do not forget to check if the cd object is exist or not.
- Create a **writeToBin ()** method that will write the HashSet object to the binary file named as "**binary**.**bin**".
- Create a **search(…)** method that will take a document id as a parameter. If found it returns the object with that id, otherwise returns null.
- Create a **getIds()** method that returns the id's of the document objects in the HashSet as a String array in a sorted order.
- Create a **displayByName()** method that returns the content of the documents in the HashSet according to the document name by using the DocumentNameComparator.

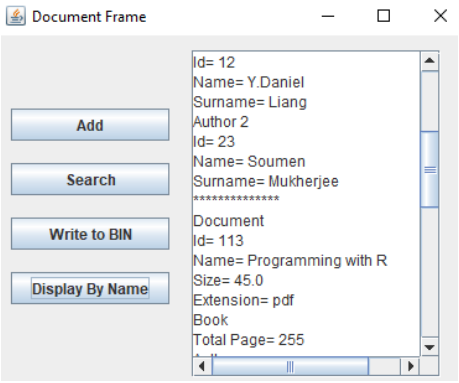
**PART B: Implement your GUI**



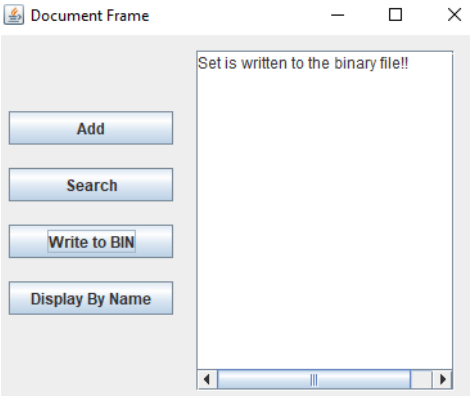

1) When the program starts in the main class;
   a) File will be read by invoking the readDocumentsFromFile() method.
   b) The start-up frame should be created and sets its visibility to true.
   c) Set a message to the text area that the file is read.
      i. There are 4 buttons and 1 text area as shown below.
      ii. Set the title to "Display Frame"



d) When the user clicks on the **"Display by Name"** button display the content of the HashSet by invoking the **displayByName()** method.

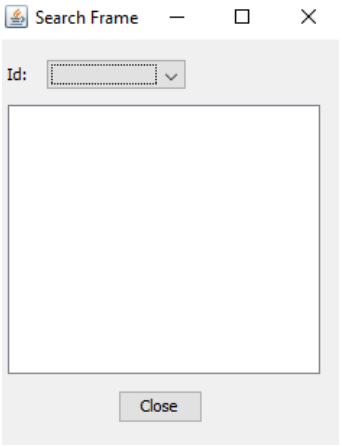



e) When the user clicks on the **"Write to BIN"** button write the content of the HashSet to the "output.bin" binary file by invoking the writeToBin() method. Also display "Hash Set is written to the binary file!!" message on the text area.

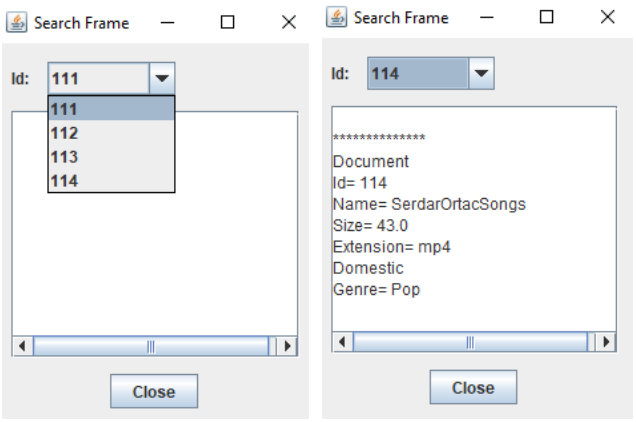



f) When **"Search"** button is clicked open the SearchFrame.

i. If the hash set is empty, give a warning message and do not open a frame
ii. Otherwise; A frame should be created and sets its visibility to true
iii. Set the title to the "Search Frame"
iv. There are 1 buttons, 1 label, 1 combo box and 1 text area.
v. All ids of the document objects in the HashSet will be shown on the combo box as sorted in ascending order.

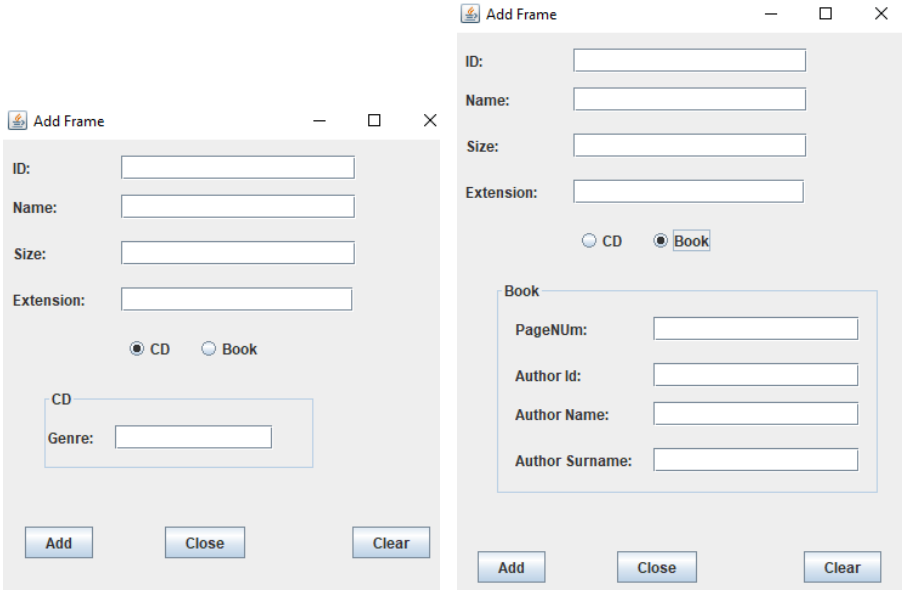



- When an id is selected from the combo box, display the content of the document with that id on the text area by invoking the search(..) method.

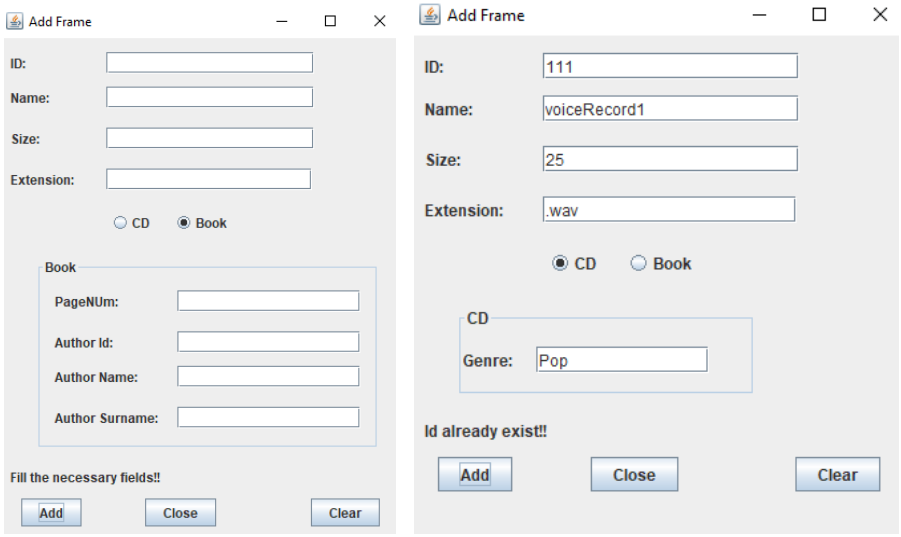


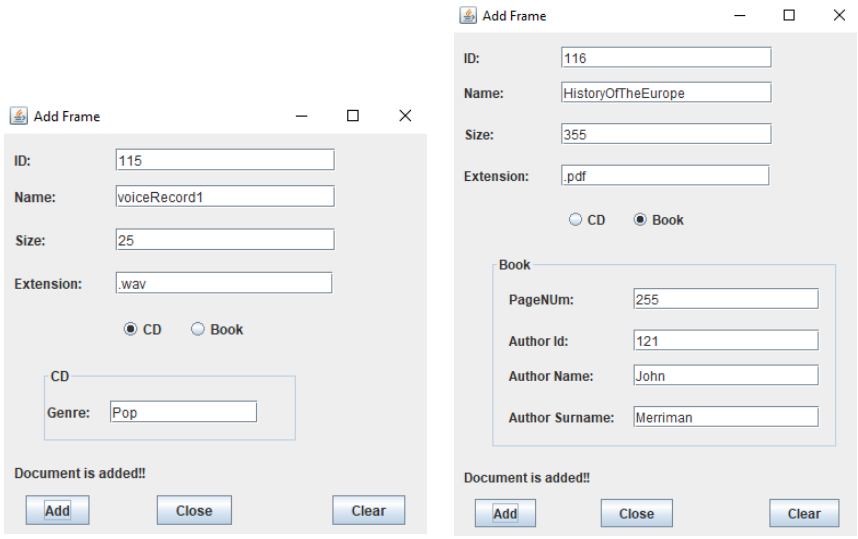

- When “CLOSE” button is clicked dispose the frame.

g) When the user clicks on the **“Add”** button
   i. A frame should be created and sets its visibility to true
   ii. Set the title to the “Add Frame”
   iii. There are 9 text fields, 3 buttons, 2radio buttons, 9 label and 1 message label.




- When the **“Add”** button is clicked on the AddFrame,
  - Get information from the related fields,
  - If the CD radio button is selected; add the document to the HashSet by invoking the addCD(…) method.
  - If the Book radio button is selected; add the document to the HashSet by invoking the addBook(…) method.






- When **“Clear”** button is clicked clear all the text fields and the message label in the frame.
- When **“Close”** button is clicked dispose the frame.