

**OBJECTIVES:** Binary Files, Serializable Objects

**Instructor :** Burcu LİMAN

**Assistants:** Burcu ALPER, Leyla SEZER

### Serialization:

**Serialization involves saving the current state of an object to a stream, and restoring an equivalent object from that stream.** The stream functions as a container for the object. Its contents include a partial representation of the object's internal structure, including variable types, names, and values. The information stored in the container can later be used to construct an equivalent object containing the same data as the original.

**For an object to be serialized, it must be an instance of a class that implements either the Serializable or Externalizable interface.** Both interfaces only permit the saving of data associated with an object's variables. They depend on the class definition being available to the Java Virtual Machine at reconstruction time in order to construct the object.

The Serializable interface relies on the Java runtime default mechanism to save an object's state.

**Writing an object is done via the writeObject () method in the ObjectOutputStream class (or the ObjectOutputStream interface). Writing a primitive value may be done through the appropriate write<datatype> () method.**

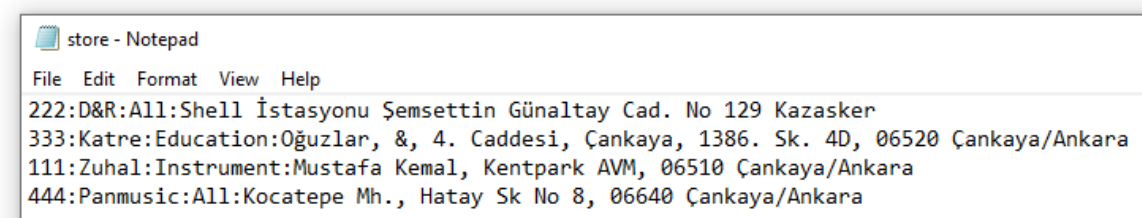
**Reading the serialized object is accomplished using the readObject () method of the ObjectInputStream class, and primitives may be read using the various read<datatype> () methods.**

Write a Java program that reads music store informations from a file named "store.txt" and stores them into the related structures. In part b, program creates a GUI for the related informations and first gets the data from the file, then makes add and display operations.

### **PART A: Implement your classes**

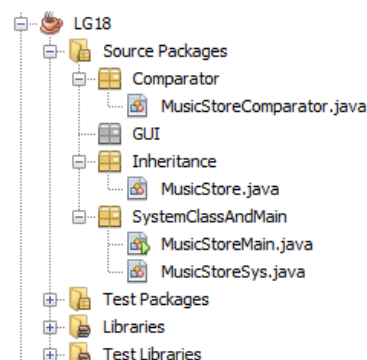
Your program will get input from 1 text file with the following structures;

The **store.txt** file includes information about music store as; **id, store name, sold product type and store address**



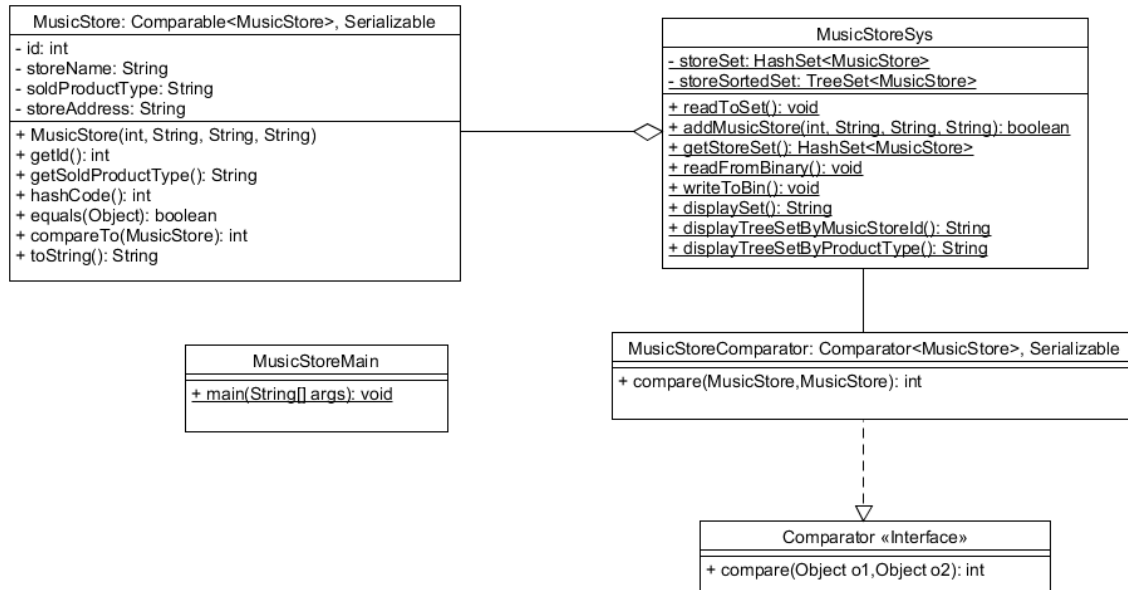
```
File Edit Format View Help
222:D&R:A11:Shell İstasyonu Şemsettin Günaltay Cad. No 129 Kazasker
333:Katre:Education:Oğuzlar, &, 4. Caddesi, Çankaya, 1386. Sk. 4D, 06520 Çankaya/Ankara
111:Zuhal:Instrument:Mustafa Kemal, Kentpark AVM, 06510 Çankaya/Ankara
444:Panmusic:A11:Kocatepe Mh., Hatay Sk No 8, 06640 Çankaya/Ankara
```

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 HAS-A relationship between;
  - MusicStore – MusicStoreSys



#### Information of the MusicStore class structure:

- Implement the given data members and methods according to the uml-class diagram.
- Implement hashCode() and equals(..) methods for the id.
- Implement **compareTo(..)** method that compares the id.

#### Information of the MusicStoreComparator class structure:

- Implement **compare(..)** method that compares productTypes of the objects.  
**Hint:** There can be more than one object with the same product type.

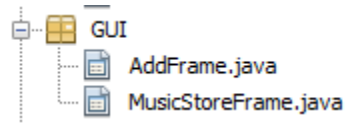
#### Information of the MusicStoreSys class structure:

- Create the data members;
  - A HashSet which stores **MusicStore** objects.
  - A TreeSet which will store **sorted** MusicStore objects.
- Implement a **readToSet()** method, in which reads information from the text file and stores them in the HashSet.
- Implement a **addMusicStore(..)** method that takes the Music Store information as parameters. Then it creates and adds the MusicStore object to the HashSet.
- Implement a **getStoreSet()** method that returns the HashSet.
- Implement a **readFromBinary()** method that reads the binary file in to a TreeSet by sorting it in ascending order according to the productType.. While reading your objects, do not forget to cast them into HashSet<MusicStore>.
- Implement a **writeToBin()** method that writes the content of the HashSet to the binary file named as “output.bin”.
- Implement a **displaySet()** method that displays the content of the HashSet.
- Implement a **displayTreeSetByMusicStoreId()** method that returns the content of the HashSet in ascending order according to the storeId. Store the sorted HashSet in a TreeSet.
- Implement a **displayTreeSetByProductType ()** method that sorts the TreeSet according to the product type.

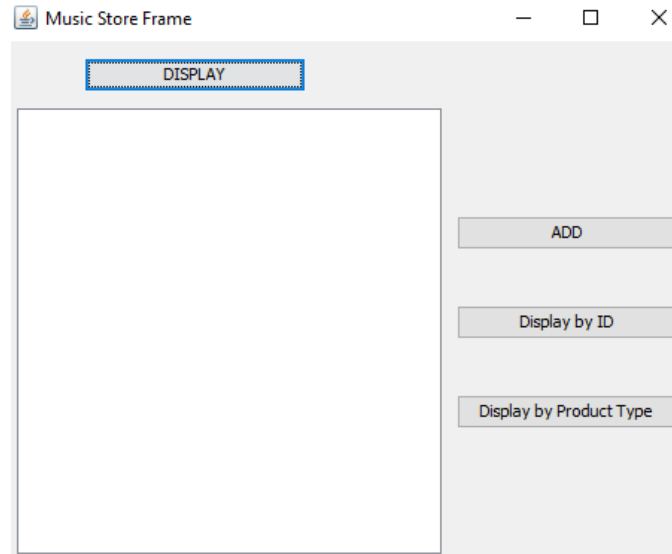
#### MusicStoreMain:

- Read the file by invoking the **readToSet()** method.
- Set the visibility of the MusicStoreFrame to true.

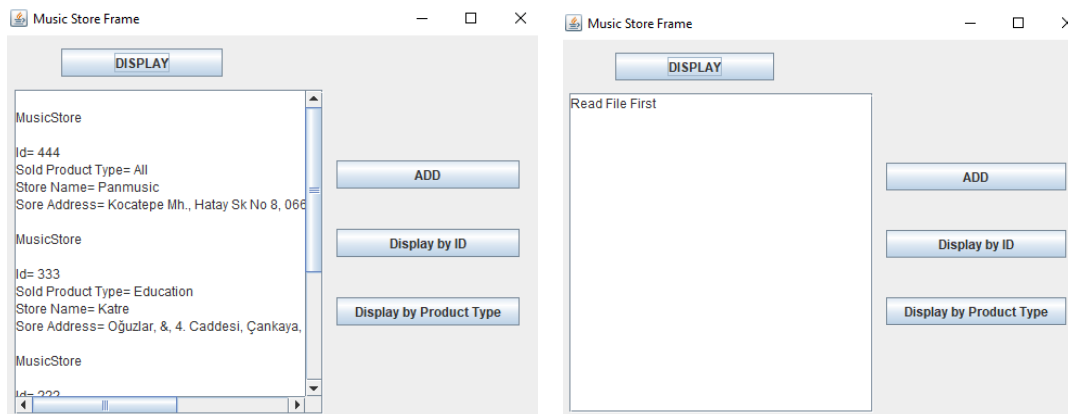
## PART B: Implement your GUI



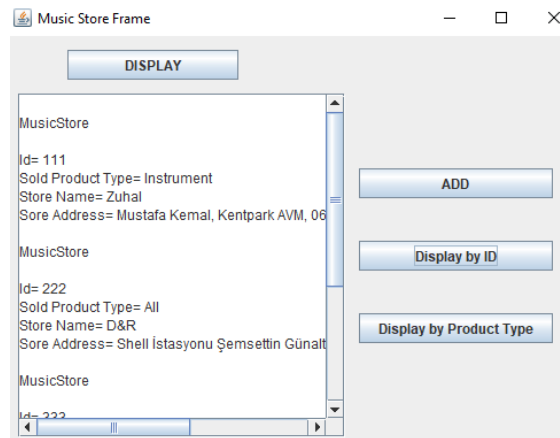
- 1) When program starts in main class;
  - a. File will be read by invoking the **readFromFile()** method.
  - b. The start-up frame should be created and sets its visibility to true;
    - i. There are 4 buttons, 1 text area as shown below.
    - ii. Set the title to the **“Music Store Frame”**.



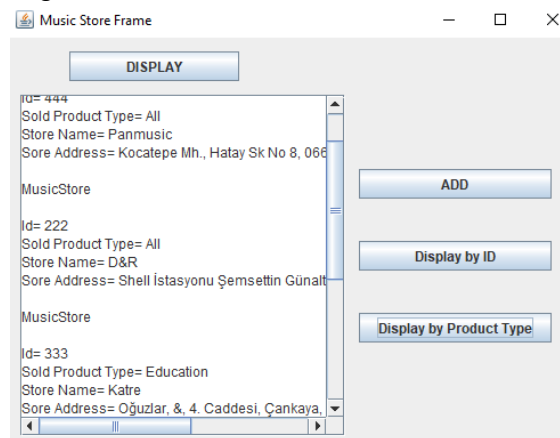
- c. When user clicks on the **“DISPLAY”** button, content of the HashSet shown in the text area by invoking the displaySet() method. If the related structure is empty give a warning message.



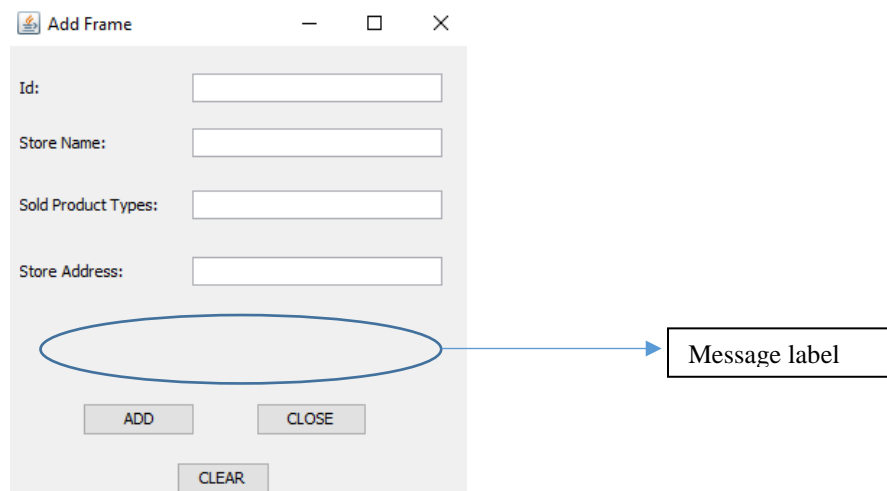
- d. When **“Display by ID”** button is clicked display the content that is sorted according to their ids, by invoking the `displayTreeSetByMusicStoreId()` method. If the Music Store HashSet is empty give a warning message.



- e. When **“Display by Product Type”** button is clicked first read the binary file by invoking `readFromBinary()` method then display the content of the TreeSet by invoking the `displayTreeSetByProductType ()` method. If the Music Store HashSet is empty give a warning message.



- f. When the user clicks on the **“ADD”** button;
- A frame should be created and sets its visibility to true;
  - Set the title to the **“Add Frame”**
  - There are 2 buttons, 4 text fields, 5 labels as shown below.



- iv. When **"ADD"** button is clicked;
  - Add the Music Store to the storeSet by invoking the **addToStoreSet(..)** method. Also display a message on the label.
  - Write the content of the HashSet to a binary file by invoking the **writeToBin()** method.
  - Give a message on label for addition and writing operations.
- v. When **"CLEAR"** button is clicked clear all the text fields and the label for the message.
- vi. When **"CLOSE"** button is clicked dispose the frame.

The 'Add Frame' window contains four text input fields: 'Id:' with value '444', 'Store Name:' with value 'Maya Müzik Evi', 'Sold Product Types:' with value 'Education', and 'Store Address:' with value 'Tunaha. 187, Sk, 067'. Below the fields, a message label displays: 'A Music store with the same id already exists in the system'. At the bottom are three buttons: 'ADD', 'CLOSE', and 'CLEAR'.

The 'Add Frame' window is shown with the 'Id:' field cleared. The message label now says: 'Fill the necessary fields'. The 'Store Name', 'Sold Product Types', and 'Store Address' fields remain filled with their previous values. The 'ADD', 'CLOSE', and 'CLEAR' buttons are at the bottom.

The 'Add Frame' window shows the 'Id:' field with value '342'. The message label displays: 'Music store is added.Tree Set is written to the binary File'. The 'Store Name' is 'Maya Müzik Evi', 'Sold Product Types' is 'Education', and 'Store Address' is 'Tunahan, 186, Sk, 067'. The 'ADD', 'CLOSE', and 'CLEAR' buttons are at the bottom.

The 'Music Store Frame' window has a 'DISPLAY' button at the top. Below it is a scrollable list box containing two entries:
 

- Id= 333  
Sold Product Type= Education  
Store Name= Katre  
Sore Address= Oğuzlar, &, 4. Caddesi,
- Id= 342  
Sold Product Type= Education  
Store Name= Maya Müzik Evi  
Sore Address= Tunahan, 187. Sk., 067

 To the right of the list box are three buttons: 'ADD', 'Display by ID', and 'Display by Product Type'. A blue circle highlights the second entry in the list, and a blue arrow points from this circle to the 'ADD' button in the 'Add Frame' window shown in the previous screenshot.