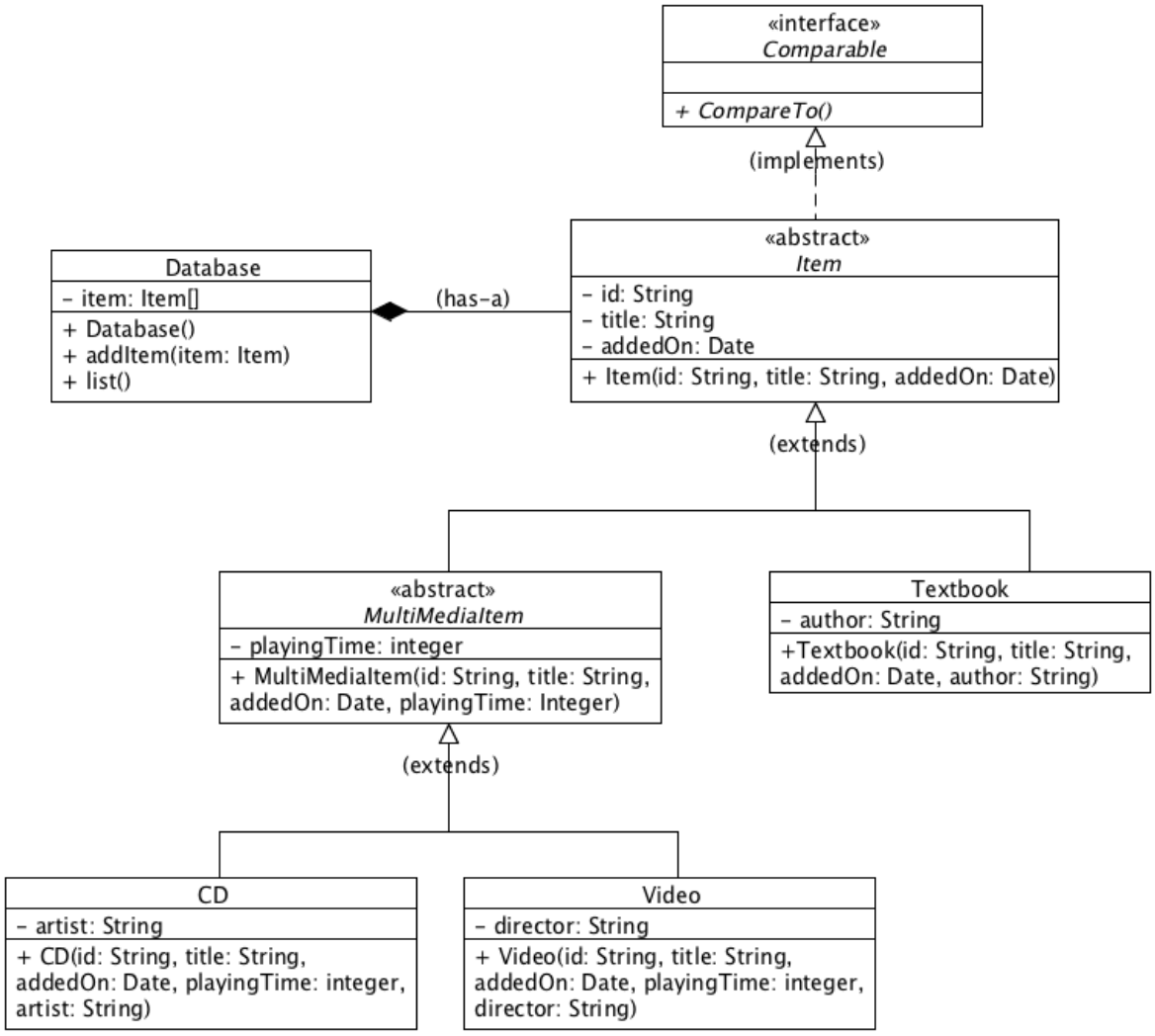
Comparator application

Consider the management and maintenance of a "library database" in a certain school. It holds the information of material resources in use for students in that school. This information is recorded based on the *object-oriented design*. The skeleton of the database structure is shown using the *Unified Modeling Language* (UML) diagramming method illustrated below. UML's standard notion is still being finalized for global use, and so annotations, such as (*has-a*) to express object composition relationship and (implements) to express the use of an interface, are added to the diagram:



The above structure shows instance variables and constructors only. Plus sign (+) indicates public visibility whereas minus sign (-) sets out private scope. The Database may use ArrayList instead of array to hold the collection of Item's references. Add an appropriate set of overloading/overriding methods to support the object hierarchy assumed in this context. Implement the hierarchy, populate a Database object with at least two items per non-abstract class instances, and show the "unsorted" and "sorted" list of the set of those instances. The minimal requirement of this assignment is to realize the database sort based on the lexicographical order of ids as well as any combination of values of instance variables. For instance, database records are shown based on the lexicographical order of "title," followed by "addedOn," and followed by "director."

Realize your implementation that satisfies the following minimal requirements:

- Properly exhibits right logic, *i.e.*, readable and compilable coding
- Properly realizes object hierarchy with added methods
- Properly realizes object listing
- Properly realizes sorting of database items based on their id values
- Properly realizes sorting of database with any fields designated sorting order of items in database

See Comparator example here to realize all the requirements defined above. ComparatorChain comes from here. There seems to be no need of having the two substring() statements in the posted sample program. Explore the code to understand what all the usage of `ComparatorChain` means.

As a hint of your implementation, the following coding of a main function should list the output of what this assignment mandates:

```
    public static void main(String args[]) {
     ComparatorChain chain = new ComparatorChain();
     Database library = new Database();
     Calendar cal = Calendar.getInstance();

     // adding database entries
     cal.set(1890, Calendar.AUGUST, 10);
     Date date = (Date) cal.getTime();
     library.addItem(new Textbook("TB15", "TextX", date, "John Doe"));

     cal.set(1954, Calendar.JANUARY, 18);
     date = (Date) cal.getTime();
     library.addItem(new Video("V09", "VideoB", date, 70000, "J. Smith"));

     cal.set(2000, Calendar.FEBRUARY, 29);
     date = (Date) cal.getTime();
     library.addItem(new Textbook("TB01", "TextY", date, "John Doe"));

     cal.set(2000, Calendar.FEBRUARY, 29);
     date = (Date) cal.getTime();
     library.addItem(new CD("CD07", "CD1", date, 1000, "B.D."));

     cal.set(1990, Calendar.APRIL, 30);
     date = (Date) cal.getTime();
     library.addItem(new CD("CD10", "CD1", date, 800, "X.Y."));

     cal.set(2000, Calendar.FEBRUARY, 29);
     date = (Date) cal.getTime();
     library.addItem(new CD("CD05", "CD1", date, 1000, "B.C."));

     cal.set(1890, Calendar.JULY, 2);
     date = (Date) cal.getTime();
     library.addItem(new Video("V12", "VideoA", date, 7000, "Joe Smith"));

     // print unsorted database
     System.out.println("----- DATABASE BEFORE SORTING: -----\n");
     library.list();
     // sort and print sorted database (by id)
     Collections.sort(library.item);
     System.out.println("----- DATABASE AFTER SORTING BY ID (default): -----
\n");
     library.list();
     // sort by other fields
```

```
     System.out.println("----- DATABASE AFTER SORTING BY OTHER FIELDS: -----
");
     System.out.println("------------ (title, addedOn, director) -----------
\n");
     chain.addComparator(new sortByTitle());
     chain.addComparator(new sortByAddedOn());
     chain.addComparator(new sortByDirector());
     Collections.sort(library.item, chain);
     library.list();
   }
```

Note finally that you only need ComparatorChain.java to realize this assignment. You don't need to structure your folders to force importing the package like`org.apache...` as in the posted example. Just place it into your source folder without its `import` statement!