Java Coursework Report

Outline: ***1 - Summary of Features, 2 - Design and Programming Process, 3 - UML Diagram***

1)Summary of Features

My list-based web application is a TV Show Catalog. The user gets to create new categories, add TV shows to categories they see fit, edit and delete those TV shows and categories and view every category and their contents. They can also use the search option where they can find any TV Show or category that matches their search criteria.

The TV Show Categories have a name and multiple items. Each item has a name, a description, a link to a website where the show can be watched, an image about the show, and a link to a similar category to the TV show where the contents of that category can be viewed. In the search page, once the search string has been submitted by the user, the web page displays every item that matches the string with all the items contents(e.g. image, url, description). However the categories are displayed as clickable links, when the user clicks them, a page with all the content of that category is displayed.

The data is stored in a JSON file. Data is written to the JSON file every time a new TV show or category has been added, or the current one's has been edited. Each time the program is run, the data from the JSON file is read, so once the user enters data into the web-based TV Show Catalog, their data is not lost.

In this web application the Model-View-Controller method has been strictly followed and every feature mentioned in the coursework has been implemented.

1. Categories and TV shows are identifiable by name, a category can store multiple TV Shows and the program can store multiple categories, TV shows include text, URL, image, links to other categories
2. Creating, editing, deleting and viewing TV Shows
3. Adding, editing, deleting and viewing categories
4. A TV show can contain a link to a category
5. A search displays all TV shows and categories that match the search criteria
6. TV shows and categories are immediately saved to a JSON file when there is a change and read from that file when the web application is run

2)Design and Programming Process

As I had no prior knowledge or experience about HTML, web development, Servlets and JSPs, it was a very rough start for me. I have spent many hours learning about all the aforementioned, and even then finding a starting point was difficult.

I started the project by making the Item and ItemList classes. The ItemList class has a private instance variable of List type called "items". This *items* list is what stores all instances of the Item class within the ItemList class. Instances of the Item class can only be added to the *items* list after an ItemList instance has been created. Once an ItemList instance has been created, it is immediately added to the private instance variable of type ArrayList in MotherList class. The ArrayList in the MotherList class stores all the ItemList instances created in the Model class. Basically, the ArrayList in MotherList class stores every ItemList and Item that is created in the web application. I have implemented the MotherList class after learning about JSON. I thought if I stored all the content I have in a single list, it would be very convenient for me to read from and write to a JSON file.

Since the ArrayList class in MotherList class contains all instances of ItemList and Item, the ArrayList can be used to write to a JSON file. Everytime the user creates an ItemList, Item or edits an ItemList, Item; the ArrayList in MotherList class is used to write to a JSON file. The ModelFactory class creates a single static Model instance therefore it's always in scope as long ModelFactory is. Hence, I have added the method to read the JSON file which initializes the ArrayList in the MotherList class, loading the previous ItemList and Item instances the user has created in the web application.

When the web application is run the first page that comes up is the index.html. In the index file, there exist clickable links for viewing all categories and TV shows (all ItemList's and Item's), adding a TV show (Item), searching for a category or TV show etc. Some links are direct links to another page without going to a servlet first. If the link is a direct link this means that the page doesn't need a servlet to grab information via using the model class. If the page requires to get information from a servlet, the link goes to a servlet first and stores the acquired information in the request object of the Servlet and then forwards or redirects the user to the page that they requested. I think this is a good implementation choice because I have avoided writing extra code for Servlet's that wouldn't have added anything to the program.

For the edit and delete functionality of my program, I have chosen to include them both in a single page for both a category (ItemList) and TV show (Item). I think creating a separate page for deleting is unnecessary because it's a single click of a button and it provides similar functionality to editing. How my edit/delete Servlet's work is a bit special so I would like to explain it. Firstly the editServlet decides the type of the request by checking the link that the user clicks and displays the category or TV show edit/delete page accordingly. Then the page is displayed and the user edits or deletes the category and TV show. The page delivers the information the user submits to their servlet (editItem or editList servlet). The servlet checks whether the delete button has been clicked or not. If clicked, deletes the item or list, if not edits the item or list. I think this is complicated to understand and not the best implementation however my goal with this one was to use less Servlet's and JSP's because I thought I was being excessive and I definitely have achieved that goal.

I think in a list application, it is very intuitive to use an Item class to represent elements and an ItemList class to represent the list that the elements are being stored in. I have separated the image reading functionality in a new class because I thought I could encapsulate it with a simple read method in my Model class. I have also used a MotherList class which made reading and writing to JSON very convenient and was overall a smart design decision. I think I have shown good OOP design practice because the classes I used were quite intuitive and cohesive. I have managed to easily put them together in my Model class.

My application follows a strict Model View Controller (MVC) design and this is the part I'm most proud of. In the beginning I was using all my classes in the JSP's files, failing to see that this was against the MVC design. Moreover, I had a lot of methods in Item and ItemList class because I thought I was making my web application more abstract, however the code was just as complex in my Model class. I got rid of the unnecessary encapsulations and fixed every single JSP file. Many methods returned Item or ItemList class objects therefore I had to fix my methods as well since it would be against MVC design to get Item or ItemList objects in my JSP files. My criticism for myself in this coursework would be the confusing naming of attributes and variables in Servlet's and JSP's. Other than that, I believe that my application follows good OOP design and very readable code. Overall, I believe that I have produced a high quality work that is very functional and satisfies every criteria of the coursework.

UML Class Diagram for my Web-Application

***I had to do the UML Diagram by hand as the arrows got too tangled in the tools (draw.io, visual-paradigm) I was using and it was impossible to understand the diagram.*