Project Write Up:

By Danilo Da Silva and Ashley Paddock

Original Project Proposal:-----------------------------------------------------

We will be creating a Java application for the management of lists.   
Users should have the ability to

First Iteration: Basic list management

* Create a list
* Add and Remove items from list

Second Iteration: enhance list features

* Set list types (grocery, to - do, shopping, goals, etc.)
* Categorize items within lists
* Prioritize items within list

Third Iteration: (Optional enhancements depending on time)

* Create private lists (requires username & password to view)
* Create user profiles
  + So multiple users can edit single list and each item belongs to specific user
* Record history/build dictionary of items users have used in the past
  + So users may quickly build new lists

Awesome features that we won’t have time or knowledge to do:

* Add notifications so users would have the option to be notified when a list is updated
* Add message feature so users could communicate with each other
* Add commenting feature so users could make comments on lists and/or items in list

Basic Idea:   
A list management application that can be updated and shared among people would be really useful.   
Families from a distance could view and add to a list.

Example Use Cases:

1. One large list for all family members to add what they would like for Christmas gifts/ Christmas dinner/ideas for things to do on vacation/etc. for when they visit on Holidays

2. List for parents to allow children to add items they need for school/ want for dinner/ want for Christmas/Birthday, etc.

Original Basic Use Cases: ------------------------------------------------------------------------------

As a user I would like to have an app that I can keep track of different types of lists.

* Create a list
  + As a user I would like to have the ability to create a list.
  + The user should have the ability to create a new list
    - The user can select ‘Add List’
    - The user is prompted to give the list a name.
* View List
  + The user should have the ability to view an existing list.
    - The user can select ‘View List’
    - The user is prompted to give the list name
    - The list with the given name will be displayed for the user to review.
* Add and Remove items from list
  + As a user I would like to have the ability to add and remove items from my lists.
    - The user will select ‘add to list’/ ‘remove from list’
    - The user will be prompted with 1. The List Name and 2. The item name
    - The item will be added to the list with the given name / removed for the list with the given name.
    - The user will be prompted to save the add/remove
    - The user will have the option to ‘Confirm’ / ‘Cancel’
      * If user confirms, the list is updated and displayed for user to view.
      * If user cancels, the changes are reverted and the list is displayed for user to view.
* Set list types (grocery, to - do, shopping, goals, etc.)
  + As a user I would like to have the ability to give a list a TYPE so that I can save multiple lists and organize them by type.
    - Users will have the ability to add a TYPE to each list.
    - All TYPES a user has previously used for a list will be stored in a list.
      * The list of TYPES will be used for the user to search by
      * The list of TYPES will be used as a dictionary for new list TYPES in the future.
    - Users will select ‘Update Type’
    - User will be prompted for a list name
    - User will be told to select from list of previously used TYPEs or ‘Add New Type’
      * If user selects from list of TYPEs, the list with the list name provided is updated to have the select TYPE.
      * If user selects to add a new TYPE, the user will be prompted to insert a new TYPE name.
      * The new TYPE name is added to the user’s TYPE dictionary and the list with the given list name is updated to have the newly created TYPE.

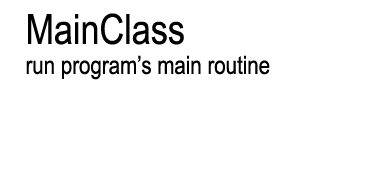
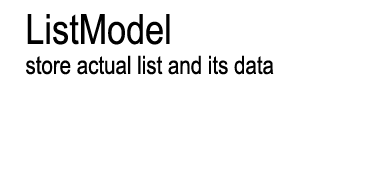
Original Design Specifications: ----------------------------------------------------------------

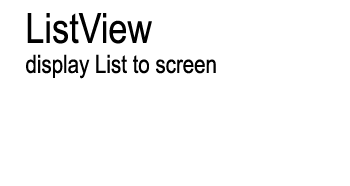
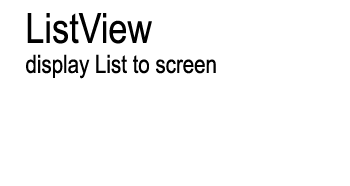
**Updated Application Requirements:**

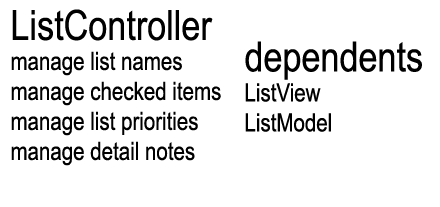
Any tablet or computer device with a web browser that has Java runtime plug in enabled.

The application runs as a web based java applet.

**CRC Cards:**

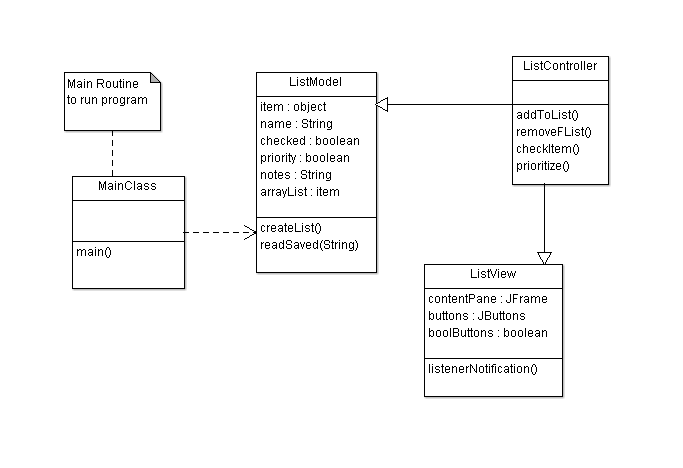
 

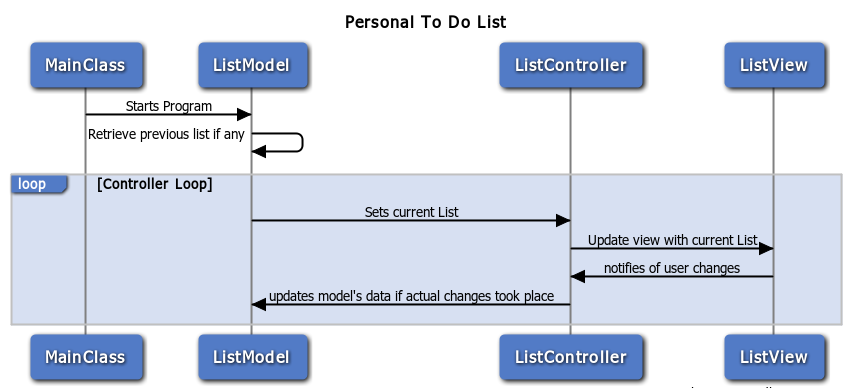


**UML Diagrams:**

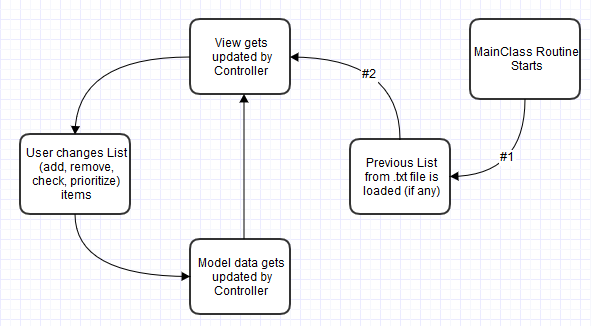
Class Diagrams:



Sequence Diagram:



State Diagram:



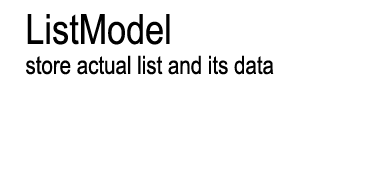
Final Design Specifications: ----------------------------------------------------------------------

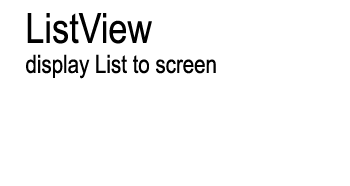
**Updated Application Requirements:**

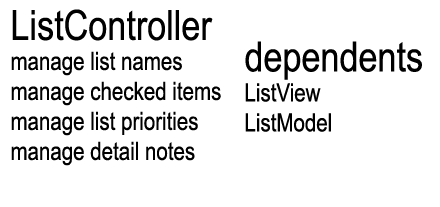
Any tablet or computer device with a web browser that has Java runtime plug in enabled.

The application runs as a web based java applet.

**CRC Cards:**

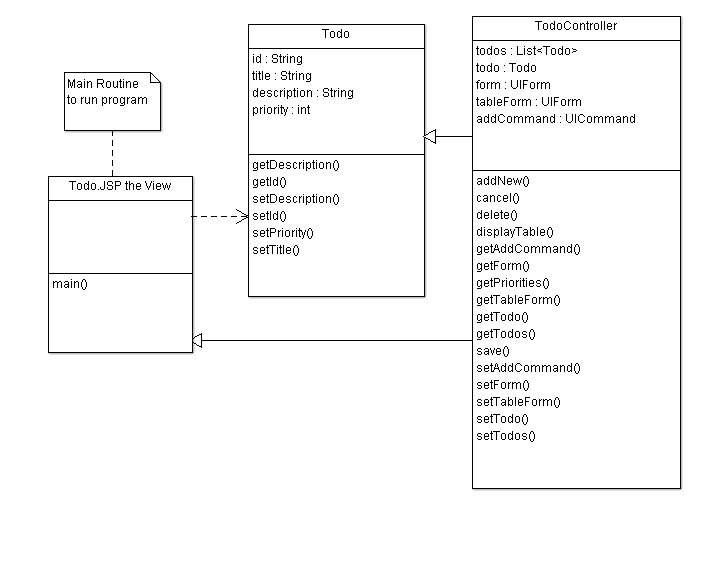




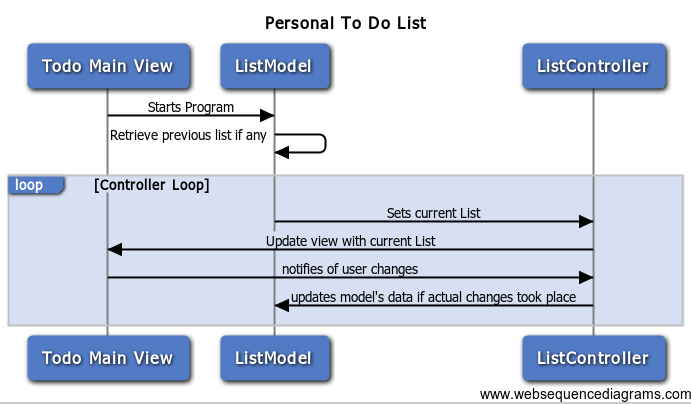


**UML Diagrams:**

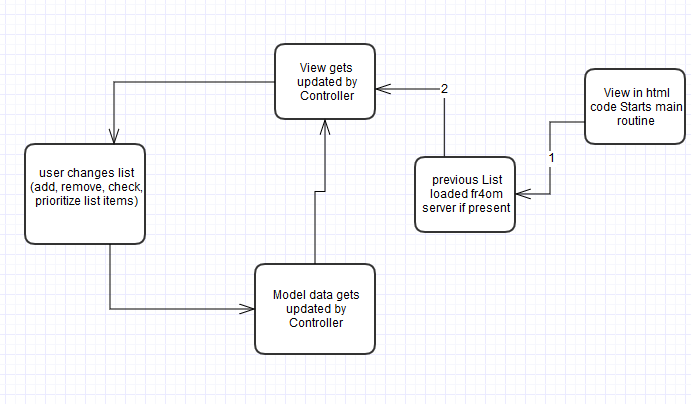
Class Diagrams:



Sequence Diagram:



State Diagram:



Used Design Patterns:

* Observer Pattern
  + MVC architecture for overall program structure
    - View in JSP File, Model for Data Storage and Controller for Methods in JAVA files, each in appropriate packages
* Strategy Pattern
  + view using layout tools for dynamic view
* Composite Pattern
  + Using CSS Styling for JSP file with HTML code
* Decorator Pattern
  + Change preset methods and classes for our program

**Conclusion:**

The inherit difficulty of this project with JAVA being the language to be used was heightened by the use of MVC with most common applications in the industry. Because most applications in the industry with JAVA do not make common use of MVC model. Our original choice of applications was to make the To Do list as an android app. Yet Android does not make use of MVC.

Our project in android had to be halted half way through because when it came to design the model and view, XML would have to be used. Since both of us lacked experience with both java and android, much less their use with XML, we decided to give up on the android app.

The time lost hurt us in being able to finish the project with the persistency functionality. We tried multiple different models of java web apps. Until we decided to use the Faces model from JSF. We implemented JSF model and libraries, along with a JSP file with HTML code as our view.

It still proved challenging due to the way the model relates to the controller and model JAVA files. The web applet runs on Tomcat 7.0 server. Which after many failed attempts, library errors, and file set up location errors, we were able to run the JSP file on the server (using localhost) and get view file in communication with the model.

The persistency of the to do list data is true while the server is running. We weren’t able to achieve a true persistency functionality due to time and language constraints we had to face when learning JSF and JAVA web applets.

We cannot deny we learned with this assignment. Yet more could be taken into our advantage if more environment ranges were given to us, our if MVC model was the key one utilized.