William Borum
CS220
Prof Arias
02/17/2017

Project 2: Semester Project – Movie Theatre Ticketer

Of the three typical project groupings, I immediately gravitated towards a database project. I find it more enjoyable and practical to work with real-world data. Also, getting experience using a database, I feel, is most helpful in preparation for any type of applied math or computer science job. To make this project more interesting, I thought I would use a database that had to do with movies. In line with this idea, I propose to make a movie theatre ticketing program. I will lay out the goals, aims, and aspects of a project that deals with movie theatres.

Goals/Aims

1. Choice
   a. I want to provide the user with many options to tailor the experience uniquely to that specific person.
   b. I want to mimic the way people decide to go to the movies in the first place. When moviegoers do not like one movie, they will move onto the next one.
   c. Moviegoers prefer to go to the movies at different times, from midnight releases to midday matinees.

2. Usability

   a. I want to make a project in java that will walk the user through the process of figuring out the movie he/she wants to see, to picking a time, to picking the number of seats and finally ordering the tickets.

   b. I want to have a logical and efficient work-flow/user interface. The user should feel as though he/she is being walked through the process of ordering tickets, and not forced to make the program work.

3. Relevance

   a. Most of all, I want this project to resemble the real-deal, in that one would expect to go to the movies the same day and not need to learn anything new.

   b. The program should therefore use real-life ticket prices, times and movies.

   c. The project should make a user feel like they are actually buying a ticket.

Aspects

1. Provide the user the option to pick from existing movies close to them.
2. Provide the user the option to pick from existing times/showings at a theatre near them.
3. Provide the user with available tickets to the movie of their choice.
4. Give the user a rundown of what they ordered/what movie they will see/where they are going.

Above is what I hoped to accomplish with this project, and how I envisioned the project would unfold in terms of expectations and what I knew I could accomplish with java.

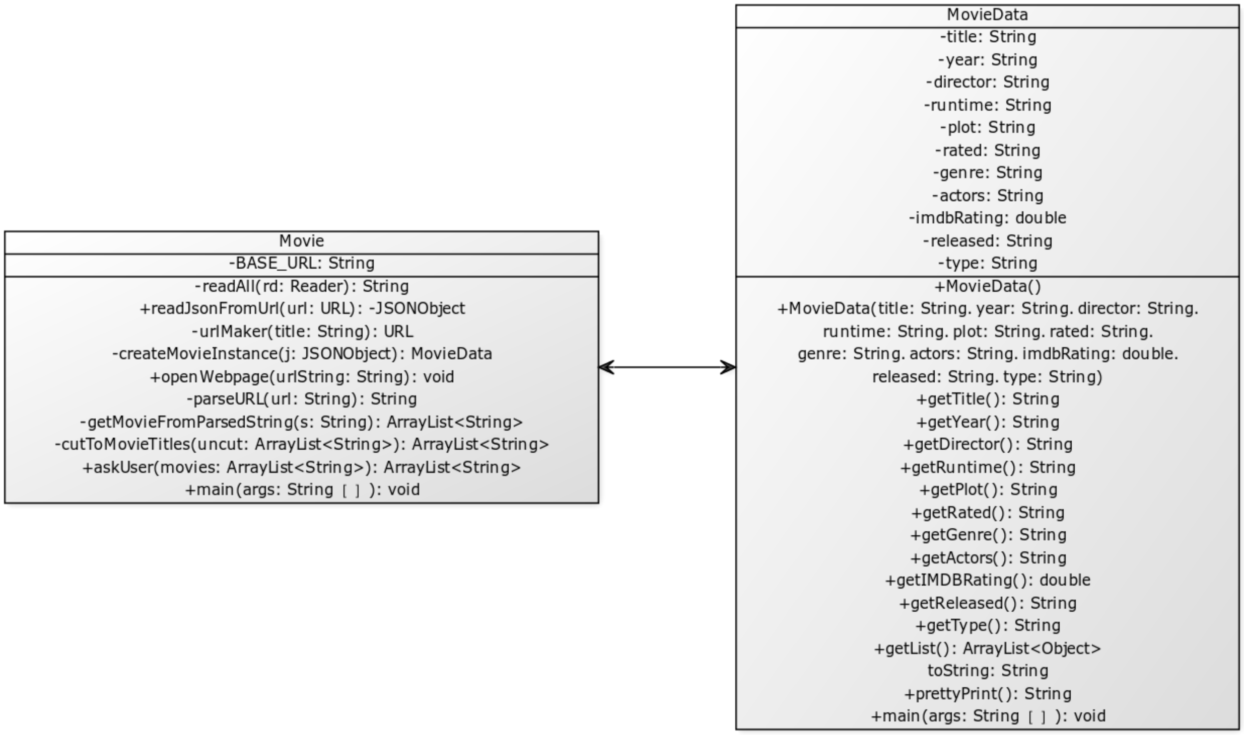
Next is what I had accomplished at the milestone. I had a basic working prototype for how the user would choose a movie and how the movie's information

Abstract

The project that I settled on has to do with movie theatres and the interaction with the moviegoer and the information necessary to get them to buy a ticket. The java programs I created are meant to reenact that experience and spur users to go see movies for themselves.

Detailed System Description

The user starts with the Movie.java file and uses the main method to interact with the program. The method invokes another called askUser() that prompts the user to enter the number of the movie that the user wants to get more information about. The list of movies is taken from fandango.com, by scraping the html code for movies near a certain zip code. The user is then presented with key facts about the movie and its plot/synopsis. This is done by grabbing JSON data from omdbapi.com using the title as a search parameter. The information is parsed and put into the MovieData.java collection so that multiple movies can be displayed should the user need to search multiple movies. Below is the UML Diagram for the Movie.java and MoviData.java.


Movie
-BASE_URL: String
-readAll(rd: Reader): String
+readJsonFromUrl(url: URL): -JSONObject
-urlMaker(title: String): URL
-createMovieInstance(j: JSONObject): MovieData
+openWebpage(urlString: String): void
-parseURL(url: String): String
-getMovieFromParsedString(s: String): ArrayList<String>
-cutToMovieTitles(uncut: ArrayList<String>): ArrayList<String>
+askUser(movies: ArrayList<String>): ArrayList<String>
+main(args: String [ ]): void
MovieData
-title: String
-year: String
-director: String
-runtime: String
-plot: String
-rated: String
-genre: String
-actors: String
-imdbRating: double
-released: String
-type: String
+MovieData()
+MovieData(title: String. year: String. director: String.
runtime: String. plot: String. rated: String.
genre: String. actors: String. imdbRating: double.
released: String. type: String)
+getTitle(): String
+getYear(): String
+getDirector(): String
+getRuntime(): String
+getPlot(): String
+getRated(): String
+getGenre(): String
+getActors(): String
+getIMDBRating(): double
+getReleased(): String
+getType(): String
+getList(): ArrayList<Object>
toString: String
+prettyPrint(): String
+main(args: String [ ]): void

In the coming weeks, I have plenty to work on. I need to let the user pick the zip code that corresponds to where they live. I also need to use the openWebpage() method to give the user a way to actually buy a ticket, either on Fandango or on another website. Lastly, I need to handle errors more gracefully; right now, there are 3D versions of movies that will not show information using omdbapi, and some other small errors result from over-specific titles.

Requirements

My program helps users to decide what movie they want to see, and it does so by walking the user through a methodical process. This level of attention is what I hope will get through to people, as nowadays with everything online, people will not take time to decide what movie they want to see. This project will get those who use it to think on what makes a good movie theatre choice, and remind them what makes the movies great.

Literature Survey

Many online services offer similar functionality to my program. Some of the biggest ones are IMDB, Fandango, and Google, which all display this information in beautiful ways through html. Fandango is great in that you can actually buy tickets to the movies. Google is good as well because most people will not need to enter another website, as Google displays times for movies on its own search page. And lastly, IMDB is fantastic in that it has nearly all information on movies, actors, directors, reviews, and anything related to movies you can think of.

User Manual

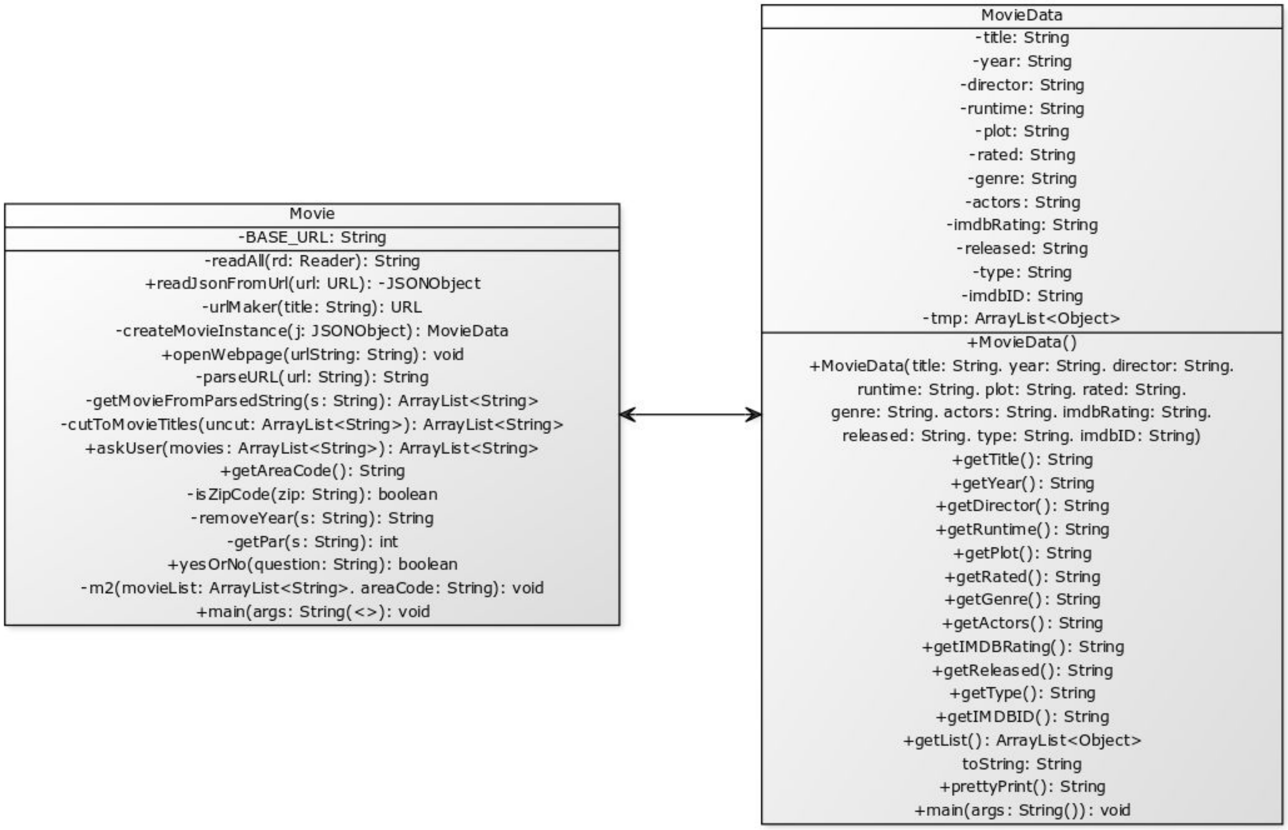
Start by making sure Movie.java and MovieData.java are in the same directory. Compile Movie.java and run it to activate its main method. Look at the list of movies near Poughkeepsie, NY. Find one that looks interesting and type the corresponding number into the terminal. Press enter. A couple of sentences of details will be displayed about the movie you have chosen. Repeat for any movie that you like.

The final touches were made in the weeks leading up to the demo. I had to make the assistant come to life, in a way that would actually help a user to decide what movie they wanted to see, and allow the user to buy tickets.

By following my own advice, I made the changes I set forth in my proposal. Firstly, I added a way to pick your own zip code for a personalized list of movies. I also added a way to get the user to see the IMDB page for the movie he or she is looking at, in hopes that videos and pictures will spur a trip to the movies. I also added the option to go to fandango and purchase a ticket at a local theatre to see the movie that is being offered, and likewise, see the available times for all movies in the area.

But more importantly, I managed to provide a logical progression through the process of looking and buying tickets to a movie. Previously, I had only a static list and an option to get a blurb about the choice. Now I have a functioning assistant to say the least.

Below are the updated UML diagrams.


Movie
-BASE_URL: String
-readAll(rd: Reader): String
+readJsonFromUrl(url: URL): -JSONObject
-urlMaker(title: String): URL
-createMovieInstance(j: JSONObject): MovieData
+openWebpage(urlString: String): void
-parseURL(url: String): String
-getMovieFromParsedString(s: String): ArrayList<String>
-cutToMovieTitles(uncut: ArrayList<String>): ArrayList<String>
+askUser(movies: ArrayList<String>): ArrayList<String>
+getAreaCode(): String
-isZipCode(zip: String): boolean
-removeYear(s: String): String
-getPar(s: String): int
+yesOrNo(question: String): boolean
-m2(movieList: ArrayList<String>. areaCode: String): void
+main(args: String(<>): void
MovieData
-title: String
-year: String
-director: String
-runtime: String
-plot: String
-rated: String
-genre: String
-actors: String
-imdbRating: String
-released: String
-type: String
-imdbID: String
-tmp: ArrayList<Object>
+MovieData()
+MovieData(title: String. year: String. director: String.
runtime: String. plot: String. rated: String.
genre: String. actors: String. imdbRating: String.
released: String. type: String. imdbID: String)
+getTitle(): String
+getYear(): String
+getDirector(): String
+getRuntime(): String
+getPlot(): String
+getRated(): String
+getGenre(): String
+getActors(): String
+getIMDBRating(): String
+getReleased(): String
+getType(): String
+getIMDBID(): String
+getList(): ArrayList<Object>
toString: String
+prettyPrint(): String
+main(args: String()): void

As can be seen, I added quite a few methods, mostly to help loop the listing of movies and open the IMDB and Fandango pages in your local browser.

New User Guide

Start by making sure Movie.java and MovieData.java are in the same directory. Also ensure that the json dependency is met, as Eclipse does not have org.json in its default library. To do this in Eclipse, add the provided jar to your build path by right clicking the Project > Build Path > Configure build path> Select Libraries tab > Click Add External Libraries > Select the Jar file. (http://stackoverflow.com/questions/8997598/importing-json-into-an-eclipse-project) Compile Movie.java and run it to activate its main method. The assistant will ask for a zip code, enter one that you live near, or one that you want to see movies for. The program will test if the input resembles a 5 digit zip code or not, and ask again if the input does not match. A list will appear with numbers next to movies that are being shown in or around the specific area code. This is done by parsing the html of the Fandango page of the area code the user put in at the start. Look at the list of movies near the specified area code. Find one that looks interesting and type the corresponding number into the terminal. Press enter. A couple of sentences of details will be displayed about the movie you have chosen. The assistant will ask if you want to see the IMDB page for that selection; answer with y or n. If yes, the program will open the specified IMDB page in the default browser in a new tab. From here, you can find all the media, information and reviews of the specified movie. When satisfied, return to the terminal, where the assistant will ask if you would like to buy tickets to this specific movie; answer with y or n. If yes, the program will open the Fandango page for the area code, which will show all available times and available tickets, prices, and seating if available. Lastly, the assistant will ask if you want to do the same with another movie, in which case, you can repeat the above for any movie that you like.

Conclusion

It was very satisfying to develop something that I am interested in and to utilize the tools we learned throughout the semester. The ability to use an api like omdbapi was very liberating. I felt that it was one of the first projects that I have done that could resemble a finished product for everyday use. The fact that this assistant will work the same 6 days from now as it will 6 months from now is stunning in its own right. I have to say that a great deal of effort was put into making this project work as intended, especially in parsing the html of Fandango. I am happy in what I was able to accomplish, and I hope this will help others build on the fun that comes from movie-going.