**JOURNAL**

This course covered a wide variety of topics. There was a lot to go over and a lot to discuss. I admit, I had a very difficult time in this class, and a lot of it was due to poor time management, coupled with unfortunate timing from my employer. I got behind real early and real fast and I didn’t start playing the catch-up game until later on. I guess it was the structure of the course that didn’t mesh well with my schedule and with my learning process. We all learn in different ways, but I found this method to be extremely difficult for me. I was, however, able to do a lot of research and learn many new things about programming in Java.

In the beginning, my first step was to look into all of the topics and try to get a general idea of what they were about. I started with Java collections and tried to gain some additional insight as to what they were used for and what they were. What I found is that a collection is basically a grouping of different items or objects in Java. There are many different kinds of collections, such as lists, trees, sets, and maps.

A list, kind of like an array is a collection of elements, and each element on the list has a particular order within the list and is tied to a unique index that can be used for access, retrieval and reference. In a list, you are allowed duplicates.

A set is different from a list, as it is just simply a collection of things. Its basic function is to have a congregation of elements and then with that collection, you can check to see if a certain element exists or does not exist. This can be useful when doing something like checking a set to see if someone has a membership. There is no particular order, no stored index or anything like that. Each elements is either in the set or it is not in the set.

A map is different in the sense that it contains two different kinds of elements: the keys and the values. The keys are mapped to the values and you can use the keys to look up or modify the values, as well as sort or compare, or whatever you would like to do with a collection.

Collections, therefore have numerous uses in applications, because a lot of the time, there will be some kind of grouping of elements, whether it be items for inventory or a listing of members or customers or if it is a list of things to do. Once you know what kind of groupings that you will have in your program, it is then up to you to decide which collection will be the best for your particular need.

The next topic that I decided to do research on was Hibernate. This technology was a bit difficult for me to wrap my head around. The way I understand it, you are basically linking your program to a database for information retrieval and use within the program. It is free to install alongside Java and has many built-in features, including its own language to implement querying and data retrieval. The need for something like this stems from the difference in how Java object and relational databases store information. They each store information in a different way, Java doing so in objects and databases doing so in a more tabular format, with rows and columns in tables. Object-relational mapping takes the important information form the tables and directs them to the points within the object where they need to be.

Hibernate performs its work in segments called sessions and transactions. A session is an instance of a connection from the server to the database. Once a session is established, things can happen. Each time something happens or a unit of work is done, it is considered a transaction.

To install and configure hibernate, you would need to download and install the hibernate library and then put the .jar file in the appropriate classpath folder, with the configuration files in the correct places.

The next technology that I did research on was HTTP url connections. This was interesting to me, because it was a combination of java programming and using web interfaces to gather information from online. There are a lot of useful applications for this technology feature, especially. as it deals with utilizing online resources. There are different data formats that you can transpose, such as the JSON format. The way this works is that information is collected by usage of a “get” request to the server for the requested page. The server, if successful in finding the requested information will send it back in the form of a “post.”

This has a lot of good applications for programs in web applications, especially in the sense that you can create programs that will consolidate information from many different sources into one easily, accessible location. Therefore, you will be able to get more information in a faster way by automatically requesting what you need and having it readily available. Another application is with web scraping and data mining. Basically, you are able to pull the data you need from a website and then put in it a format where you can better organize and analyze the information to find out more about it.

The next area of study was the unit testing framework, Junit test. Unit testing is a type of testing where you will analyze a particular portion of a program. This way, you can compartmentalize your code and make sure each section works before moving on. That way, you will be able to test and catch things as you move instead of testing the system as a whole and then trying to go back and find where an error is if the code does not work.

The JUnit framework comes with its own API of different tools that can be used to run and re-run different units tests for a particular program. Examples of some of these tools are fixtures, which are used to create a set of objects to use as a baseline for unit testing. Test suite can be used to bundle several different tests together. This is useful when you know you have a common set of tests that are applicable to several different units of a project. You can simply execute a test suite to run all the tests at once. Test runner is the feature that is used to execute the test cases. This is able to run different test suites or cases that you have created. Typically, a test case is run using @ notation, so in your code if you want to run a unit test, after creating the necessary test case information you use @(test case name) before the part you want to execute the test case scenario on.

This is a simple technology to integrate with other technologies because it is simply a testing framework and not matter what kind of application or technology you are using, you are able to test it out to see if it works.

The next technology that I came across and researched / tested as the Java threading. This was an interesting technology, because it has to do with the optimization of the program to make it run more smoothly and efficiently. A thread can be described as the path of execution for a program. Typically, a program will have one thread, or one path. In this case, it will work on all of its actions in a sequential manner and execute them until it is done. However, what you can do in an application is create multiple threads so it can execute many things at once. This way, the program can run faster and can multi-task. You can even prioritize different threads and different tasks based on which ones you want to run first.

I found this technology interesting because I feel like that is the next level of programming. On the basic level you have your functionality, making sure that the program runs and that it does what it is supposed to do. On the next level, you start to increase the quality by optimizing the program to either run faster, be more efficient with space, or to have additional functionality and features that make it nicer or easier to use. Multithreading programs fits in this category because you can optimize the program to run better by adding and managing the threads that it uses.

The next area of research that I got into had to do particularly with application control patterns. There were two similar patterns that I researched into, namely the Model-View-Control pattern and the Application Controller pattern. Both of these are similar ideas with a slightly different way of execution.

The MVC pattern consists of three parts based on its name, the model the view and the control. The idea behind this is to separate the code for the key elements of the GUI in order to separate the different elements and features and create specialized classes to handle different elements. The view component typically contains the frames and the physical elements, the model component will usually contain objects or elements that will be contained within the frames or the view elements. The control component will have methods and variables that will determine what the elements will do and how they will function and interact with each other. This kind of implementation can be wise, because you are able to group similar tasks and ideas with each other and have them run together. That way, the code becomes more logically cohesive and you are able to more easily and readily mange and edit the different components

The application controller method has a similar goal and idea behind the way it works. It uses similar elements such as a view, but the execution is different in that you will create different views based on what you want the application to do. Then, the controller will switch between viewing elements based on what parameters or conditions you meet. The idea behind this is modularity. You can separate different views and elements and work on different implementations and features separately. This will make it easier to write, release, test and develop code incrementally

Next on the list was the client/server socket input output. Honestly, this is something that was a little over my head, but I was able to gather some information and learn a bit about its purpose and execution. The basic idea behind this is to utilize TCP/IP ports to transmit data across a network. This can be very useful and vital in some programs, as it allows the user to communicate data anywhere they need using their network or the internet. The primary function of this is to transmit data needed for the program between a client and a server. This allows you to host the required information on a server and then write a client-side program that can make requests to the server for information. That way, the client-side application will be lighter and does not need to store all of the information all the time; it will only get what information it needs. This leads to a faster, more efficient and secure program as the client will only gather the information that it requests form the server and the other things on the server will be secure.

Input and output are moved using different methods as part of the java.io API. DataInputStream can gather and sent input to the server from the client, and DataOutputStream can transmit data form the server to the client or visa-versa. There are also methods like DataPrintStream that can print data directly to the sever/client when necessary.

Another essential part of programming that is often overlooked is the planning and diagramming side of things. A good program needs to be planned and thought out beforehand in order to truly work into a well-organized program. The Unified Modeling Language or UML, is a standard in how to visually represent or model different aspects of a software program. Having a standard way to visually represent things makes it a lot easier for people to create plans for a software program that can be universally recognized and interpreted. There are many different components and aspects to a program, and therefore there are different kinds of UML diagrams and documents that can be created to give an overview and visual representation of different aspects of the program.

One kind of UML diagram is the Use Case Diagram. This diagram is a model of how a program fulfills a specific requirement. This is important because programs are designed with certain requirements in mind that they need to fulfill and this kind of diagram will help in making sure that the program achieves those requirements. There are different components to this diagram, such as the use cases themselves, the actors or primary components or users that will be playing roles in the system, and the associations between them and how they function.

The next kind of diagram is the class diagram. The class is one of the most important aspects of a program, since they are the primary building blocks. A class diagram is helpful because it is able to lay out all of the classes, their attributes, their methods and how they relate to and interact with other classes. It is important to know the relationship between one class and another because then you can gain a greater understanding of how the program is to work. It is also a good thing to get an overview of each class and what it should contain, because then when the coding needs to happen, you can simply refer to the class diagram and know exactly what you need to implement into each class.

The next kind of diagram is the State Diagram. This diagram is concerned about the different phases that the program or its components go through. In this kind of diagram you can take an element, like the UI or an object and you can detail the different things that it does or the different conditions that it can have affecting it. You can also model the different transitions between these states or conditions and how, when and why they happen. This can give you a good overview of the different possibilities for the program and what different things will look like based on what conditions are being met in the program.

Finally, there is the Sequence Diagram. This diagram is primarily concerned with the different processes that the program will undergo and their sequential order. With this diagram you can get a good idea of the natural flow of the program and which events are triggered and in what order they are triggered. The diagram is primarily concerned with participants and each event or action is documented and annotated.

All in all, there is a lot more that goes into programming than I originally anticipated. There are a lot of different technologies that can be utilized in order to make a great program. There are a few specific things that I have taken away from my time and study in this course.

First of all is the importance of planning. I think a mistake that a lot of programmers make is to just jump right into the coding once they have an idea of what they want. However, there are many benefits to carefully planning our your program and documenting its design. For instance, doing this beforehand will give you a greater insight and idea on what you need to do to implement your ideas, and it will be less likely for you to get stuck or to not know where to go. Also, careful documentation is important because once you design a program for someone, you may not always be there to help with it from there. If the program is carefully organized and documented, it can then be better understood later down the road.

Next, I learned how different topics and technologies can be integrated together. For instance, HTTP connections can be used to gather information in a JSON format. IT can then use client/server socket connections to store analyzed data on a server database using Hibernate. Programs can be improved and optimized by applying some of these technologies. Collections can be used to store many elements instead of listing them and declaring them individually. You can also improve the performance of a program by doing so , and by implementing ideas such as multithreading to run many tasks at once in a program. You can combine multithreading with HTTP connections to establish multiple connections at the same time and retrieve data form online sources more quickly. There is just so much that you can do once you have a clear understanding of these technologies.

My final thoughts are on the importance of trying things out. I found throughout my research that I became much more familiar with the topics that I used examples for and that I found and wrote and messed with code. Research and study is good, but application will help you to understand a topic much better in many cases. This can apply to programming to to anything in life, really. In order to truly learn, we must apply our knowledge. Wisdom, I believe is the application of this knowledge.

In closing, there are so many different technologies out there and new methods and new practices and processes are being developed and improved upon and perfected every day. It is up to us to be diligent in keeping up with these practices and to apply them into our lives. If we are diligent in our study and we can practice what we learn and then try to apply it to other areas of our knowledge and expertise, we will be a lot better off in our careers and in life.