Jinshan Jiang

CIS 35B

21 March 2016

Professor Singh

Lessons Learn

1. It is BAD to use array for the structure because using array has to declare it with a certain size; However, using ARRAYLIST is a better structure because ArrayList is like a linked List, it is always suitable for different data size, and it wouldn’t waste memory spaces like array does. For Example, in this quarter, I have been doing this project that I will be able to add car and options anytime I wish. That is when arrayList becomes handy. Here are some code examples:

**private** ArrayList<Option> choice = **new** ArrayList<Option>(1);

**private** ArrayList<OptionSet> opset = **new** ArrayList<OptionSet>(1);

**private** ArrayList<String> allCarsMake = **null**;

allCarsMake.add(**ProxyAutomobile**.*car*.get(key).getBrand()); 🡨 This is how to add to an ArrayList.

1. Why not use static anything?

This is a question that the instructor asked in the class when we were dealing with our project. Static stuff could be very handy when the project is very small and not so much input and output changes. The reason is because that the static stuff will be like “Global Constants”, they are always the same everywhere, and they are not able to change it. This is not going to be benefit to this program because FIRST, this project is very big, it contains so many packages and files. So, when you use static, which means this variable or object is not going to change; Second, the problem with this is that when you need to change something, or trying to do something with them, you program will be messed up. Then it will be very hard to found where you declared those static stuff at to be able to change them. Therefore, using static data type is very bad for the program and it is very not flexible for the design and structure.

1. Finding the patterns is very important for programing because of two reasons. FIRST is because patterns can help the code to be more re-usable, and this should be the fundamental idea of any design of programs. Then, when a program run into any types of after of any type of users, it will come up a solution without crashing. SECOND, patterns helps programs run better because it coverage is wilder, and it can be implemented into different similar issues or projects. For Example, in this program, when I read file, or parsing files. I focused on the pattern of the file. And then it will be able to read all kind of file.

String **brand** = buff.readLine();

String **modelname** = buff.readLine();

**float** **baseprice** = Float.*parseFloat*(buff.readLine());

**int** **optionSetNumber** = Integer.*parseInt*(buff.readLine());

**do** {

**try** {

**if** (baseprice <= 100) {

**throw** **new** ExceptionManager(202, "Missing Base Price!");

} **else** {

auto = **new** Automobile(brand, modelname, baseprice, optionSetNumber);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("~ Base Price Updated!");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

**break**;

}

} **catch** (ExceptionManager **e**) {

baseprice = Float.*parseFloat*(e.fix(202));

}

} **while** (**true**); *// infinite loop until it breaks itself*

**while** (!eof) {

String **line** = buff.readLine();

**if** (line == **null**)

eof = **true**;

**else** {

**if** (building(auto, line, counter, buff))

counter++;

**else**

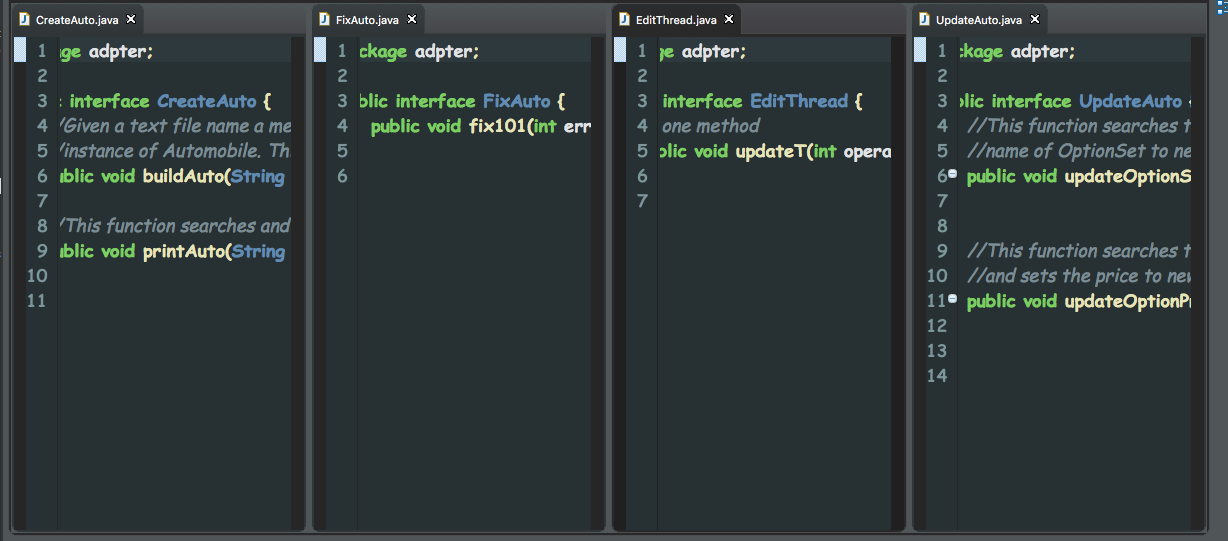
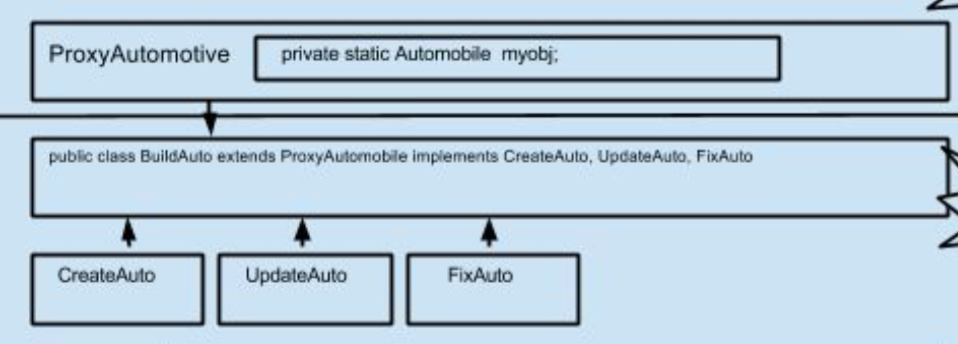
System.***out***.println("Building Error!");

}}

1. Using LinkedHashMap which contains a key, and a value, this helps managing a better data structure. Data structure is sure one of the most important things to concern when started coding a new program, and LinkedHashMap is a good idea to manage created objects. It is basically a Linked Map that is hashed, and that is why it is very good for the structure. For Example.

**private** **static** LinkedHashMap<String, Automobile> *car* = **new** LinkedHashMap<String, Automobile>();

a1.put(auto.getBrand(), auto); 🡨 This is how I added new object to the LHM

1. Talking about Interfaces, interfaces are a general idea of what will be in the code, and what is going to be implemented. For building interfaces program, its good for the program structure because it is much more clear and organized. Also, another big benefit from using interfaces is to build a better capsulation structure for the program. Here is how I implemented in my program, and a design diagram that clear shows how are they going to make capsulation. 
2. Using Synchronized Methods, in another words, multithreading. The benefit of using multithreading is to be able to run programs more efficiently because multiple threads are going to run the same time and its going to use the methods one after the other. Also, using synchronization is good for managing a big program because you will be more easy to control different threads at different paste. For Example:

**public** **synchronized** **void** **updateOptionSN**(String brand, String modelname, String optionSetname, String newName) {

**while** (ava == **false**) {

**try** {

System.***out***.println("Waiting to update Name...");

wait();

} **catch** (InterruptedException **e**) {

System.***out***.println("Done waiting! ");

}

}

Automobile **a1**;

a1 = **this**.findAuto(brand);

a1.updateOptionSet(optionSetname, newName);

ava = **false**;

notifyAll();

}

**public** **synchronized** **void** **updateOptionP**(String brand, String modelname, String optionSetname, String option,

**float** newprice) {

**while** (ava == **true**) {

**try** {

System.***out***.println("Waiting to update Price...");

wait();

} **catch** (InterruptedException **e**) {

System.***out***.println("Done waiting! ");

}

}

Automobile **a1**;

a1 = **this**.findAuto(brand);

a1.updateOption(a1.getOptionSet(a1.findOptionSet(optionSetname)), option, newprice);

ava = **true**;

notifyAll();

}

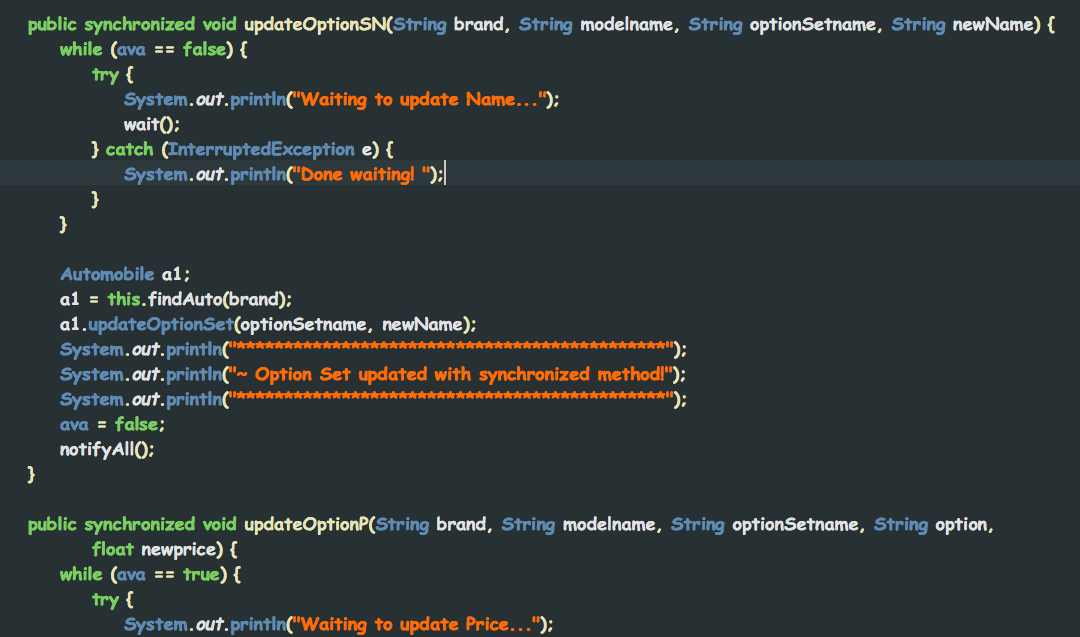
1. What implementation strategy can be used for creating a race condition for testing Multithreading?

To be able to create the race condition for testing multithreading, we need to put both threads to the sleep and then notifyall(), then, there the race started. Whichever got the chance to get the methods, the rest of them goes to sleep again for the next time wake up.

1. What strategy is used for synchronizing, so you end up with a scalable application?

The strategy for using synchronizing is toe analysis the methods that need to be synchronized, and then, make the order of which one executing first, and later. Then implement the synchronized methods.

1. I still remember in the lecture that professor Singh talked about marketing people and programmers. It is always important to understand each other to make our work easier. As a programmer, we should write code more marketing people friendly, which should contain the design diagrams and code documentations that can help people to understand the code better.
2. In the class updateOptionPrice and updateOptionSetName methods in EditOption class are designed to be synchronized for multithreading because when working on the same instance of linked hash map, the update methods are having a chance of having conflict of operating. In Edit Option class, I designed a private boolean instance ava, to playing the role of switch, that can turn on different updating methods.



1. What is the advantage of using Serialization? What issues can occur, when using Serialization with Inner classes?

**Built into the system**: You don't need to rely on third-party tools, libraries, or configuration.

**Relatively simple to understand**, at least in the beginning.

**Every developer knows it** (or should). Regardless of whether Java developers approve or disapprove, they are likely to be familiar with serialising Java objects.

**And the issues**: the issues that occur when I was trying to implement it was that I had to implemented everywhere, and it kind of messes my code a little bit, also that it would return exceptions when the serializing goes wrong. Examples:

public class Automobile implements Serializable{

private static final long *serialVersionUID* = 1L;

public class OptionSet implements Serializable{

private static final long *serialVersionUID* = 1L;

1. Using property file become my new favorite because property file is very straight forward and convenient tot use. Instead of passing a whole file, I can just load up the property object and pass that object instead. Also, it is much more easier and more organizing to reading data from the file, and finding the pattern for the parsing. It is very handy when working with big data and big programs. Here are some examples of paring the file, loading and passing the object:

FileInputStream in = new FileInputStream(filename);

Properties props= new Properties();

props.load(in);

for (int o = 1; o <= optionSetNumber; o++) {

options = props.getProperty(i+"Option"+o);

StringTokenizer st = new StringTokenizer(options, ",");

String tempOptName = st.nextToken();

int tempOptSize = Integer.*parseInt*(st.nextToken());

auto.setValueofOptionSet(o-1, tempOptName, tempOptSize);

for(int os = 1; os <= tempOptSize; os++)

{

info = props.getProperty(i+"OptionValue"+o+"\*"+os);

StringTokenizer \_st = new StringTokenizer(info, ",");

String tempname = \_st.nextToken();

int tempPrice = Integer.*parseInt*(\_st.nextToken());

auto.setValueofOption(auto.getOptionSet(o-1), os-1, tempname, tempPrice);

}

1. Inner class is defined as a classes that is declared and write inside of another class. This is the type of containment. In this assignment. Options are contained in the OptionSet class, which is a inner class for OptionSet class. By doing this, Option and OptionSet now have relationship and this is a way of managing data. Here is how I implemented

public class OptionSet implements Serializable{

private Option choice;

*// \*\* Inner Class \*\**

*// Option class*

private static final long *serialVersionUID* = 1L;

private String nameOfOption;

private float price;

…

1. When to use association?

The association relationship is a way of describing that a class knows about and holds a reference to another class. This can be described as a "has-a" relationship because the typical implementation in Java is through the use of an instance field. The relationship can be bi-directional with each class holding a reference to the other. Aggregation and [composition](http://java.about.com/od/c/g/composition.htm) are types of association relationships.

1. When to use inheritance?

Inheritance is one of the feature of Object-Oriented Programming. Inheritance allows a class to use the properties and methods of another class. In other words, the derived class inherits the states and behaviors from the base class. The derived class is also called subclass and the base class is also known as super-class. The derived class can add its own additional variables and methods. These additional variable and methods differentiates the derived class from the base class.

1. When to use polymorphism?

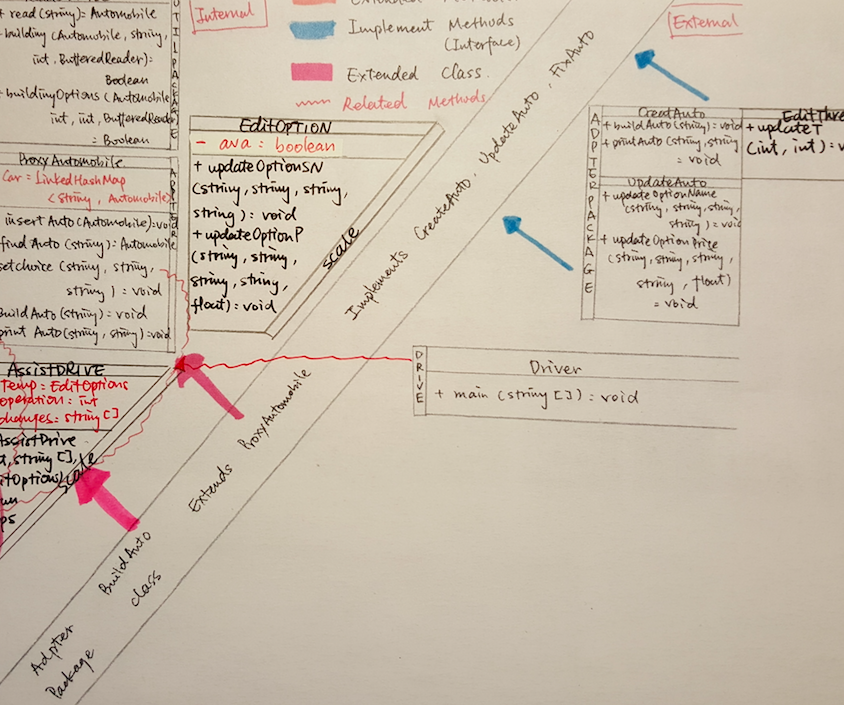
polymorphism is the ability by which, we can create functions or reference variables which behaves differently in different programmatic context. Polymorphism is one of the major building blocks of object oriented programming along with inheritance, abstraction and encapsulation.

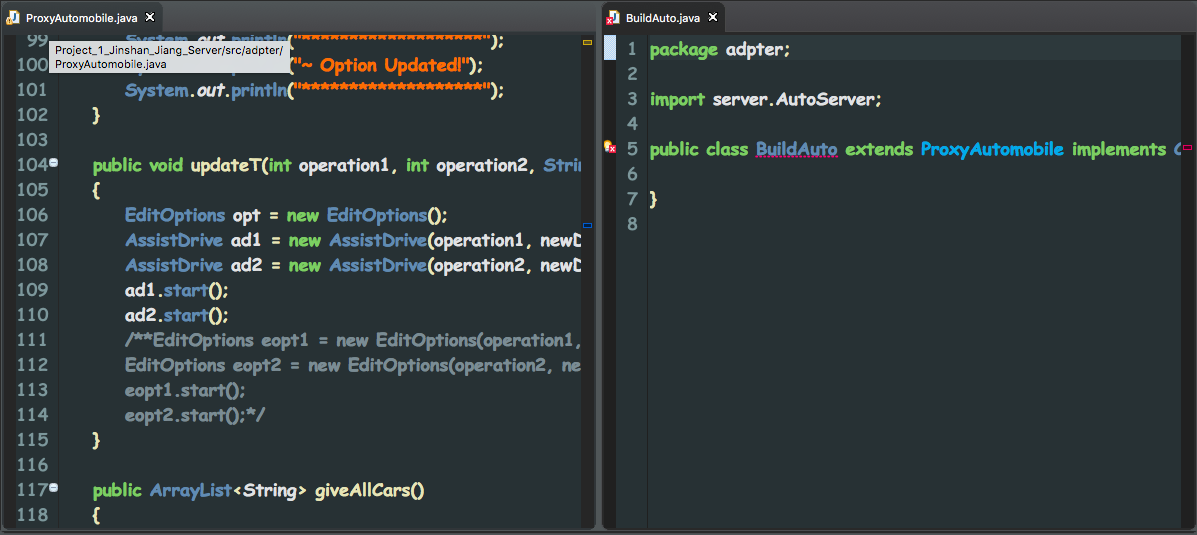
1. When to use capsulation?

Understand encapsulation is the realization that it has two main objectives:

(1) hiding complexity and

(2) hiding the sources of change.

Therefore, to build a encapsulation structure to hide all the structure about configuring the Automobile object. To be able to do the encapsulation, the bridge that I built for the external to access the internal code, I have BuildAuto, which implements all the interfaces but has nothing in the body, and all the methods declared in the interfaces are written in proxyAuto. Here is an example of my encapsulation and a diagram to illustrate that.



1. I still remember Prof. Singh said in the class that “In the future, in the work. 80% of the things you will never heard of, but challenge yourself and study for it. Google it.” This is definitely a golden nugget, or a heads up for my future. But in another work, if I know everything, there are no more challenge, then there are no more fun.
2. Throughout this whole quarter, one thing that Prof. Singh keep have us to do beside coding is to write good documentation. This a good habit for future development. Writing the code documentation can help other people understand my code. And it is good when you have to come back and trying to pick up your code after a long time.
3. Templates are commonly used for a fixed data type. Where has the same amount of different data type. The convenience of using a template is that you don’t need to declare variable or objects over and over again, but using the one is designed already to make the code more efficient.
4. When writing the code, try to make variable, object name as simply as you can because later in the program. When there are tons of variables that you need to use and pass as parameters, it is very hard to identified each of them if they don’t have a unique name that indicates their usage. For example:

private String brand; 🡨 brand indicate automobile brand

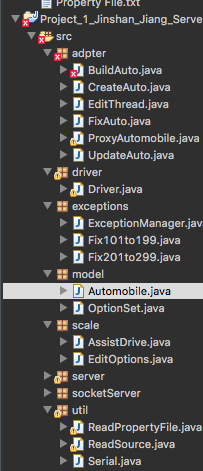
private String name; 🡨 name indicate name of the automobile

private float baseprice; 🡨baseprice indicate the basepriece of the car, and so on…

private ArrayList<Option> choice = new ArrayList<Option>(1);

private ArrayList<OptionSet> opset = new ArrayList<OptionSet>(1);

1. Drawing a diagram is very useful because diagram can help me to design and organize. When writing a big structure program like this. It is very good to have a diagram to assist developing thinking.
2. Using packaging of codes instead of putting everything all together. This a good habit to have when dealing with program that contains a lot of files because this way the codes will be more organized and categorized.



1. Design FIRST, code AFTER. This is definitely a golden nugget that I learn by doing all these assignments. A well designed code can always look neat, simple, understandable, and useful. The best example to give is the exception handling in this program. Since we planed ahead. We were able to log all the errors in error numbers along with a error message, which we also going to use them to fix different errors. Here is some code to show:



1. Driver method, or main method is supposed to be clean and organized. Inside of main, it should only manage and calling different methods to do tasks for the program, but nothing other than that. This shows how organized and well written your program is written.



1. Analyzing requirements - Looking at the data and coming up with a class structure to handle all data - at an abstract level. A good programmer’s job.
2. Self-healing program. This is another big golden nugget in this quarter. Self-healing program is an idea that a programmer must have nowadays when they are designing a new product. This requires the program to be well designed and under no circumstances, the program will crash from anything because there are always supposed to be something to capture the exception, and do something to it.
3. What role(s) does an interface play in building an API?

Interface is playing a very important role in API because interfaces are the most core file in an API. It can be used for communication with other project, also be able to organize the whole API. Therefore, it is very important.

1. Using BufferedReader when read data from a file is a good way of checking data ahead, and it is good to determine the data type. Therefore, it could be very useful when dealing with data that doesn’t follow a certain order.

while (!eof) {

String line = buff.readLine();

if (line == null)

eof = true;

else {

if (building(auto, line, counter, buff))

counter++;

else

System.*out*.println("Building Error!");

}

}

buff.close();

1. A reminder for myself, whenever I want to compare two strings, do NOT use == operator to compare, its not going to tell you that is a error, but == does NOT do anything to the string. USE .equal(string) instead. This has been wasting a lot of my time when I was trying to compare strings, and I had a hard time finding the reason.
2. User friendly program is important as well. To be able to make satisfied customers, or at least you want the to user to know what were you doing. Program need to well written that will prompt up a words that communicate with the user.
3. When using server end and client end, one very important thing to learn which is socket. A socket need the ip address and a gate to build the connection.

SocketClient clientS = new SocketClient("127.0.0.1", 1034);

SocketServer sc1 = new SocketServer(1034);

1. Garbage collection is very important for a program. Always remember to garbage collection when everything is done.
2. When dealing with a lot of classes, make sure to understand all of purpose of all the instances in every class. So when declare them, make them the right type of either private, protected, or public. This is small, but it is very important for building structures.
3. What strategies can be used to design core classes, for future requirements, so that they are reusable?

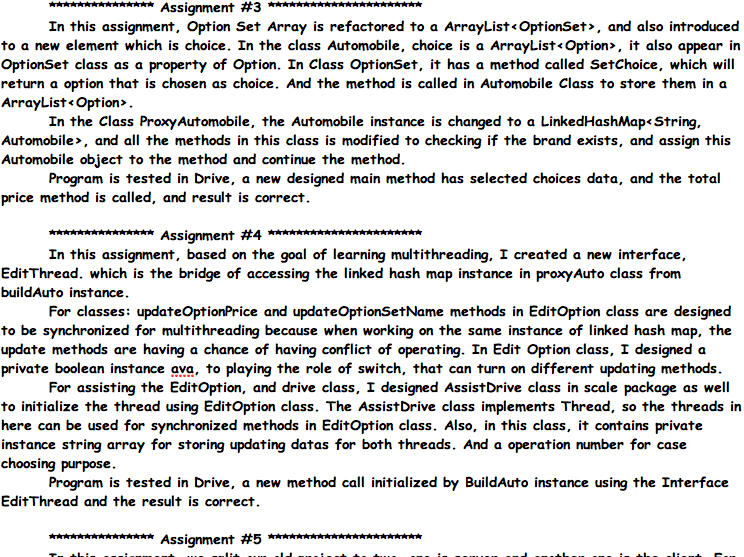
First, you need to analysis the requirement, then read about the data, or the input source. The most important thing is looking for patterns of all of them. All the code is not supposed to be coded based on individual cases, but in a more common cases, and that is how to make them reusable.

1. What strategies can be used to design core classes, for future requirements, so that they are extensible?

In order to make the program extensible, the way is to designed well at the first place, a good design at the beginning can build up a good fundamental structure that can be extensible later in develop level.

1. What strategies can be used to design core classes, for future requirements, so that they are easily modifiable?

The trick to make the code easily modifiable is not how good your design is. It most importantly, according to my experience through out this quarter. A good documentation of code can help the developer to understand you code better, and also it shows the purpose of the steps. That’s how to make code easily modifiable.

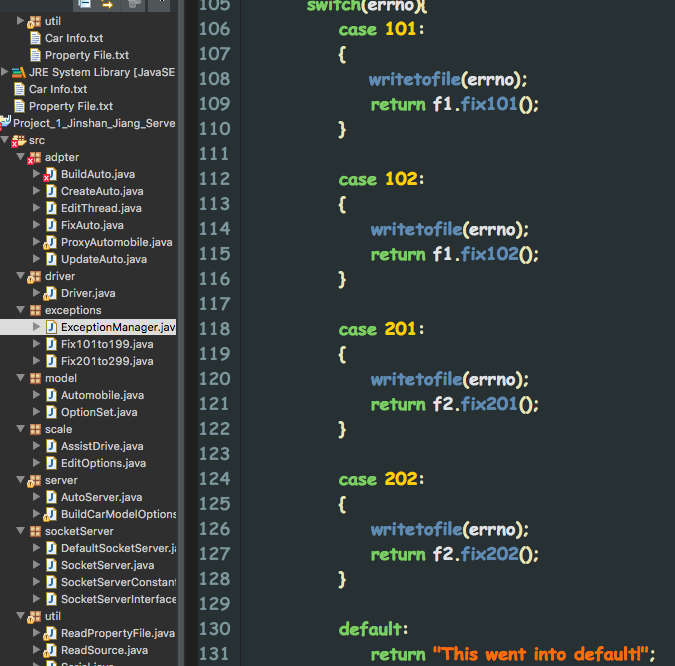


1. Is there any advantage of creating an abstract class, which contains interface method implementations only?

Yes, the advantage is to use interface methods contained abstract class is to encapsulate the inter-structure from exposing. And the same time, implementing the bridges, which are the interfaces.

1. How can you create a software architecture, which addresses the needs of exception handling and recovery?

Creating a exception management, and then log all the exceptions by error number and message. And according to the numbers to category different kind of exceptions. Then, when it comes to the exception handling, it will need to fix method to keep it running from crashing.



1. Through out this quarter, the most important part was not how to write code, but more importantly how to analysis, and find the pattern for the code and design.
2. Why did we have to make the Automobile object static in ProxyAutomotive class?

As we talked about up there how much we should not use static stuff. But we still going to use it because static here becomes handy because when we need to pass the Automobile object here, when we pass this object around, we do not want anything to change the car; Therefore, we put the Automobile Object a static object, so that we can use it anywhere we want and the object is not going to be changes.

1. What is the advantage of adding choice variable in OptionSet class? What measures had to beimplemented to expose the choice property in Auto class?

The advantage of adding choice variable in the OptionSet Class is that in this assignment, Option Set Array is refactored to a ArrayList<OptionSet>, and also introduced to a new element which is choice. In the class Automobile, choice is a ArrayList<Option>, it also appears in OptionSet class as a property of Option. In Class OptionSet, it has a method called SetChoice, which will return a option that is chosen as choice. And the method is called in Automobile Class to store them in a ArrayList<Option>. Choice contains all the user selections, and it will store in the array for later total price calculation.

1. What is the advantage of exposing methods using different interfaces?

By using different interfaces, the code is much cleaner and organized because then different methods regarding different issues will be in the different interface. And when needed to exposing the methods, the particular interface can do it without exposing all the methods.

1. When encounter a new input file, the file could be user friendly, but that is not good for programing. Therefore, analysis the file and reorganize it to the way which is programming friendly. In that way, the file can follow the certain pattern, and the code will be more reusable.
2. Its always good to study, or observing the behaviors of the code, and then modifies it. During this quarter, when I was doing all the assignments, whenever, I encountered something that I’m not familiar with, I would open up a new workspace and try out the new code, and see the behaviors are. That way, I can understand it better and implement it to my code easier.
3. The reason why the property file was loaded in the client side but not the server side was because of that in the realistic world, the server is not going to have the file unless the client pass the file to the server. Therefore, an easier way to manage this is to load it in the property object, and then pass this object to the server end, then parse it from the object.

clientS.run("Property File.txt",1);

public void run(String filename,int choice){

if (openConnection()){

handleSession(filename, choice);

*//closeSession();*

}}*//run*

1. For some checking or updating methods, instead of using void return type, I used Boolean because this is better way for me to track them, and in case there is a bug, this can easily determine where the bug might be.
2. For the big program like this, the simply structure it is, the easier code it will be. The more parameters it passes, the more confused. Therefore, through the whole program, I was trying to keep my code simply, so that when I run into some complex structure, I can still have a clear mind and using the right variables. For example:



1. Reminder to myself, it is important to try to keep my data or instances declarations together in one place instead of being separate all of the places. That way, its easier to manage, and clean up. Also, it is easier for me and other people to read the code.
2. Summary all of plan of attacks, its always analysis (draw design diagram will be handy), modification (programmer friendly source and files), implementations (put in the code and try to make it work without a bug), and debugging (troubleshooting, better structure/ideas implemented. Fixing bugs).