The focus of this topic is to explain the purpose of Abstract Factory and give an example of what is Abstract Factory is and the important of implementation. We will begin our discussion by first explaining what is Abstract Factory and give an example of what Abstract factory is. Let us begin our discussion on what is Abstract Factory by defining what does it mean to be and Abstract factory.

Abstract factory is, "intent: Provide an interface for creating families of related or dependent objects without specifying their concrete classes (Nahar and Sakib p1)". An example of Abstract factory is an interface called credit, where the functions, that are in place into the interface, are simple enough for any class to construct. For instance, the add() method that was inherited in both College and HighSchool class will allows us to add a Node to an ArrayList in either A or B ArrayList. The Node is a subclass, that is in both College and HighSchool, The Node subclass allows us to store in the name of the student, what class they took and the credit hour they receive. Now for the most part High School credits and College credits values are different, However, in this javafx project, we allow any value that is greater than zero be placed into credit. The only retrain we are adding is, the student's name and wither it was a College or High School credit. Another example, is the delete () method that removes a credit, that already exist on either student A or student B and its possible to remove it by determining a student's name, there are only two students A or B, and class name. This will delete a specific course from either student A or student B. If you would like more examples or like to know about other different methods implemented in both College and HishSchool class please look at table Methods list and their descriptions inherited by credit interface. I would like to mention that in order to add to the any one of the lists, the student's name must be A or B and the type of credit must be High School, with space or College.

At this time let us focus on why is it important for an interface to be Abstract factory? There are several reasons why it is important for an interface to be Abstract factory, including creating basic methods that can be implemented by several different classes, like our previous credit interface example. Other reasons to do an Abstract factory is to have some kind of blue print that the programmer can follow when creating other classes. For example, we can use our interface credit but this time lets create a new class call news. You may ask why are we creating a class news when news has nothing to do with College or HighSchool classes. Although that might be true it does not matter because an Abstract factory can be used by any class regardless of what the class is. Ok, so let's get back to our example news class. I would like to mention that news class is not in my Credit project for javafx. It is only an example that we are will using in order to state that even though the methods might be written differently it still constructs the same or similar methods inherited by credit. The class news might have to initialize variables that are different, then College or HighSchool class. For instance, instead of an ArrayList of Nodes we have a list of Articles with amount, values, name of article and so on are being stored. Clearly, we do not have any of those variables for our subclass Node but in the subclass Articles it does. This proves that the interface credit can be used in any class and any class can have

additional methods that affect the given interface methods that where inherited once a class use extends.

To recap what we discuss about Abstract factory we first explain what it means to be an Abstract factory, which is, “intent: Provide an interface for creating families of related or dependent objects without specifying their concrete classes (Nahar and Sakib p1)”. From here we gave examples of different classes that inherited the methods and implemented them differently depending on the class. For instance, College class and news class used the add() method similar ways but College class used to store credit for a student versus news class used it to add an Article to an ArrayList. We also focused our attention on the Credit project and what implementations were made after both College and HighSchool classes inherited credit interface. Also, before I end this great discussion on Abstract factory, I would like to mention that I created some tables that contain major highlights in my Credit project. Please take a look at each table because it will help in creating an entry properly into either A or B list.

Focuses on Credit project, inside College and HighSchool class:

Constructor either College or HighSchool class:

Variables and Type	Variables and Types need to be store
String Student	String a
String classname	String b
int hour_credit	int d
String type	String c
ArrayList A	Initialization
ArrayList B	Initialization

Methods list and their descriptions inherited by credit interface:

Methods	Descriptions
public void add(){}	Adds a Node to A or B ArrayList.
public void delete(){}	Deletes a Node to A or B ArrayList.
public void printALL(){}	Prints all Nodes from A and/or B ArrayList.
public int sizeA(){}	Size of A ArrayList.
public int sizeB(){}	Size of B ArrayList.

Subclass Node list of importance:

- **Creating a Constructor for subclass Node that stores in the following values into:**

Variables and Type	Variables and Types need to be store
String Student	String Student
String classname	String classname
int hour_credit	int hour
String type	String type

How to use:

- You can use these as examples in order to test my code is working if you like or similar format. Afterwards just click **Submit button** to add it to a list and View **button** to see the information you provided. You are also welcome to use the **return button** to go back to the previous page if you would like to add more to be printed out.

	Example 1	Example2
Student Name:	A	B
Name of Class:	History	History 101
Credit Type:	High School	College
Credit Hour:	1	3

Word Cited

1) Nahar, Nadia, and Sakib, Kazi. “Automatic Recommendation of Software Design Patterns Using Anti-Patterns in the Design Phase: A Case Study on Abstract Factory.” Http://Ceur-Ws.org/, 3rd International Workshop on Quantitative Approaches to Software Quality, 2015, ceur-ws.org/Vol-1519/paper2.pdf.