APROC Coursework – Notes

These notes are intended as guidance for the coursework development and also to answer some of the most frequently asked questions - they are not intended to be a step by step guide or to indicate the only way of the application development.

- You must adopt an Object Oriented approach to your application - simply writing all of the code within the main method and perhaps calling a few methods within one class is not acceptable.
- The user interface part of the assignment should be in a separate class, so that changes to the user interface do not affect the rest of the application. This will allow you to develop the program as an application and later change it to an applet using the AWT/Swing to construct the user interface.
- A possible design approach is to view the pipes' types as different classes.
- When you create an object you pass the necessary information to initialize it through a constructor, for example, to create an object that represents a pipe, that the customer wants to order, you can pass to a constructor the necessary information that the customer specifies.
- Your application must determine the type of the ordered pipe, based on the characteristics/parameters specified by the customer. The clients must not be asked what type of pipe they want to order.
- When creating your class hierarchy pay attention which methods and instance variables of your superclass should be inherited. If your sub-classes inherit methods or instance variables that make no sense to have, your class hierarchy is incorrect. If, for example, you have a class that provides your user interface, it is incorrect to use it as a superclass of the classes representing the pipe types, since these classes do not need their own user interface.
- Adopt an incremental approach to the development of your application. When you have one part working, make a copy of the Java files, so that you can return to this point later if you run into a problem.

One possible design of your application (to start with, not the best one) is to create 5 classes (corresponding to each type) and one interface class (could be even your main class) in which you will prompt the client to enter the necessary order parameters (attributes such as, radius/diameter, length, plastic grade, colour, insulation, reinforcement, chemical resistance, quantity, etc.).

You should validate and handle I/O exceptions in this class.

To validate your input, you have to assume some intervals (boundaries) for the input data (which you should describe in your report). For example, minimum and maximum diameter/radius and length of a pipe. Say, the "*PipesR'us"* company does not produce pipes with length less than 0.1 meters (10 centimetres) and larger than 6 meters, and assume it cannot produce more than 100 pipes of one type for a given order, etc. (you can assume different values of course).

When you are designing your class/es, you have to decide what data fields and methods should be included in them. For example, if you decide to work in millimetres, your 'size' variable (length) could be of type ***integer*** (but later, when calculating the cost, you will have to convert the volume into cubic inches, because the cost is given per cubic inch of plastic (Table 2 and Table 3 of the coursework)). If you decide to prompt the length in meters, then the type should be

double, or *float*, etc. Don't forget that you are interested in the volume of plastic material needed and not the whole pipe volume.

Once you have validated the user order, your program should determine, based on Table 1, what is the type of the ordered pipe.

Table 1. *Types of plastic pipes available.*

Type	Plastic's grade	Colour print			Inner insulation	Outer reinforcement	Chemical resistance
		0	1	2			
I	1 – 3	YES	NO	NO	NO	NO	YES/NO
II	2 – 4	NO	YES	NO	NO	NO	YES/NO
III	2 – 5	NO	NO	YES	NO	NO	YES/NO
IV	2 – 5	NO	NO	YES	YES	NO	YES/NO
V	3 – 5	NO	NO	YES	YES	YES	YES/NO

How to determine the type?

You can use 'brute force' approach: e.g.,

*if (**grade** greater than 0) and (**grade** less than 4) and all others are not ordered*, then the ordered pipe is of **type I**.

(There are more intelligent ways to do this (to implement Table1), and you will get additional points if you have found such. For the beginning, you can work with the 'brute force' approach, and later if you have time, you could try to improve it.)

Once this is done, you can create one pipe object of the corresponding type. In other words, you may have five classes, each one representing one type from Table1.

For example, for class *TypeI*, you may have attributes *length, diameter* and *grade.* You may also have at least two additional methods (apart from the constructors, access, and update methods), one to calculate the volume of a pipe, and one to calculate the cost.

When you are calculating the volume of a pipe, make sure that you have converted the length to inches. Once again, make sure that for calculating the cost you use only the volume of the plastic needed and not the whole volume.

Say in your main class you have determined that client's order is a pipe of type I, then you can create an object of *TypeI* and for this object you can call the *cost()* method to calculate the cost and show it to the user.

At this point, you have a working version of your coursework and you are on the 'save side'.

Nevertheless, your application is far from an excellent one. You will notice that your '*type'* classes contain attributes and methods that are repeated, for example, the 'length' data field will be repeated in all of them, also the method for calculating the pipe's volume will be the same for all classes, etc.

So, designing appropriate hierarchy (data methods and data fields at the appropriate level of abstraction) will allow you to fully implement and use Java inheritance and polymorphism techniques in your application.

What left is to develop your GUI interface class using AWT/Swing classes and to add it to your application (if you prefer, you can start with developing your interface first).
Don't forget that you should hand in your coursework report on **4.XII.2015** and give a demonstration (submitting a disk with your source code and Java Netbeans IDE project files) of your software by week 11, during your group lab session.