**Criterion C: Development**

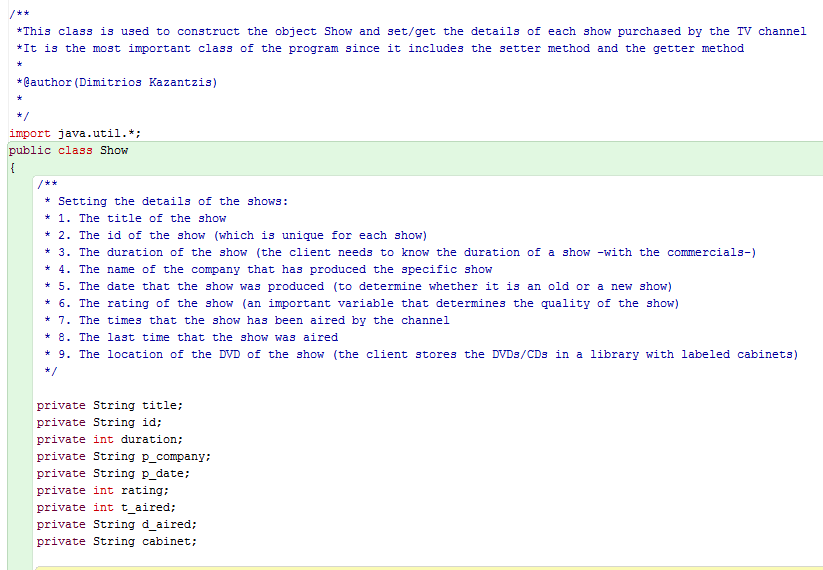
The techniques used in the development of the program are:

1. Encapsulation
2. Polymorphism
3. External Libraries
4. File Access
5. Sorting
6. Recursion
7. Validation
8. Exception Handling
9. Complex use of algorithm
10. Searching
11. Encryption/Decryption
12. User Library

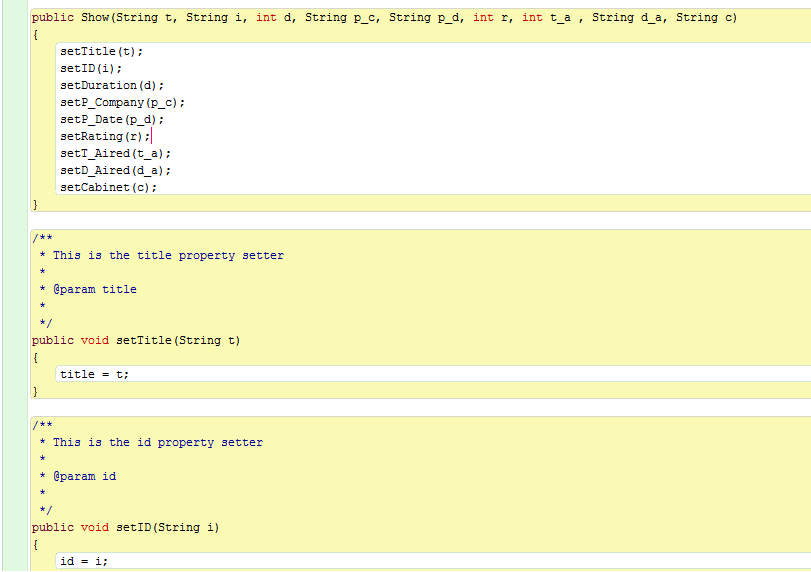
**1. *Encapsulation***

Encapsulation is used in order to increase security and don’t allow an unauthorized user to change the specific data. More specifically, I constructed an encapsulated class by making all the data members of the class private. Then, I use the setter and the getter methods in order to set and get the appropriate data in the class.

This is shown in the **Show** class where the data of the object is defined:



All instance variables are declared as **private**

In the **Show** class the setter method is also used: 

**Mutator** method

**Constructor**

As illustrated above, all the variables have been created as private while they are utilized though public methods (the setter and the getter methods). This is due to the advantage offered by the use of Object Oriented Programming which allows the programmer to make the program extensible and maintainable since it allows code blocks to be interchanged in a more convenient and flexible way by using the same interface.

**2. *Polymorphism***

Polymorphism is found to be useful in the development of my program because it offers the ability to have a common interface and code while using multiple implementations, which increases extensibility.

The rating are different because the User class must get user input whereas the computer class must calculate the go it wishes to take; they both however complete a round in the gam

**3. *External Libraries***

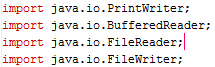
Through the whole program external libraries are used in order to increase usability of existing algorithms. It also reduces the probability of bugs and error in the algorithmic process of the product. Such libraries are found in many classes of the project:

**i.** Library **FileReader** is used to read the contents of the file with the shows purchased.

**ii.** Library **FileWriter** to write/add new data in the file.

**iii.** Library **BufferedReader** provides the ability to read data inputted in the file, while buffering characters in order to make reading more convenient.

Απόσπασμα οθόνης



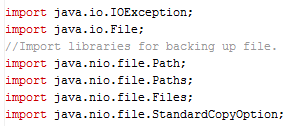
Απόσπασμα οθόνης

These two external libraries were used in order to validate the data that is inputted by the client. More specifically, the library **java.io.IOException** was used in order to validate input data, informing the user that characters cannot be used when choosing an option (in the form of integer). The library **java.util.regex.Pattern** was used to validate letters and numbers used for the ID of each show and also dates (day, month, year).

Απόσπασμα οθόνης

External libraries were also used in order to back up the file with the shows purchased by the TV channel. This method is found in the **BackUp** class as show below.

These libraries were used in order to increase usability and efficiency of the program.





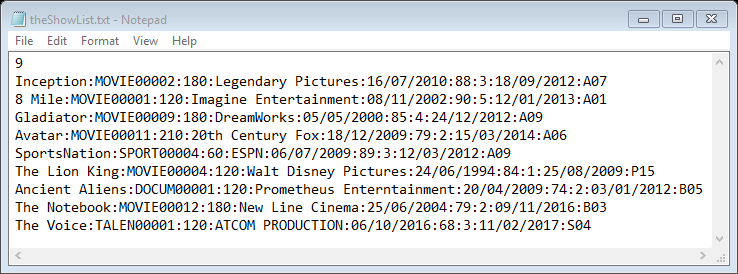
The above screenshot shows how external libraries are used in a public class. It is evident that importing libraries allows greater usability of algorithms and coding. This library helps to back up the data stored in the main file and thus decrease probability of possible data loss to occur. Since the system used by the client is critical, it is very important to use methods which decrease uncertainty and possibility for data loss and increase safety.

**4. *File Access***

File access is very important for the proposed product since it offers the ability for data to be saved in a file even when the program is not running. This file contains the shows purchased by the TV channel. I decided to use a simple txt file to hold the data, since the product should be user-friendly and greater complexity may lead to the creation of a less effective-efficient product. I also decided to name the file as “theShowList”.

Each show has specific details (this is shown in page 1, in the **Show** class).

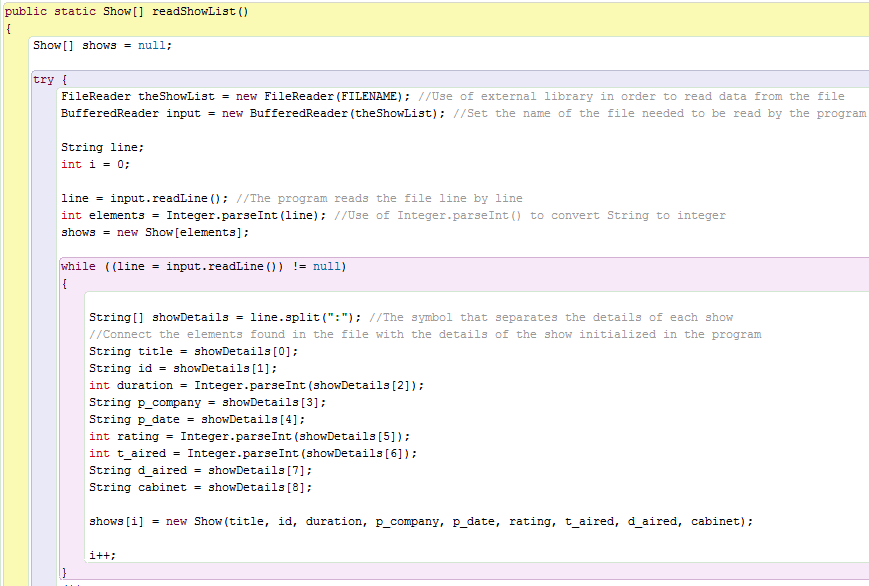
Text File



The code used to construct this method is found in the **ShowList** class. The below method shows the code used in order to program to reader data from the file. As explained each show has specific details which are separated in the file with the use of the symbol “:”

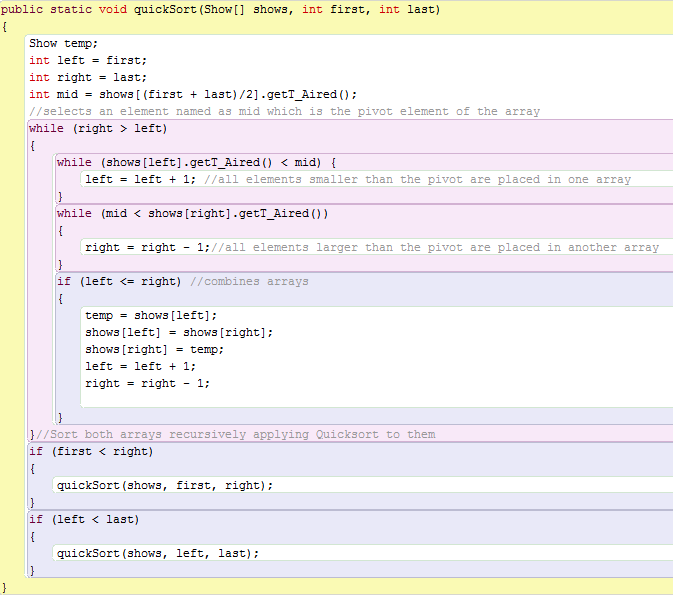
The program reads the file line by line

Reading the file with the name “theShowList”



**5. *Sorting*** and **6. *Recursion***

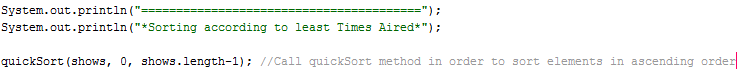
Sorting data was necessary in my program when recommending high quality shows. More specifically, the program in order to recommend shows, it is important to short shows according to least times aired since the TV channel prefers to broadcast shows that have not been played many times in TV. I decided to use the **Quicksort** algorithm since it offers greater complexity, it is easy to implement and it can sort data very quickly. It also offers a **recursive** process that makes the program compact. The use of recursion in the algorithm of my product offers the ability to make the algorithm more understandable, reusable and elegant.



The **partition** part, which is explained on the code

Use of **recursion**

Then, the **quickSort** method is called by the **RecommendShow()** method in the same class in order to sort the elements in ascending order:

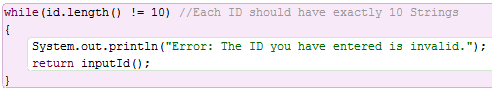


**7. *Validation***

Validating data when inputted is very important for the effective and efficient development of the product. In order to validate data input I decided to check data with the use of **Regex** (regular expression) as is shown below. Since each show has nine details, nine methods are used in order to validate data. However, validation of ID is shown below since this method uses the most demanding validating process in the program.

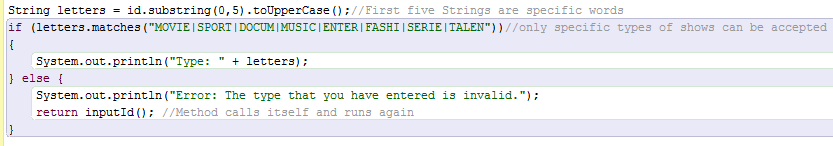
In order to construct the ID of a show a use of a **classification system** is required. Thus, I decided to set each ID of a show to have five letters and five numbers. The first five letters determine the type of the show and the five numbers allow the program to store at most 99999 shows of each type.

**i. Firstly**, the program checks whether the length of the ID is equal to 10, as shown below.



If the data input is invalid the method calls itself and runs again.

**ii. Secondly,** use of **substring** method is required in order to check/validate the first five strings of the id.



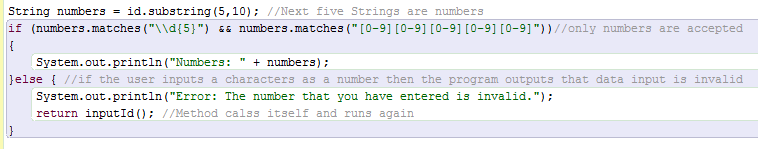
Use of **Regex**

The method of **toUpperCase()** is used in order to increase usability of the program. More specifically, the user can enter the phrase “MOVIE” in every form\* he prefers and it can be read efficiently by the program.

\*The user can either enter “Movie”, “movie”, or “MOvie”, etc.

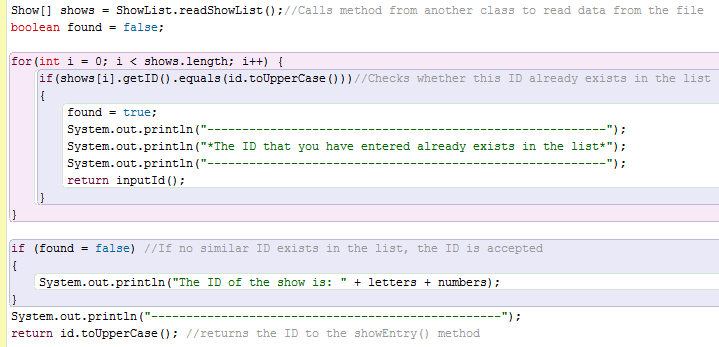
Use of **Regex** to check that only numbers are used

**iii.** If everything functions properly then the program checks the next five strings of the id.



**iv.** Since each show has a **unique** ID the program checks whether the ID entered by the user already exists in the list.

Calls a method from another class in order to read the “**ShowList**”

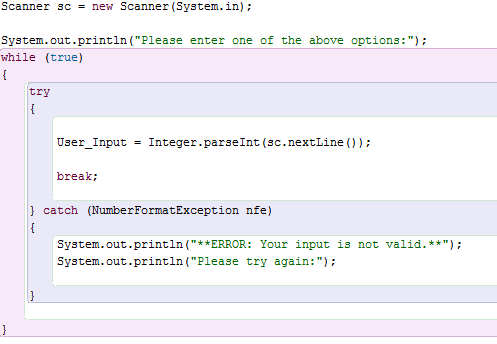


Use of **loop**: the program searches through the list of shows

**8. *Exception Handling***

Exception handling increases functionality and efficiency since it prevents the program from crashing if any runtime error occurs. It is also increases usability since it maintains the normal flow of the product[[1]](#footnote-1).

The use of Checked Exception is evident in page 4, in the method that backs up the original file. However, exception handling can be also found in other classes, such as in the **Main\_Menu** class where the user is asked to choose 1 of the 9 options offered by the program.



Use of **Try**-**Catch** method in order to prevent the program from crashing in case the data that the user inputs is not an integer.

**9. *Complex Use of Algorithm*** and **10. *Searching***

The program uses a **RecommendMethod()** in order to recommend/suggests shows according to specific details. As explained above, the program first sorts the shows according to the least number they have been aired. Then, the program asks the user what type of show does he prefer. Then, the program needs to check the ID of each show to determine its type. This method uses complex algorithm, since it recreates arrays of shows taking into consideration each time a specific detail.

The program **searches** through the list to find shows with the specific **type** chosen

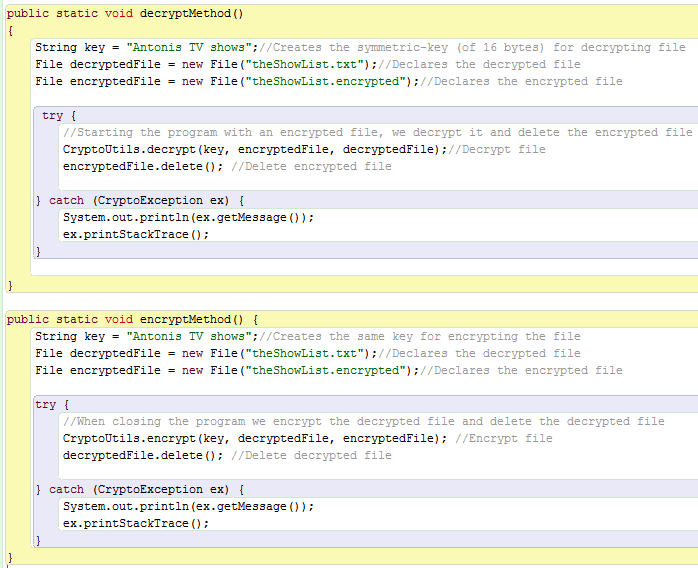
The user enters a certain **duration** since timelines in each TV channel are very strict



The program creates each time a new **array** of shows that have the desired characteristics

**11. *Encryption/Decryption*** and **12. *User Library***

Since the proposed system is critical, encryption and decryption are required in order to secure the application. I decided to use encryption and decryption through **Java Cryptographic Extension** (JCE) framework which offers various standard cryptographic algorithms. Thus, I chose to encrypt and decrypt the main file using the **Advanced Encryption Standard** (AES) algorithm. This type of encryption standard is a **symmetric-key** algorithm that uses the same key in order to encrypt and decrypt data[[2]](#footnote-2).

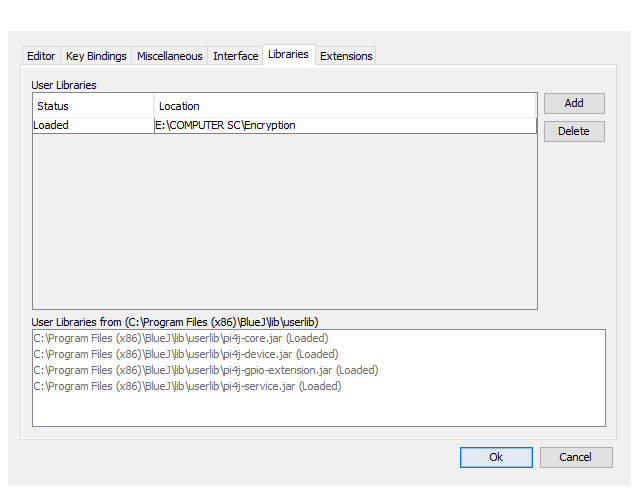


Use of the **same key** to encrypt and decrypt file

The program calls another method\* from a **user library**.

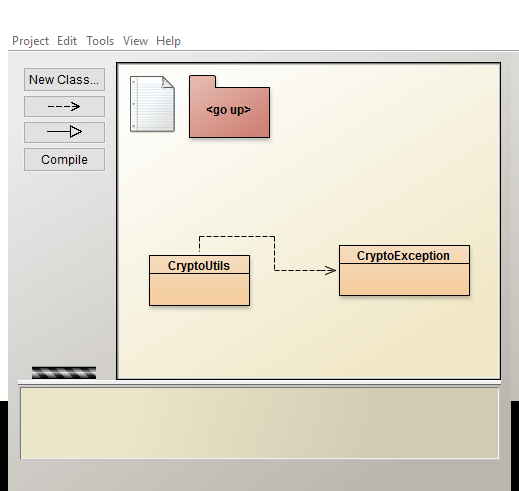
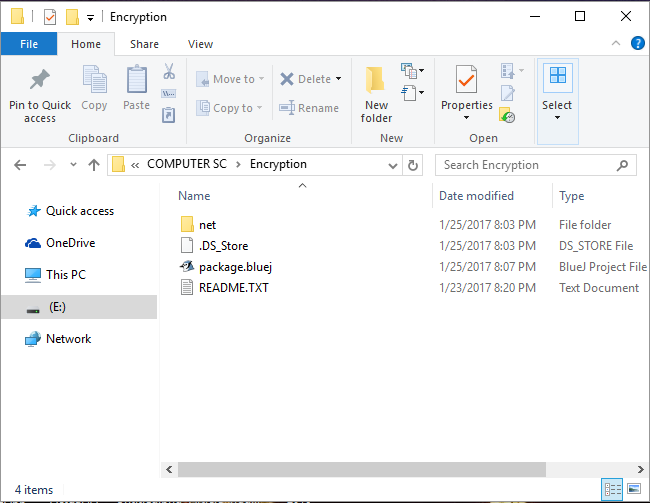
\*Inspired by: <http://www.codejava.net/coding/file-encryption-and-decryption-simple-example>

As shown, these two methods call the **CryptoUtils** method which is located in another project. This was done by calling a **user library** found in another file of the computer. A user library is actually a set of JAR (**Java ARchive**) files. As shown, a user library can be added to a project in order to build path through the option Preferences.



The program loads certain JAR files which hold associated metadata and code that is used in order to encrypt and decrypt files.

The file **Encryption** circled contains the following information:



This package holds the **CryptoUtils** method of which code can be found in the appendix

1. <http://www.javatpoint.com/exception-handling-in-java> [↑](#footnote-ref-1)
2. <http://www.codejava.net/coding/file-encryption-and-decryption-simple-example> [↑](#footnote-ref-2)