**Object oriented programming:**

Object oriented programming is a programming paradigm based on the concept of using objects, a type of data structure. Objects contain fields or in other words also called attributes which relate to the object and code (functions) which are called procedures or methods.

Object oriented programming is solving the problems of procedural programming.

There are 4 main concepts:

* Encapsulation
* Abstraction
* Inheritance
* Polymorphism

**Modularity:**

Modularity is when a big problem is decomposed into sets of modules in order to reduce complexity. Someone might imagine it as a way of mapping encapsulated abstractions into real, physical modules. Modularity is linked to encapsulation. For example, imagine there is a class in python called “Car”. This “Car” class can have attributes like “brand”, “colour” or “mileage” as well as different methods like “start\_engine” or “stop\_engine”.

**Encapsulation:**

Encapsulation means to bundle the data and the methods that work on that data without some unit, like a class in Java into a single unit. It is used to provide privacy and maintain control over the transparency of data at the implementation stage of a project. This means that a user is not able to access state values for all of the variables of a particular object. Encapsulation can also be used to hide data members as well as data functions or methods which are associated with an instantiated class or object (creating a new object for a class by using a new keyword). An example is a calculator, we don’t know how the inside works, but we do know that we can enter 2+2 then “=”. Another example would be Containers, in which data and methods are bundled together into one single package.

**Abstraction:**

Abstraction is when the internal details from an application are hidden from the outer world. It is being used to describe things in more simple terms as well as to create a boundary between the application and the client programs. Additionally, abstraction can be used to create a boundary between the applications and the programmes of the client. A simple example would be making tea with a water boiler. This is because in order for someone to use the water boiler, they would need to know how to use it. Water must be provided and then turned on.

**Inheritance:**

Inheritance is when a class is deriving from another class. This means that the child class is inheriting the parent class’ protected and public properties as well as methods. It can also have its own properties and methods in addition. It can be identified by using the extends keyword.

**Polymorphism:**

Polymorphism is when the set grade method will get overwritten. It occurs when there are many classes which are related to each other through inheritance. Inheritance is letting us inherit the attributes and methods from another class. These methods are then used to perform different tasks.

**Different object oriented programming languages and their uses:**

* ActionScript was originally developed by Macromedia and lets you add complex interactivity, playback, control as well as data display to the application.
* C++ is a programming language and coding language, usually used to develop browsers, operating systems, applications and in-game programming as well as software engineering, data structures etc.
* Common Lisp includes grammarly which is using AI in order to analyse text and also suggest improvements. It is also used for Being which uses a server which is written in the Lisp variant.
* C# is used for general-purpose programming languages for example, to create a number of different programs and applications like mobile apps, desktop applications, cloud-based services, websites and enterprise software and games.
* Dart is a client based language which is optimised for developing fast apps on any imaginable platform. The goal of Dart is to offer the most productive programming language for a multi-platform development while being paired with a flexible execution runtime platform for app frameworks.
* Eiffel is a purely object-oriented language. It is providing an open architecture for interfacing with “external” software in any other programming language. It is also possible for example to program machine and operating system level operation in C.
* Java is an object oriented programming language which is being used to create software on one platform and be able to run it on virtually any other platform.
* Haxe is an object oriented programming language which is an open source software as well as a high-level strictly typed language. It also has a fast-optimising cross compiler. With Haxe you can build cross-platforms applications which are targeting JavaScript, C++, C#, Java, JVM, Python, Lua, PHP, Flash, and allows access to each platform's native capabilities.
* JavaScript is an object oriented programming language which is used to create dynamic and interactive web content like applications and browsers. It is popular because it is the most used programming language in the world.
* Etc.

**What may affect performance and security within object-oriented programs:**

Using object oriented programming, could do the opposite of improving the performance. OOP uses a lot of resources. The performance could be affected because of sloppy or inefficient design/code. This is because the programs within OOP have limited reusability. That means that coupled code could occur which is then difficult to reuse anywhere else. Therefore, it makes it challenging to maintain the base of the code for the long term.

When talking about security, the programs within object oriented programming have insecure data storage and transmission abilities. For example, when credentials or sensitive data is being processed, it is very important to use secure storage and a very secure transmission method. Insecure data storage such as storing passwords in plain, simple text, insecure data transmission or sending sensitive information over unencrypted connections can expose the data to unauthorised access or interceptions.

Object oriented programming uses garbage collection which is a memory management technique. The efficiency as well as frequency of different garbage collection cycles can have an affect on the performance of the programs. This is because if the garbage collection occurs too often or is inefficiently implemented, then pauses or “stuttering” could occur once the program gets executed. This will lead to a huge decrease in performance.

**Importance of principles of Object oriented programming:**

**Abstraction:**

Abstraction in object oriented programming is very important as it gives someone a rough idea of the information without going into much detail. It allows users to create a basic idea of what the problem is and how it can be solved. This means that all specific details are removed and anything that will not help to solve the actual problem.

**Modularity:**

Modularity in object oriented programming is important as it improves the quality of one specific code. This means that when the programme is broken down into smaller parts and each user is taking responsibility for one certain section, the quality of every section will then be improved.

**Encapsulation:**

Encapsulation in object oriented programming is important as it can prevent any unwanted access to sensitive data as well as hide information through access modifiers. Encapsulation can also reduce human errors. Encapsulation is making the maintenance of code more manageable as the code changes can be independent.

**Inheritance:**

Inheritance in object oriented programming is important as it allows developers to create classes which are built on already existing classes. These newly created classes which are based on old ones, inherit the methods of the old as well. This will make a lot of things easier for the developers as there is no need to write the same code again which saves time.

**Polymorphism:**

Polymorphism in object oriented programming is important as a specific routine can be used for variables of different types at different times. Polymorphism also gives a programme the ability to redefine methods for derived classes.

**How does mathematics play a big role in Object oriented programming:**

Mathematics plays an important part in object oriented programming. This is because different mathematicians define new objects in order to contain ideas. OOP allows people to define analogous objects in Python as well as describe their behaviours. An example would be Data structures. Data structures are very important to organise as well as manipulate data within object-oriented programs. Mathematics is also very important when designing and analysing different types of structures like arrays, linked lists, stacks, queues, trees, graphs as well as a hash table (data structure used to store data in key-value format which has direct access to its items immediately). Another thing to consider would be optimisation and numerical methods. This is because there are a lot of programs that are involving problems that need optimization. Mathematical optimization techniques like linear programming, nonlinear optimization and metaheuristic algorithms(higher-level procedure which is designed to find or generate) are used to solve problems efficiently as well as accurately. Numerical methods like numerical integration, solving differential equations or curve fitting are also used in programs that deal with scientific simulations and mathematical modelling.

**How effective is Object oriented programming regarding the principles:**

Object oriented programming has quite a few good things about it. One good thing would be that one reason why OOP is used is because it is all about thinking about and organising code in order to maximise the reusability. This means that a programme is compromising objects that can interact with the user, other objects or programmes. This will make programmes more efficient and easier to understand. The principles are important as:

* Abstraction helps to reduce programming complexity as well as efforts
* Encapsulation prevents unwanted access to sensitive data and also hides information through access modifiers while also reducing human errors
* Inheritance adds public as well as protected methods of the superclass to a subclass, it also allows to replace their implementation.
* Polymorphism allows different implementations of the same method depending on the requirements of the class.

**Introduction Tasks:**

**Refactoring the shop:**

text:class Staff:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

class Staff(Staff):

def \_\_init\_\_(self, name, age, number, group):

self.name = name

self.age = age

self.number = number

self.group = group

def getDetails(self):

return self.name, self.age, self.number, self.group

staffs = []

def addStaff():

staffs.append(Staff(input("Name: "),input("age: "),input("Number: "),input("Group: ")))

def printAllStaffs():

for staff in staffs:

name, age, number, group = staff.getDetails()

print("Name: ", name, "Staff number: ",number)

addStaff()

printAllStaffs()

**Screenshot:**



**What a constructor is:**

A constructor in object-oriented programming is a special method which is automatically called when an object of a class is being created. It is used to initialise all the properties of the object and can also optionally take arguments in order to set the initial values of properties. In python, a constructor is defined by using the “\_\_init\_\_()” method.

An example of a constructor in python:

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

When an object of a class (“Person” in this case) is created then the “\_\_init\_\_()” method is automatically called. The object’s “name” and “age” properties are initialised which would look like this:

person = Person("Alice", 30)

print(person.name) # Output: Alice

print(person.age

**Getters and setters (as well as accessing methods):**

In object-oriented programming, getters and setters are different methods which are providing a controlled access to the properties of an object.

A getter is a method which is getting the value of the properties of an object. This means that it is retrieving the value(s) of the property/properties.

Whereas a setter is a method which is setting the value of the properties of an object. This means that it is updating the value(s) of the property/properties.

They are both often used methods for enforcing certain constraints on the value of an object’s property.

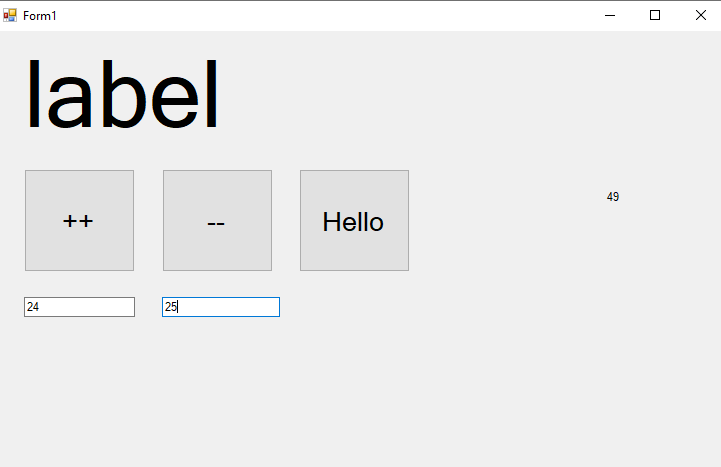
For example:

Getters: A getter might be used to return a calculated value which is based on the current state of an object.

Setters: A setter might be used to enforce that a property’s value is always between the number 0 and 100.

**Visual Studio:**

Here, I have created a Click Counter:



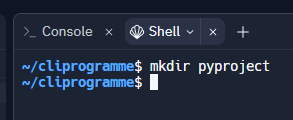
This Counter is able to add as well as subtract to any number typed in and if no number has been typed in it starts to count from 0. Additionally, it can add both numbers inside the 2 text boxes together and display it on the right.

**Task 1 - Creating a CLI programme:**

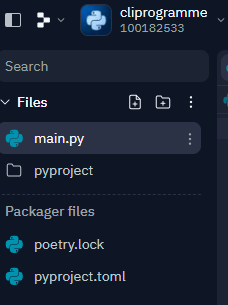
I am going to set up my diary class in which I am gonna store the cli programme code.

The first thing that I am going to do now is to use the code “mkdir pyproject” which means: make directory and the directory is called “pyproject”. You can see below in Figure 1 & 2.

**F1:**

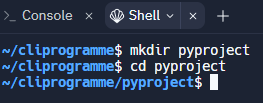


**F2:**



Then I am changing the directory as shown in Figure 3.

**F3:**



As shown in Figure 4 & 5, I have created sub files in my directory.

**F4:**



**F5:**

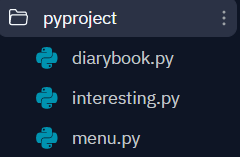


Figure 6 is showing the code that I have entered inside the diarybook.py file.

**F6**

**import datetime**

**#Store the next available id for all new diaries or recent ones**

**last\_id = 0**

**class Diary:**

**'''Represent a diary in the diarybook.'''**

**def \_\_init\_\_(self, memo, tags=' '):**

**'''Initialize a new diary with memo and tags. Creation date of new notes and id are automatically set'''**

**self.memo = memo**

**self.tags = tags**

**self.creation\_date = datetime.date.today()**

**global last\_id**

**last\_id +=1**

**self.id = last\_id**

**def match(self, filter):**

**'''checks if the diary matches the filter text.**

**Return true if it matches exactly, false if it does not match.**

**Filter is case-sensitive'''**

**return filter in self.memo or self.tags**

**Testing if the Diary code is working:**

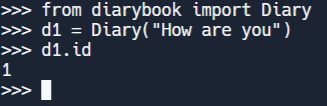
In Figure 7, I used cd python to go to the pyproject directory and then write python to use python and then type in “from diarybook import Diary” (diarybook is the name of one of the files inside the pyproject directory).

**F7**



In Figure 8, it is shown clearly that the testing continued.

**F8**



In figure 9 it is shown that after creating the Diary and me accessing it, I have received the creation date.

**F9**



In figure 10 it is shown that I have checked if d1.match(“How”) is the same as the function, however it should be because d1.match(“How”) is the function.

**F10**



**Summary of the test:**

Firstly, I have created instances of the Diary class and have named it d1.

After that, I have added a memo called “How are you” to the Diary which is used as a remembering function for the Diary.

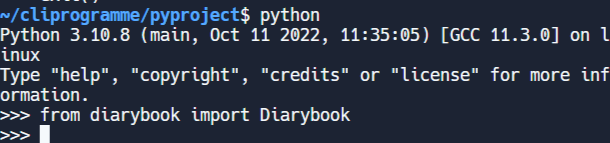
Last but not least, I have revealed the instance id, the date the instance id has been created and also how filter methods can be used to match the text “how” with the memo that has just been created/entered.

**Building Diarybook class:**

By building a diary book class which will contain a list of diaries belonging to users, it allows users to create a new diary, list all the diaries in the diary book as well as to find a particular diary through the filter.

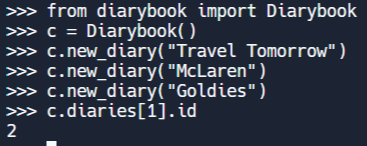
In Figure 11, it is shown how I have solved a problem that occurred. Python was not working properly, so I saved the code, exited python and then entered it again and it worked.

**F11**



As shown in Figure 12, new diaries have been created.

**F12**



**Task 2:**

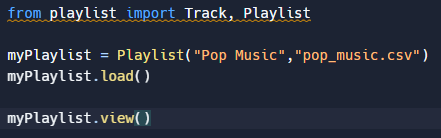
This task is about implementing a class in python in order to maintain an MP3 playlist. This class will be used when developing the software for a music app on a smartphone or an mp3 player.

This MP3 playlist will be stored as a queue of MP3 tracks. When new tracks are being added to the playlist, they are enqueued at the end of the playlist. The key features that you are tol implement within this class are the ability to:

* Enqueue an MP3 to the playlist,
* Remove an MP3 from the playlist,
* Load a playlist from a text file,
* Save a playlist on a text file,
* Shuffle all the songs in the playlist,
* Display all the tracks from the playlist,
* Count the number of tracks in the playlist,
* Calculate the total duration of the playlist,
* Empty/Reset the playlist,
* Check if the playlist is empty.

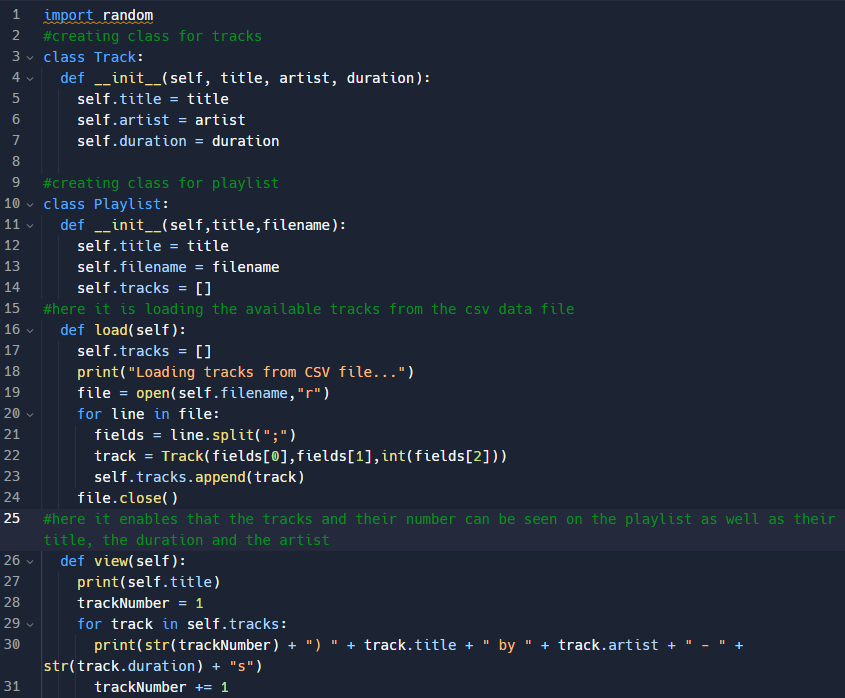
First of all, as shown in figure 1, I started with creating a playlist and used the code to connect to the other 2 files which are 1, the playlist code and 2, the datafile with the song names.

**Figure 1:**



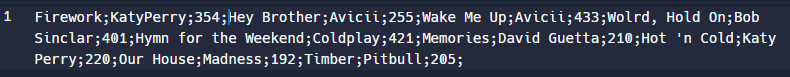
As shown in figure 2, this is the playlist python file. This is where I have created two classes which will be used for the tracks and playlist.

**Figure 2:**



In Figure 3, it shows the playlist that I have created and uploaded in replit.

**Figure 3:**

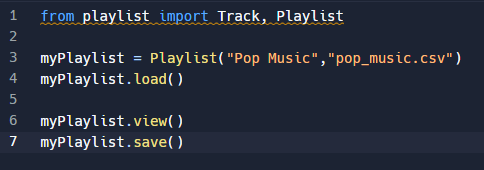


**Now I am going to add methods to my original code in order to fulfil the requirements.**

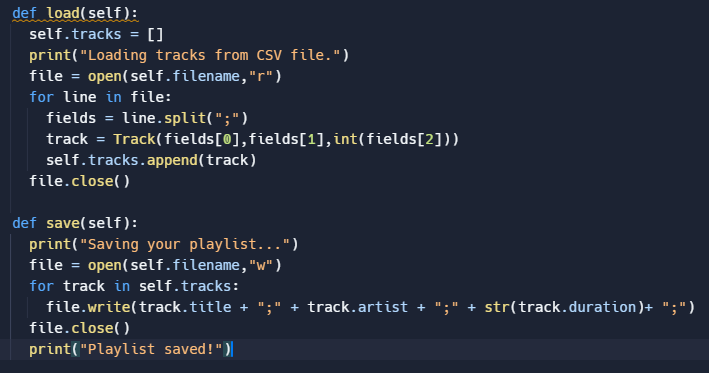
As shown in figure 4 & 5, I added the save function which enables saving the playlist:

**Figure 4 & 5:**

**Main code:**



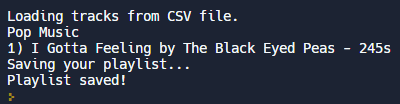
**Playlist code:**



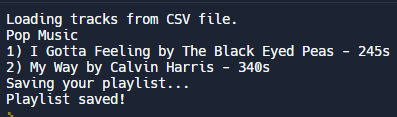
In Figure 6, 7 & 8 I have added a enqueue class method to my code which will add any song that is entered:

**Figure 6 & 7:**

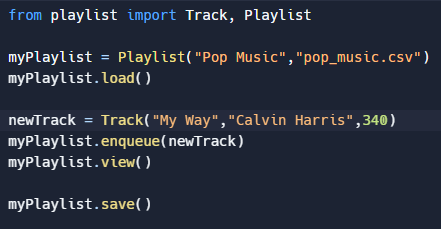
**Before adding the enqueue code:**



**After adding the enqueue code:**

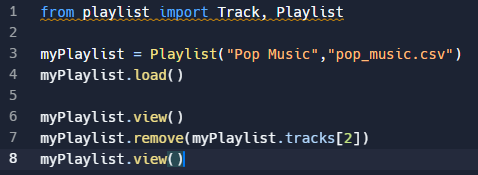


**The enqueue code in Figure 8:**



In figure 9 & 10, I added the remove method in the code which will make it possible to remove a track from the playlist. The reason for that is to test the code.

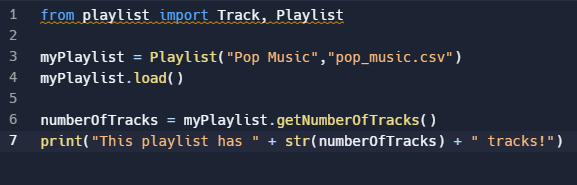
**Figure 9 & 10:**



As shown in Figure 11, 12 & 13 I am letting the system calculate the overall number of the tracks.

**Figure 11, 12 & 13:**

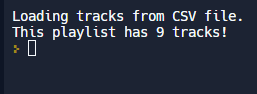
**Main code:**



**Playlist code:**



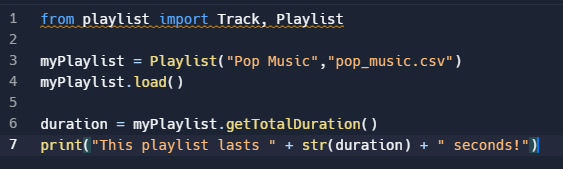
**Result:**



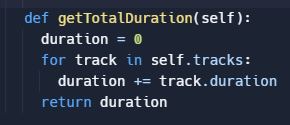
In Figure 14 & 15, I will programme the system so that it will give me the total duration of the tracks.

**Figure 14, 15 & 16:**

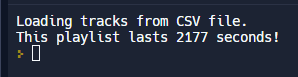
**New code implemented in the main code:**



**New code implemented in the playlist code:**



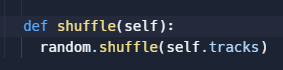
**Result:**



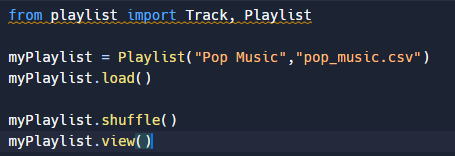
In Figure 17, 18, 19 & 20 I shuffled the tracks so that the tracks played from the playlist, are played randomly.

**Figure 17, 18, 19 & 20:**

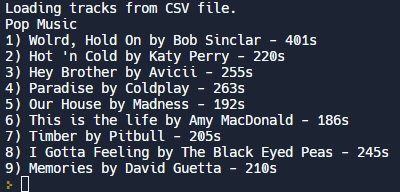
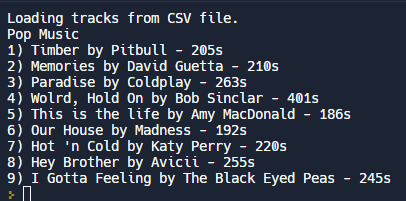
**New code implemented in the playlist code:**



**New code implemented in the main code:**



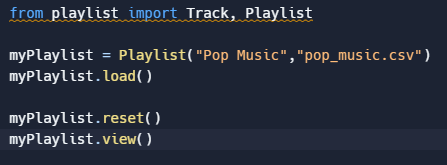
**Result (Images represent randomisation of songs):**



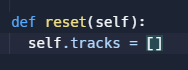
In Figure 21, 22 & 23 I resetted the whole playlist meaning that every song has been removed from the playlist.

**Figure 21, 22 & 23:**

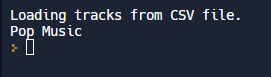
**New code implemented in the playlist code:**



**New code implemented in the main code:**



**Result:**



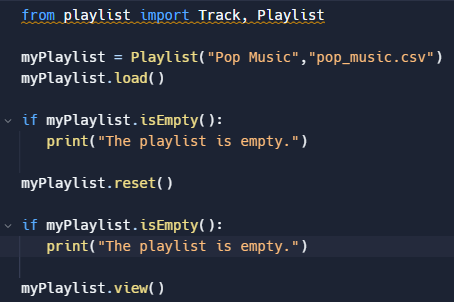
As shown in figure 24, 25 & 26, I check if the playlist is empty. I can also check if the length of its tracks list is equal to 0.

**Figure 24, 25 & 26:**

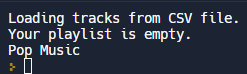
**New code implemented in the playlist code:**



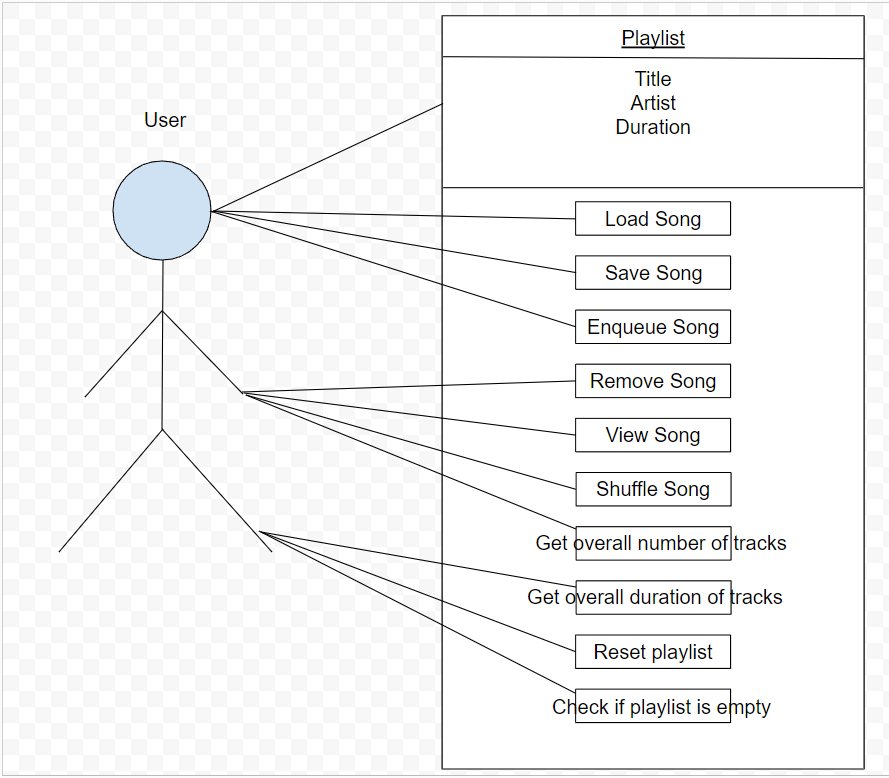
**New code implemented in the main code:**



**Result:**



**Wireframe MP3 Playlist:**



**Task 3:**

This task is about creating a student system programme with a GUI (graphical user interface).

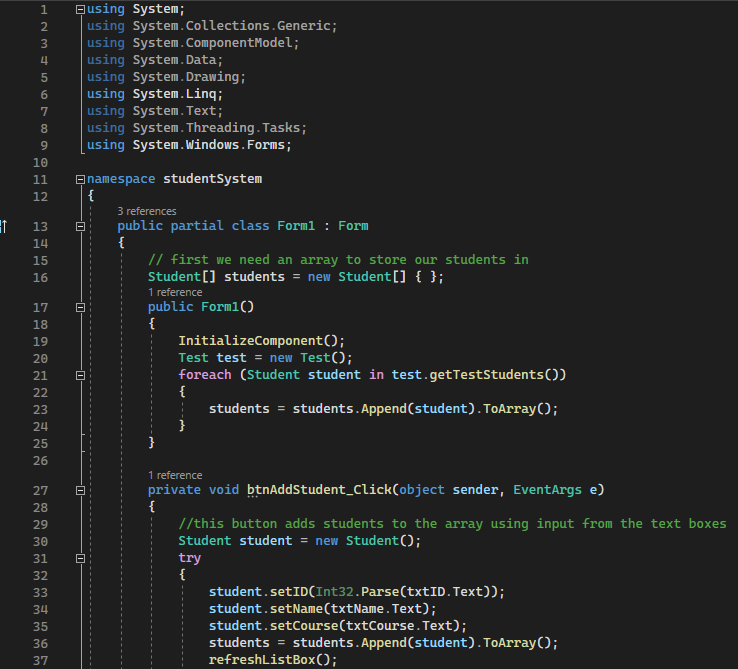
The main features of the system will be:

* Adding a student
* Removing a student
* Search for a student
* Viewing a student’s details
* Being able to modify a student’s (and courses) details

The programme will be designed with OOP so that it can be built upon in order to store data on more than just students e.g. staff, parents. Whilst keeping this in mind, the student class will inherit a person class. All people will also have a name, date of birth as well as an ID number. Students have all the attributes of a person but also have a course.

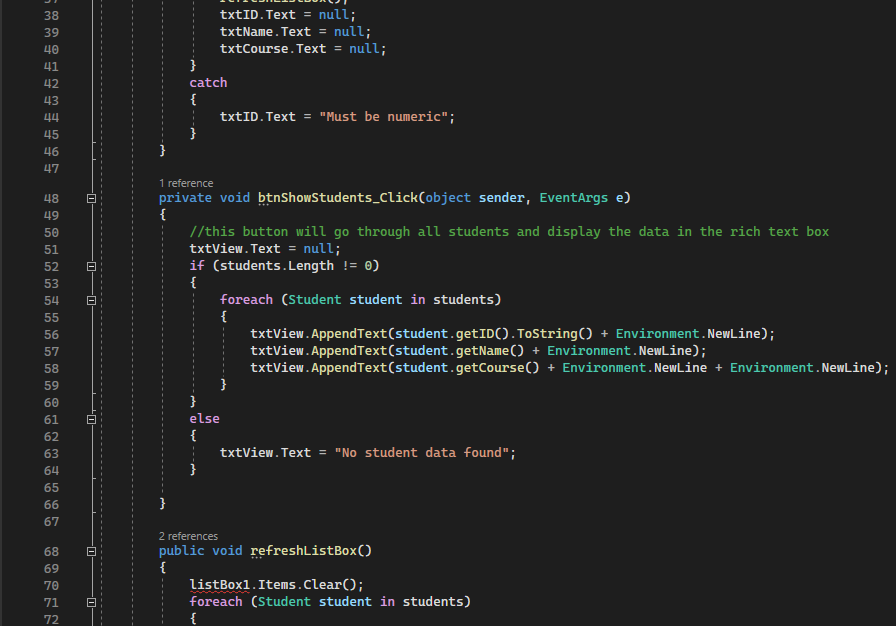
First of all, before creating the Graphical User Interface, I will create the code in order for the GUI to work. For the first part, I started with creating an array in which the students that will be entered can be saved in. See Figure 1 below:

**Figure 1:**



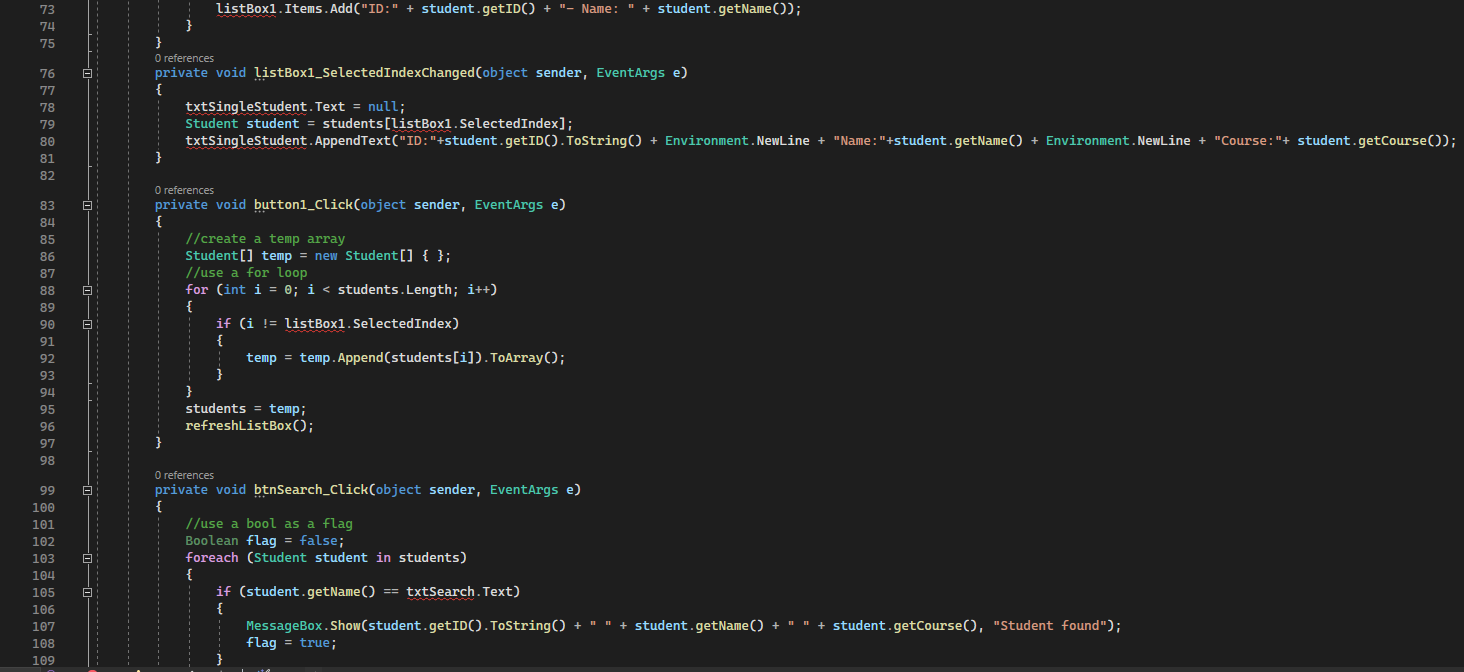
Now, I started creating the functions for the buttons. The buttons are used to collect the data by going through all the students and forward it to the rich text box where the data will then be displayed as shown in Figure 2:

**Figure 2:**

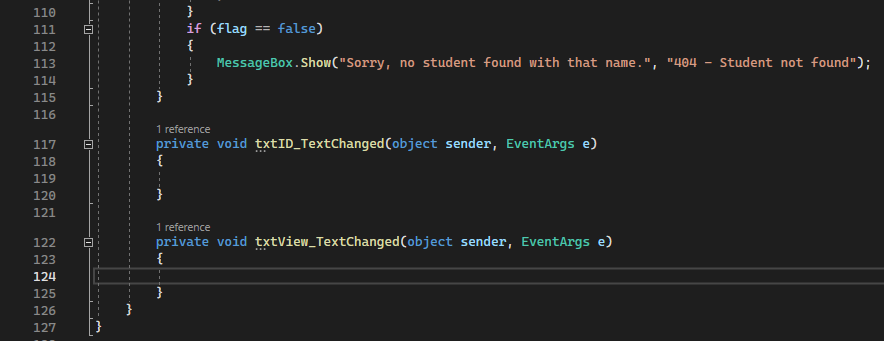


As shown in Figure 3 & Figure 4, I created a temporary array meaning that the array will not be detected/located in the PDV which would mean that it will not appear in a results table. With a temporary array, I can specify the number and size of the array bounds.

**Figure 3 & 4:**

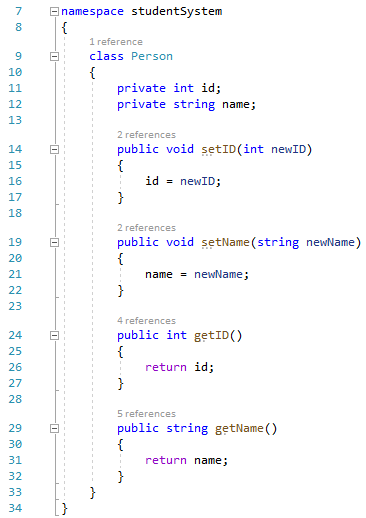


I have set up my GUI which looks like this at the moment:

****

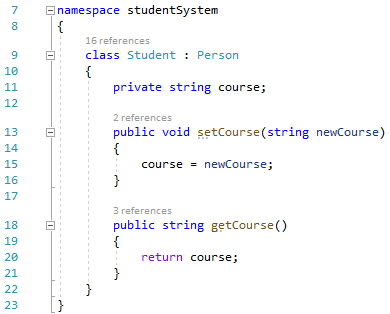
As shown in Figure 5, I have created a class for the Person:

**Figure 5:**



As shown in Figure 6, I have created a class for the Student:

**Figure 6:**



As shown in Figure 7, I created a static class for the programme:

**Figure 7:**

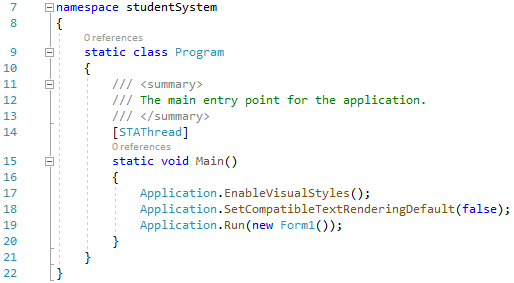
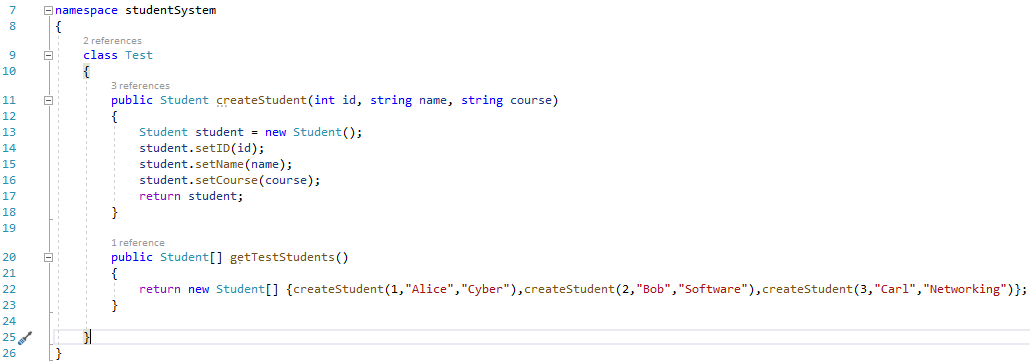


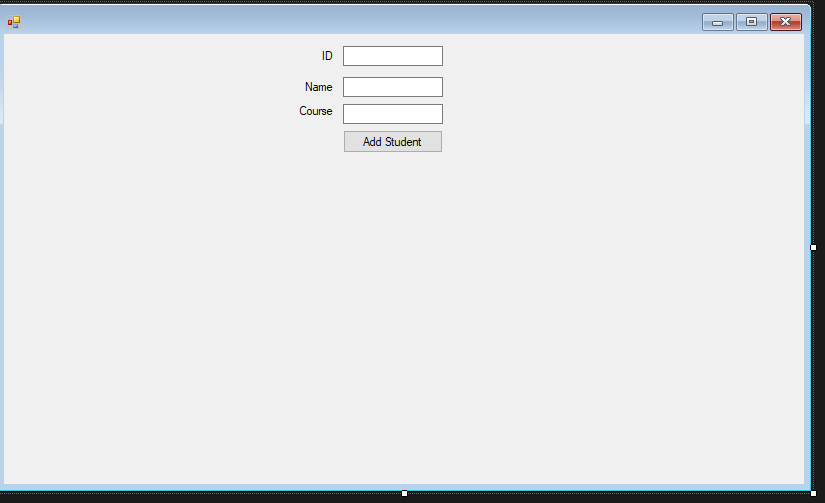
Figure 8 shows how I have tested the programme by creating a class for the test.

**Figure 8:**



Now that I have created the code, I am going to create the Graphical User Interface to add the Students.

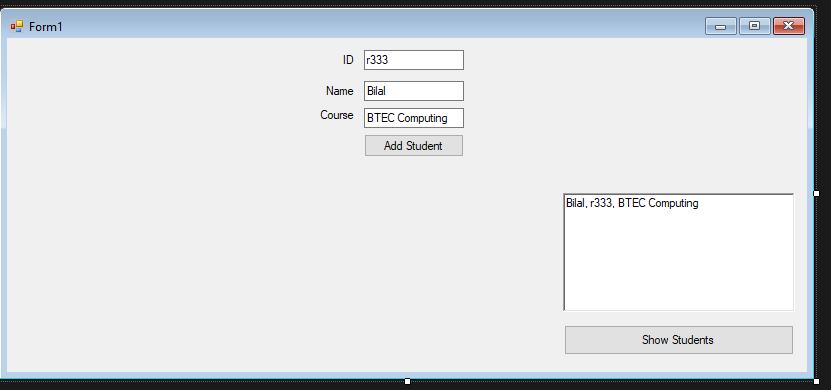
**Figure 9:**



As shown in figure 9, I have created 3 different fields in which the Student’s ID, Name and Course can be entered. At the end, there is an option to Add the student to a list which I will add later on.

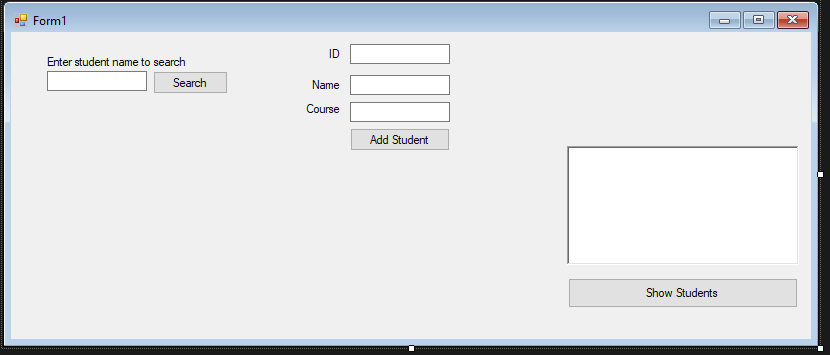
As shown in figure 10, now I have created a rich text box in which the name(s), ID(s) & Course that have been previously entered, are listed:

**Figure 10:**



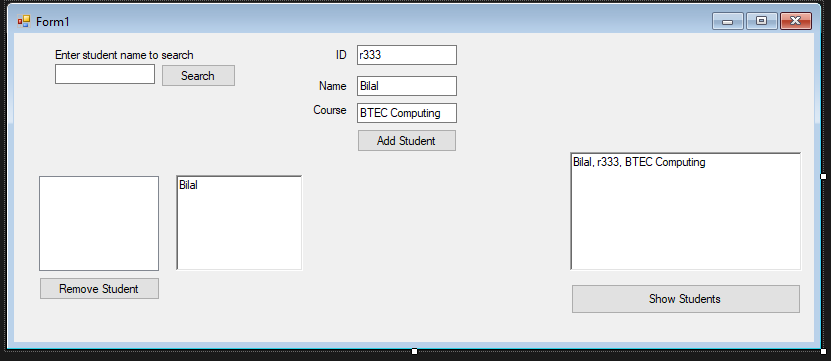
As shown in Figure 11, I have created a search bar, in which the name of the student can be typed in in order to search for it and it will appear in one of the text boxes I will add on in the next step.

**Figure 11:**

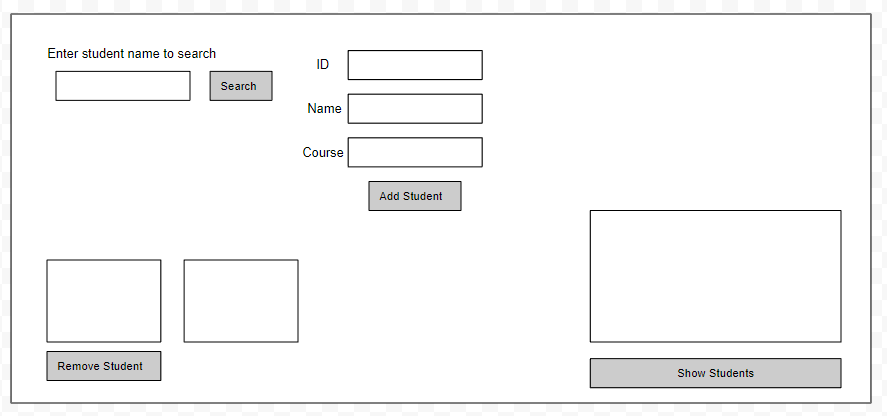


As shown in Figure 12, I have created a text box in which the student name that can be entered from the search bar will appear and one rich text box that will remove the student once entered into the system.

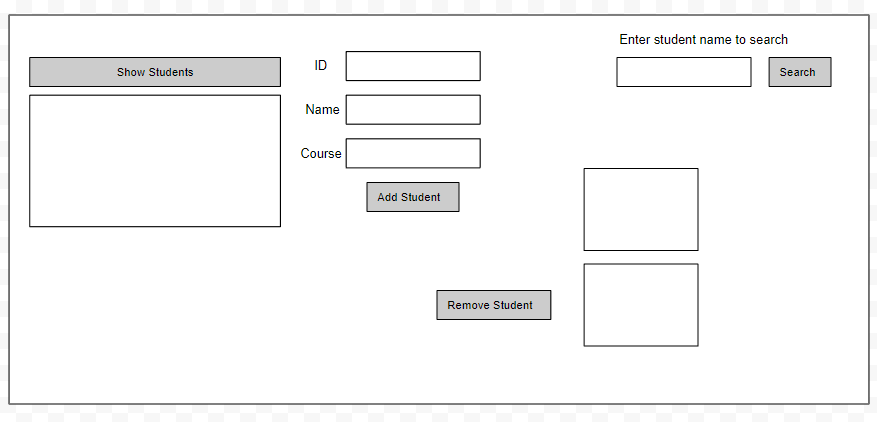
**Figure 12:**



**Initial Wireframe GUI:**



**Alternative Wireframe GUI:**



**Test Plan:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Number** | **Purpose of Test** | **Expected Result** | **Actual Result** |
| 1 | The purpose of this test was to check if a song will be added/enqueued to the playlist | I would expect that there will be one more song added to the playlist. | **Result:** |
| 2 | The purpose of this test was to create sub files in my directory | I would expect that there would be a sub directory created in my CLI programme. | **Result:** |
| 3 | The purpose of this test was check if the system can display the total duration of the songs | I would expect the total duration of the songs to be displayed in seconds | **Result:** |
| 4 | The purpose of this test was to check if the GUI is able to show students | I would expect the GUI to display the name of student(s) | **Result:** |
| 5 | The purpose of this test was to check if the “Person” class was functioning correctly | I would expect the GUI to be working and therefore displaying the ID and Name bars |  |

**Evaluation:**

The purpose of this task is to create a student system programme according to the client requirements which are as follows:

* Create a student system programme using C#
* Student System Programme must contain a GUI

Being able to:

* Add a student
* Remove a student
* View student details
* Edit a student (and course) details

When creating the programme, I used OOP in a way that it can be built upon to store data and not just on one but more than one students, for example…staff or parents. The student class will also inherit a person class. All people will have a name, date of birth as well as an ID number. Additionally, students have all attributes of a person but also have a course. I made sure to fulfil the client requirements by talking about the principles of OOP, what could affect their performance and security, how mathematics is being used within these principles and how important they are & how effective the principles of OOP are. After going through the basics and principles of OOP, I did 2 practise/introduction tasks. The first one is called “Refactoring the Shop” in which I created a shop using OOP in Python. The second is called “Click Counter” in which I created a click counter in visual studio using C#.

After the introduction tasks, I focussed on the actual tasks. Firstly, I created a CLI programme using OOP in Python and taking screenshots along the way of creating it and explaining what I did in each step. After task 1, I completed task 2 in which I created a MP3 playlist with different methods. These methods were allowing the user to: load, save, enqueue/add, remove, view, shuffle songs and get the total number & duration of songs as well as reset the playlist and check if it is empty. For the third task, I used C# in visual studio to create a student system programme with a GUI (graphical user interface). I have also created a class diagram for Task 3 representing my GUI and a wireframe for Task 2 showing what a user is able to do with my playlist.

Personally, there was a time management problem which could be fixed as the deadline was extended meaning that I was successfully able to complete work before the new deadline ended. Other than that, everything went well, especially the distribution of the tasks. This is because the tasks were easy to do, meaning that I could complete the tasks earlier than the expected deadline. Overall, these tasks has really helped me developed with C# as well as Python and Visual Studio. This will help me in the future in case I will do tasks including C#, Python or Visual Studio.