Technical Document

Unit 15 – Object Oriented Programming

By: Chester Descallar

Contents

[Brief 2](#_Toc499800765)

[Hardware Costs 2](#_Toc499800766)

[Software Costs 2](#_Toc499800767)

[Development Cost 2](#_Toc499800768)

[Tools Used 3](#_Toc499800769)

[Visual Studio 3](#_Toc499800770)

[Debugger 3](#_Toc499800771)

[Programming Techniques Used 3](#_Toc499800772)

[Classes and Methods 3](#_Toc499800773)

[Modularity 3](#_Toc499800774)

[Inheritance 4](#_Toc499800775)

[Data Handling 4](#_Toc499800776)

[Entry Validation 4](#_Toc499800777)

[External Data 4](#_Toc499800778)

[Software Structures 5](#_Toc499800779)

[Sequence 5](#_Toc499800780)

[Selection 6](#_Toc499800781)

[Iteration 6](#_Toc499800782)

[Efficiency 7](#_Toc499800783)

[Robustness 7](#_Toc499800784)

[Print Screen Evidence of Application 8](#_Toc499800785)

[Help Page 13](#_Toc499800786)

[Code Listing 14](#_Toc499800787)

[Prison System Test Plan 0](#_Toc499800788)

[Test Results Analysis 6](#_Toc499800789)

[Evaluation 0](#_Toc499800790)

# Brief

The scenario given was to create a prison system for the HM Prison Stafford. This system must allow the user to appoint a governor, warden and also add a prisoner. Along with the system, a design portfolio and evidence of the program in a report was also required to explain the contents of the prison system

Overall, the scenario took 64 hours to do the 3 assignment given. The first assignment was to research about the programming language that the program will be written on and to write a report about the features of object oriented programming and the different tools and techniques that can be used. I started my research on September 13th and finished by 18th of September; this accounted to 7 hours in 3 working days. The second assignment was then started on the 23th of September – this task was to create the design portfolio of the prison system which was to include class diagrams, pseudocode, requirement specifications and class responsibility cards. This was done in 14 hours within 6 working days finishing the second assignment on the 12th of October. The final assignment, was the longest of them all as this required me to implement the prison system, test it, and make a documentation about all of the things that I have done. The final assignment took 47 hours in total. I started the implementation stage on the 16th of October and finished it on the 23rd of November along with the implementation stage, I also did testing and improved the program – totalling to 36 hours of programming, testing and improving the program. The report then took 7 hours between the 24th of November and the 1st of December to finish all of the scenario that was needed for HM Prison Stafford.

## Hardware Costs

This is the specification of the machine that was used to create the program:

Processor: Intel Core i5-650 - £109

Graphics Card: £100

RAM: 3.0GB DDR3 - £18

This is just the basic cost of the machine without the peripherals, however the overall price might cost around £400-£500

## Software Costs

Microsoft Office 2010 - £79

Microsoft Windows 7 Professional - £129

Microsoft Visual Studio 2013 - £310

The software cost of the project would be around £518.

## Development Cost

An average salary for a software engineer is around £35 an hour meaning that it would cost around £2240 (35\*64) to do this assignment

In my program, I have used different tools and techniques, this is to make the application efficient and to ensure that the user can use it well as well as for other programmers to read the code.

# Tools and Programming Techniques Used

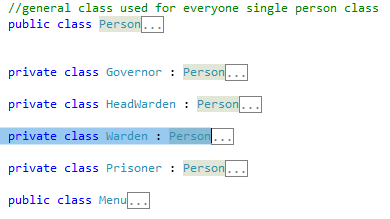
## Visual Studio

Visual Studio was the program that I created the system with – this was done in the language C#. I have used Visual Studio as one of the tools as it is a program that contains a lot of pre-built tools that allows me to use them and so that I can try and create the most efficient code. In visual studio, the program gives you an option to create a console application and this is also in the programming language C# to which is a suitable language to create the prison system. Once created, visual studio gives all the function that is required such as ‘using System’ which helps in the creation of the prison hence why I have chosen to use visual studio as one of my tools.

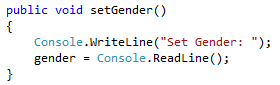
## Debugger

Another tool that I have used in the creation of the application was a debugger, the debugger is another part of what Visual Studio has to provide. I have used debuggers as it allowed to me step through my code line by line and follow how the program flows; by doing so I was able to find errors and that were quite difficult to find without having the debugger and fix them to create the application. Some examples of these are loops not properly looping, by using the debugger it allowed me to find the problem and ensure that every loop is working as expected.

## Classes and Methods

Classes defines the properties and functions of an entity acting as its blueprint. In my prison system, I have used several classes – I created a Person Class in which 4 other classes will inherit from. These are the governor, head warden, warden and prisoner class. Additionally, I also created a main menu class. I have used classes as it is essential to make an object oriented program and it makes the system a lot more efficient.

These are the classes that I have used for the program.

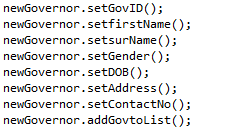
Methods are the procedures associate with the class. Without methods, the program will not be doing it as the method controls the action of the object. Hence why I have created different methods for my classes.

This is a method that I have used in my application

## Modularity

Modularity is another technique that I have used in the creation of my application, by using classes and methods I have multiple independent components and once combined complete the whole system. For example, I have a person class that gets all the personal details of a person and a menu class that displays options in a menu. When those two are put together, the user can look at the menu and choose to input the governor’s personal details. I have used modularity as it minimises codes to be repeated and allows for re-usability. Having modules also means that it certain codes in a module can be freely changed without affect the functionality of the other modules but it might change the system’s functionality. If I changed parts of the person class, it will not affect anything in the menu class.

## Inheritance

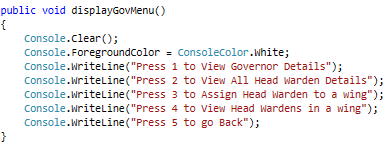
I have also used inheritance during the creation of the application – this was done by the governor, head warden, warden and prisoner classes inheriting the person class. By doing so, it allowed me to reuse codes repeatedly whilst also adding new features to a class without modifying it as sub classes can be created which can use the properties and methods of the base class. For example, I am able to use the methods to set first names, surnames, etc. in the person classes even though I’m using an object that is created from the governor class.

This is the object newGovernor created from the class Governor using methods from the Person class.

## Data Handling

Data handling is needed for the program to control what is inputted and outputted for example

There are different problems when a user inputs data such as inputting the wrong option as they have not followed instructions.

Constants are one of the commonly used data handlers in my program – I have used it to output information to the user which cannot be changed when the program is running. Thus allowing the user of the program to a certain path, controlling the data inputted by the users and preventing errors. For example, this prison system that I have created, I have added a main menu with the selection of options that they can have. All these options have predetermined output and so by displaying it to them, the program know which path it has to take once the program has been inputted.

## Entry Validation

Entry validation is important and useful to ensure that a program is efficient and robust – this is because data that is being inputted and stored is validated by the program before the program crashes. In my program, I have used different data types for variables. For example, part of the unique ID– this way, an increment number can be generated and new unique ID can be created every time a new record is created. Another example if entry validation that I have used is when the user inputs the incorrect option, it tells them that they have inputted the incorrect option and allows them to input another option – by doing so users can easily navigate through the program without having to worry about crashes that will occur if they every input the wrong option. Additionally, entry validation ensures that the program runs and will still function even if the user inputs an invalid data.

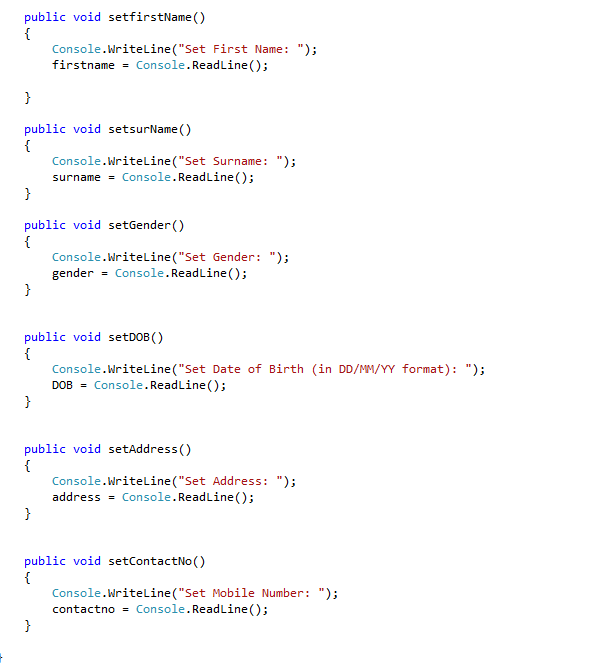
## External Data

External data can be handled in different ways such as accessing it a file from outside the program or in Visual Studio’s using the command to access a file externally as it is a built-in function. Although not included in my program, one thing that I could have used is opening an excel file and storing the data in that so data can be accessed anywhere. Once data is stored, data can also be accessed by doing a different command.

# Software Structures

## Sequence

Sequence is a process where the code in a program are executed on line at a time. In my program, I have decided to use one of the software structures – sequence.

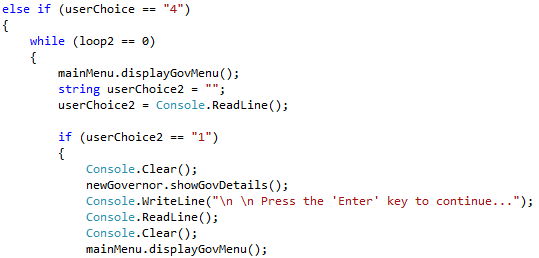
One example of sequence being used in my program is when the user wants to input details of a person such as a prisoner. Upon the user’s input, the user will be asked to input several details but there is always a pattern to it, the user is first asked their name, then surname and it keeps going until they have inputted all the details necessary. After input all the details, the user can then go back to the main menu by pressing enter. A string is used to display this message telling them what to do once they have finished inputting their details so they are not confused.

This is an example of sequence – the user inputs a detail, firstly the first name, then surname then gender, etc.

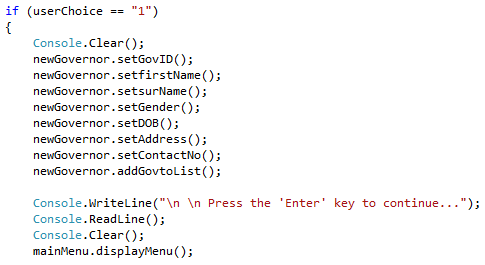
Using selection doesn’t give the user options when using the program but limits what they can do within the program preventing them from ‘breaking’ the program. Additionally, the user wouldn’t have to worry about any confusion as the program sends them to a single direction only – in this case letting the users input details of a person whether it is a governor, warden and prisoner and letting them view those records.

## Selection

Selection or conditionals is used to evaluate whether a condition met depending on the result, to allow one or more statements / codes to run, usually the condition is an expression that uses a comparison operator to compare one value or variable with another.

Throughout my program selection is commonly used. Firstly, selection is used in the main menu of the prison system – the user are given different options to input and these are displayed via a string. Once the user has inputted an option they are sent another page, for example pressing ‘4’ sends the user to the Governor’s Area where they are given more options to do specific things.

This is an example of selection used in the main menu – when 4 is pressed, all the code below it are run. Displaying the governor area sub menu.

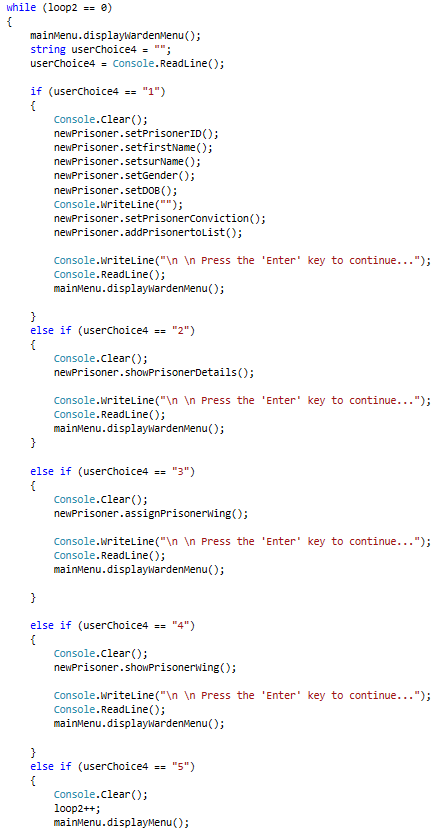
On the other hand, pressing ‘1’ will send the user to an area where they can input the governor details. In this part of the program, selection is used to guide users on what they want to do and give them options that are useable making the overall program more robust and less likely to crash and produce errors.

Moreover, selection is also used when users are assigning a person into a wing. In this part of the program, ‘IFs’ and ‘ELIFs’ statement are used to store a record to wing A or wing B. If the user inputs A, then the record the user wants to store will be stored on wing A. Similarly, an else if statement is used so that if the user inputs B then the record is store in Wing B which can then be accessed later on. Using selection in this part of the program makes it more efficient, that is because I would only need two selection lines to do a certain code instead of having more lines to store the user’s record to a wing which can take up more memory. This also provides the user more options to do in just one area of a code – either Wing A or Wing B.

This is the ‘IF’ selection that asks the user to input the governor’s details.

## Iteration

Iteration or looping is used in my program to continually repeat certain blocks of codes until a condition is met. A "For" Loop is used to repeat blocks of codes a certain number of times depending on what the programmer set it to be. ‘While’ loops are used to repeat certain block of codes an unknown number of times or until a condition is met. Looping makes a code more efficient as instead of repeating the code several times I only need a single ‘while’ line of code to repeat a series of code. It was also quick and easy preventing a programmer from being repetitive.

In my program, I have used ‘While’ loops multiple times. This was mostly to keep the program running if the user inputted the incorrect option and it also ensure that the user is not sent back to the main menu if they input the incorrect option in the sub menu. Using a ‘While’ loop makes it a lot easier and quicker for the user to use prison system as it allows the user to stay in a section area of the program and input as many options as they want in that area without worrying that they have input something in the main menu to go to that area again. This is because the ‘While’ loops enables the program to do lines of code as many times as wanted until a condition is met – which in this case once they user has inputted the correct option and they have finished doing what they need to do in that option.

This is an example a while loop, the user is kept at the sub menu Warden and is asked to input one of the options until they have decided to input option ‘5’ and go back to the main menu

## Efficiency

Efficiency is a vital part of programming as it allows several things such as debugging is easier, files can be smaller and takes up less memory space and it allows programmer to collaborate easily making them understand what the code is all about. I have created a quite efficient program although, in my opinion it could be much more efficient – there are repetitions in my code but this is to make sure that records are stored in different section and that they are all in different array to be called.

## Robustness

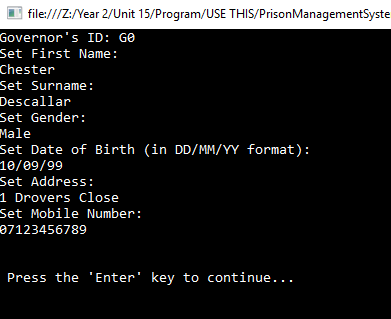
To make my program robust, I have made sure that the program doesn’t end or causes a problem to the user unexpected; instead the user redirected somewhere else where the problem is solved appropriately. This is done by using Else If statements. So that the program does not simply crash or restart but instead handles the error a different way to enable the user to use the program (the prison system). For example, if the user inputs an option that doesn’t exist in the main menu, it tells the user it is an incorrect option and allows them to input another option.

## Menu

I also approached the creation of the prison system using menus and sub menus. This is because by having menus and sub menus, it is a lot easier for the user to navigate the system making everything the program more user friendly and quicker and easier to use. Moreover, in each option of the sub menus and menus I have also changed the colours of each part – this way it is a lot easier for the user to distinguish where they are in the program and it will help them do what they want in that part.

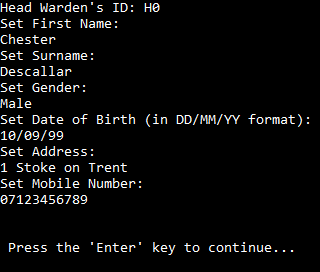
# Print Screen Evidence of Application

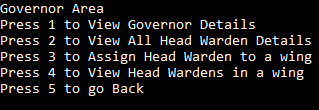
This is the main menu of my program. It allows the user to choose different options. As shown, the user has various different option such as appointing a person or going to one of the ‘area’ to view record. The main menu is also where the help page of the application is located.



When choosing one of the ‘appoint’ options the user will be sent to this page where they can input all their details. In this example, I have chosen option ‘1’ to appoint a governor hence why the unique ID at the top is governor’s ID.

This is the same for the rest of the ‘appoint’ options such as the appoint head wardens and wardens. The only difference between them is that their unique ID changes. As shown on the second appoint option below.



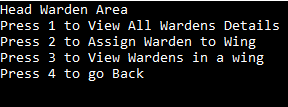


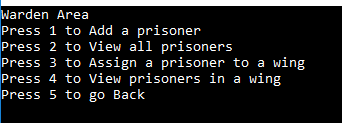
Once the user has inputted a person such as a governor and they have pressed ‘Enter’ they will be sent back to the main menu.

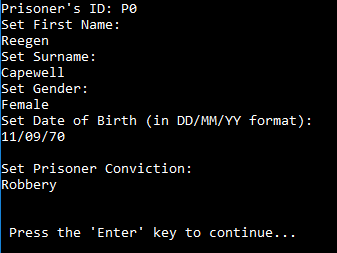
They can then input the option go to the governor’s area. This image shows the sub-menu in the ‘governor’s area’ which allows them to do several things such as view records or assign an employee.

Just like the appoint options, all the other ‘area’ options are also similar to this. Each area allows the user to view certain details and appoint certain people into a wing.

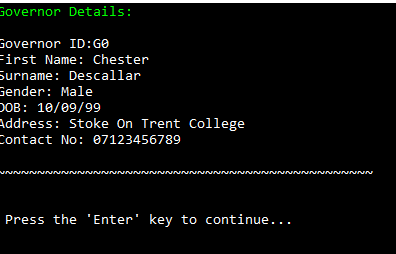
The warden area on the other hand is a variation to the first two as in here, the user will be able to add a prisoner as well – which is one of the criteria wanted by HM Prison Staff.





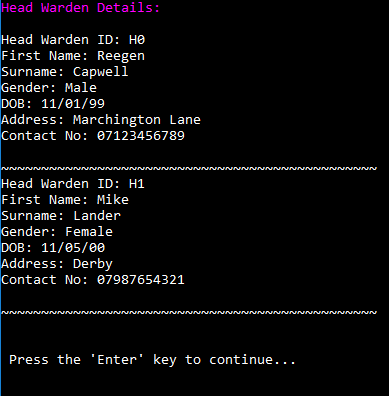


Adding a prisoner is quite similar to the other appoint options however there are some fields missing. Also when adding a prisoner the user is also asked about the prisoner conviction.



When a user wants to view all the records, this is how it is displayed to them. This is the governor record and it is displayed when the user inputs ‘1’ in the governor area sub menu.

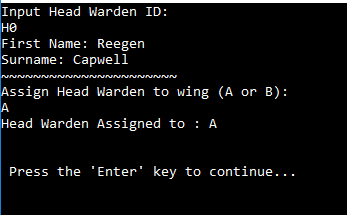
In the governor’s record, every time a new governor is added, the record gets replaced.



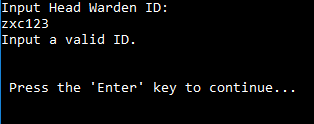
However, in other records such as the head wardens, wardens and prisoners, all the records gets shown and instead of the record getting replaced it gets added to the bottom as shown. This is the same for the other records such as wardens and prisoners when they want to view all records.



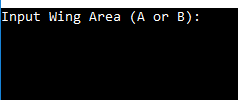
When the user wants to assign a person to a wing, upon their sub menu option input, they are then asked to input their ID, in this case the head warden – this is also the same for the wardens and prisoners.

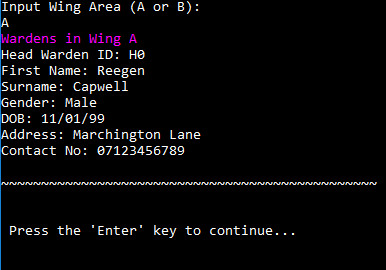


When the user inputs a valid ID, the program shows the first name and surname of that person as shown. After that, the user is then asked to assign that person to a wing. Once they have chosen and inputted what wing they are on, the program then tells the user that that person has been assigned to the wing. This is the same for other assigning options.

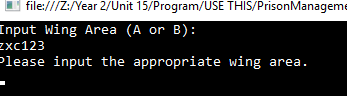


However, as shown when they input the invalid ID, the user is told that they’ve inputted the wrong ID and they are sent back to the main menu.

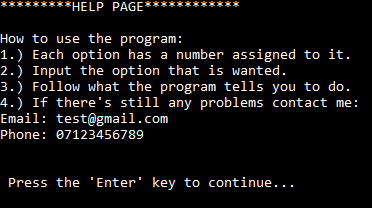




If a user wants to view a record in a wing, they are then asked which wing they want to view. If they input a valid wing, all the records in that wing are displayed. However, if the user inputs an invalid wing, the user is told that it is an invalid wing and they are then asked to put another wing.



## Help Page



This is the help page of the code, this is what the user will be sent to once they have pressed 7 on the main menu. This help page tells the user how they might approach the program.

Help page code:

else if (userChoice == "7")

{

Console.Clear();

Console.WriteLine("\*\*\*\*\*\*\*\*\*HELP PAGE\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine(@"

How to use the program:

1.) Each option has a number assigned to it.

2.) Input the option that is wanted.

3.) Follow what the program tells you to do.

4.) If there's still any problems contact me:

Email: test@gmail.com

Phone: 07123456789");

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayMenu();

# Code Listing

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace PrisonManagementSystem

{

class Program

{

//general class used for everyone single person class

public class Person

{

public string firstname;

public string surname;

public string gender;

public string DOB;

public string address;

public string contactno;

//methods for this class

public void setfirstName()

{

Console.WriteLine("Set First Name: ");

firstname = Console.ReadLine();

}

public void setsurName()

{

Console.WriteLine("Set Surname: ");

surname = Console.ReadLine();

}

public void setGender()

{

Console.WriteLine("Set Gender: ");

gender = Console.ReadLine();

}

public void setDOB()

{

Console.WriteLine("Set Date of Birth (in DD/MM/YY format): ");

DOB = Console.ReadLine();

}

public void setAddress()

{

Console.WriteLine("Set Address: ");

address = Console.ReadLine();

}

public void setContactNo()

{

Console.WriteLine("Set Mobile Number: ");

contactno = Console.ReadLine();

}

}

private class Governor : Person //this class is inheriting the person class allowing it to use the person class' methods

{

//creates are the variable needed for this class

public int listCount = 0;

public string GovFName;

public string GovSName;

public string GovGender;

public string GovDOB;

public string GovAddress;

public string GovContactNo;

public string GovID;

public int rowCount = 0;

public void setGovID()

{

GovID = "G" + listCount.ToString();

Console.WriteLine("Governor's ID: " + GovID);

}

//appends user input into the array

public void addGovtoList()

{

GovFName = firstname;

GovSName = surname;

GovGender = gender;

GovDOB = DOB;

GovAddress = address;

GovContactNo = contactno;

rowCount = 1;

listCount++;

}

//displays what the user has input

public void showGovDetails()

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Governor Details: ");

Console.WriteLine(" ");

Console.ForegroundColor = ConsoleColor.White;

int loopCount = 0;

while (loopCount == 0)

{

if (rowCount == 1)

{

Console.WriteLine("Governor ID:" + GovID);

Console.WriteLine("First Name: " + GovFName);

Console.WriteLine("Surname: " + GovSName);

Console.WriteLine("Gender: " + GovGender);

Console.WriteLine("DOB: " + GovDOB);

Console.WriteLine("Address: " + GovAddress);

Console.WriteLine("Contact No: " + GovContactNo);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

loopCount++;

}

else

{

Console.WriteLine("No Governors Available");

loopCount++;

}

}

}

}

private class HeadWarden : Person //this class is inheriting the person class allowing it to use the person class' methods

{

public string[,] HeadWardens = new string[100, 100]; //array for user input regarding head warden details

//creates are the variable needed for this class

public int rowCount1 = 0;

public int listCount1 = 0;

public string HeadWardenID;

public int x = 0;

//methods for this class

public void setHeadWardenID()

{

HeadWardenID = "H" + listCount1.ToString();

Console.WriteLine("Head Warden's ID: " + HeadWardenID);

}

public void addHWardentoList()

{

HeadWardens[listCount1, 0] = HeadWardenID;

HeadWardens[listCount1, 1] = firstname;

HeadWardens[listCount1, 2] = surname;

HeadWardens[listCount1, 3] = gender;

HeadWardens[listCount1, 4] = DOB;

HeadWardens[listCount1, 5] = address;

HeadWardens[listCount1, 6] = contactno;

HeadWardens[listCount1, 7] = "NONE";

listCount1++;

rowCount1++;

}

public void showHWardenDetails()

{

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Head Warden Details: ");

Console.WriteLine(" ");

Console.ForegroundColor = ConsoleColor.White;

int loopCount1 = 0;

int rowCount1 = 0;

while (loopCount1 == 0)

{

if (rowCount1 < listCount1)

{

Console.WriteLine("Head Warden ID: " + HeadWardens[rowCount1, 0]);

Console.WriteLine("First Name: " + HeadWardens[rowCount1, 1]);

Console.WriteLine("Surname: " + HeadWardens[rowCount1, 2]);

Console.WriteLine("Gender: " + HeadWardens[rowCount1, 3]);

Console.WriteLine("DOB: " + HeadWardens[rowCount1, 4]);

Console.WriteLine("Address: " + HeadWardens[rowCount1, 5]);

Console.WriteLine("Contact No: " + HeadWardens[rowCount1, 6]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

rowCount1++;

}

else

{

loopCount1++;

}

}

}

public void assignHWardenWing()

{

string HWardeninputCheck;

int loopForWing = 0;

int loopforAnswer = 0;

string HeadWardenWing;

Console.WriteLine("Input Head Warden ID: ");

HWardeninputCheck = Console.ReadLine();

while (loopForWing == 0)

{

if (x < rowCount1)

{

if (HeadWardens[x, 0].Contains(HWardeninputCheck))

{

Console.WriteLine("First Name: " + HeadWardens[x, 1]);

Console.WriteLine("Surname: " + HeadWardens[x, 2]);

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~");

while (loopforAnswer == 0)

{

Console.WriteLine("Assign Head Warden to wing (A or B):");

HeadWardenWing = Console.ReadLine();

HeadWardenWing = HeadWardenWing.ToUpper();

if (HeadWardenWing == "A" | HeadWardenWing == "a")

{

HeadWardens[x, 7] = HeadWardenWing;

Console.WriteLine("Head Warden Assigned to : " + HeadWardenWing);

loopforAnswer++;

x++;

loopForWing++;

}

else if (HeadWardenWing == "B" | HeadWardenWing == "b")

{

HeadWardens[x, 7] = HeadWardenWing;

Console.WriteLine("Head Warden Assigned to : " + HeadWardenWing);

loopforAnswer++;

x++;

loopForWing++;

}

else

{

Console.WriteLine("Please input the appropriate wing.");

Console.WriteLine(" ");

}

}

}

else

{

Console.WriteLine("Input a valid ID.");

loopForWing++ ;

}

}

else

{

Console.WriteLine("There are no records available.");

loopForWing++;

}

}

}

//shows the head wardens in a wing

public void showHWardenWing()

{

string HWardeninputCheck2;

int loopAnswer = 0;

int loopForWing2 = 0;

int x2 = 0;

int found = 0;

while (loopAnswer == 0)

{

Console.WriteLine("Input Wing Area (A or B): ");

HWardeninputCheck2 = Console.ReadLine();

HWardeninputCheck2 = HWardeninputCheck2.ToUpper();

if (HWardeninputCheck2 == "A" | HWardeninputCheck2 == "a")

{

while (loopForWing2 == 0)

{

if (x2 < rowCount1)

{

if (HeadWardens[x2, 7].Contains(HWardeninputCheck2))

{

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Wardens in Wing " + HWardeninputCheck2);

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("Head Warden ID: " + HeadWardens[x2, 0]);

Console.WriteLine("First Name: " + HeadWardens[x2, 1]);

Console.WriteLine("Surname: " + HeadWardens[x2, 2]);

Console.WriteLine("Gender: " + HeadWardens[x2, 3]);

Console.WriteLine("DOB: " + HeadWardens[x2, 4]);

Console.WriteLine("Address: " + HeadWardens[x2, 5]);

Console.WriteLine("Contact No: " + HeadWardens[x2, 6]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

found = 1;

}

x2++;

}

else

{

loopForWing2++;

x2 = 0;

}

if (found == 1)

{

}

else

{

Console.WriteLine("Wing Area doesn't have any details or it is not found.");

loopForWing2++;

}

}

loopAnswer++;

}

else if (HWardeninputCheck2 == "B" | HWardeninputCheck2 == "b")

{

while (loopForWing2 == 0)

{

if (x2 < rowCount1)

{

if (HeadWardens[x2, 7].Contains(HWardeninputCheck2))

{

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Wardens in Wing " + HWardeninputCheck2);

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("Head Warden ID: " + HeadWardens[x2, 0]);

Console.WriteLine("First Name: " + HeadWardens[x2, 1]);

Console.WriteLine("Surname: " + HeadWardens[x2, 2]);

Console.WriteLine("Gender: " + HeadWardens[x2, 3]);

Console.WriteLine("DOB: " + HeadWardens[x2, 4]);

Console.WriteLine("Address: " + HeadWardens[x2, 5]);

Console.WriteLine("Contact No: " + HeadWardens[x2, 6]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

found = 1;

}

x2++;

}

else

{

loopForWing2++;

x2 = 0;

}

if (found == 1)

{

}

else

{

Console.WriteLine("Wing Area doesn't have any details or it is not found.");

loopForWing2++;

}

}

loopAnswer++;

}

else

{

Console.WriteLine("Please input the appropriate wing area.");

Thread.Sleep(1500);

Console.Clear();

}

}

}

}

private class Warden : Person //this class is inheriting the person class allowing it to use the person class' methods

{

public string[,] Wardens = new string[100, 100]; //array for user input regarding warden details

//creates all the variables for this class

public int rowCount2 = 0;

public int listCount2 = 0;

public int x = 0;

public string WardenID;

//methods for this class

public void setWardenID()

{

WardenID = "W" + listCount2.ToString();

Console.WriteLine("Warden's ID: " + WardenID);

}

public void addWardentoList() //appends the user input into the array

{

Wardens[listCount2, 0] = WardenID;

Wardens[listCount2, 1] = firstname;

Wardens[listCount2, 2] = surname;

Wardens[listCount2, 3] = gender;

Wardens[listCount2, 4] = DOB;

Wardens[listCount2, 5] = address;

Wardens[listCount2, 6] = contactno;

Wardens[listCount2, 7] = "NONE";

listCount2++;

rowCount2++;

}

public void setWardenWing() //

{

string WardenIDChoice = "";

string WardenWingChoice = "";

Console.WriteLine("Input Warden ID: ");

WardenIDChoice = Console.ReadLine();

Console.WriteLine("Assign Warden to wing (A / B): ");

WardenWingChoice = Console.ReadLine();

}

public void showWardenDetails() //displays all the warden details

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Warden Details: ");

Console.WriteLine(" ");

Console.ForegroundColor = ConsoleColor.White;

int loopCount2 = 0;

rowCount2 = 0;

while (loopCount2 == 0)

{

if (rowCount2 < listCount2)

{

Console.WriteLine("Warden ID: " + Wardens[rowCount2, 0]);

Console.WriteLine("First Name: " + Wardens[rowCount2, 1]);

Console.WriteLine("Surname: " + Wardens[rowCount2, 2]);

Console.WriteLine("Gender: " + Wardens[rowCount2, 3]);

Console.WriteLine("DOB: " + Wardens[rowCount2, 4]);

Console.WriteLine("Address: " + Wardens[rowCount2, 5]);

Console.WriteLine("Contact No: " + Wardens[rowCount2, 6]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

rowCount2++;

}

else

{

loopCount2++;

}

}

}

public void assignWardenWing() //assigns the warden into a wing

{

string WardeninputCheck;

int loopForWing = 0;

int loopforAnswer = 0;

string WardenWing;

Console.WriteLine("Input Warden ID: ");

WardeninputCheck = Console.ReadLine();

while (loopForWing == 0)

{

if (x < rowCount2)

{

if (Wardens[x, 0].Contains(WardeninputCheck))

{

Console.WriteLine("First Name: " + Wardens[x, 1]);

Console.WriteLine("Surname: " + Wardens[x, 2]);

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~");

while (loopforAnswer == 0)

{

Console.WriteLine("Assign Warden to wing (A or B):");

WardenWing = Console.ReadLine();

WardenWing = WardenWing.ToUpper();

if (WardenWing == "A" | WardenWing == "a")

{

Wardens[x, 7] = WardenWing;

Console.WriteLine("Warden Assigned to : " + WardenWing);

loopforAnswer++;

x++;

loopForWing++;

}

else if (WardenWing == "B" | WardenWing == "b")

{

Wardens[x, 7] = WardenWing;

Console.WriteLine("Warden Assigned to : " + WardenWing);

loopforAnswer++;

x++;

loopForWing++;

}

else

{

Console.WriteLine("Please input the appropriate wing.");

Console.WriteLine(" ");

}

}

}

else

{

Console.WriteLine("Input a valid ID.");

loopForWing++;

}

}

else

{

Console.WriteLine("There are no records availble.");

loopForWing++;

}

}

}

public void showWardenWing() //displays the wardens in a certain wing

{

string WardeninputCheck2;

int loopAnswer = 0;

int loopForWing2 = 0;

int x2 = 0;

int found = 0;

while (loopAnswer == 0)

{

Console.WriteLine("Input Wing Area (A or B): ");

WardeninputCheck2 = Console.ReadLine();

WardeninputCheck2 = WardeninputCheck2.ToUpper();

if ((WardeninputCheck2 == "A") | (WardeninputCheck2 == "a"))

{

while (loopForWing2 == 0)

{

if (x2 < rowCount2)

{

if (Wardens[x2, 7].Contains(WardeninputCheck2))

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Wardens in Wing " + WardeninputCheck2);

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("Warden ID: " +Wardens[x2, 0]);

Console.WriteLine("First Name: " + Wardens[x2, 1]);

Console.WriteLine("Surname: " + Wardens[x2, 2]);

Console.WriteLine("Gender: " + Wardens[x2, 3]);

Console.WriteLine("DOB: " + Wardens[x2, 4]);

Console.WriteLine("Address: " + Wardens[x2, 5]);

Console.WriteLine("Contact No: " + Wardens[x2, 6]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

found = 1;

}

x2++;

}

else

{

loopForWing2++;

x2 = 0;

}

if (found == 1)

{

}

else

{

Console.WriteLine("Wing Area doesn't have any details or it is not found.");

}

}

loopAnswer++;

}

else if (WardeninputCheck2 == "B" | WardeninputCheck2 == "b")

{

while (loopForWing2 == 0)

{

if (x2 < rowCount2)

{

if (Wardens[x2, 7].Contains(WardeninputCheck2))

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Wardens in Wing " + WardeninputCheck2);

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("Warden ID: " + Wardens[x2, 0]);

Console.WriteLine("First Name: " + Wardens[x2, 1]);

Console.WriteLine("Surname: " + Wardens[x2, 2]);

Console.WriteLine("Gender: " + Wardens[x2, 3]);

Console.WriteLine("DOB: " + Wardens[x2, 4]);

Console.WriteLine("Address: " + Wardens[x2, 5]);

Console.WriteLine("Contact No: " + Wardens[x2, 6]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

found = 1;

}

x2++;

}

else

{

loopForWing2++;

x2 = 0;

}

if (found == 1)

{

}

else

{

Console.WriteLine("Wing Area doesn't have any details or it is not found.");

}

}

loopAnswer++;

}

else

{

Console.WriteLine("Please input the appropriate wing area.");

Thread.Sleep(1500);

Console.Clear();

}

}

}

}

private class Prisoner : Person //this class is inheriting the person class allowing it to use the person class' methods

{

public string[,] Prisoners = new string[100, 100]; //array for user input to be stored regarding prisoner details

//creates the variable for this class

public int rowCount3 = 0;

public int listCount3 = 0;

public int x = 0;

public string pConviction = "";

public string PrisonerID;

//methods for this class

public void setPrisonerID()

{

PrisonerID = "P" + listCount3.ToString();

Console.WriteLine("Prisoner's ID: " + PrisonerID);

}

public void setPrisonerConviction()

{

Console.WriteLine("Set Prisoner Conviction: ");

pConviction = Console.ReadLine();

}

public void addPrisonertoList() //appends the user input into the array

{

Prisoners[listCount3, 0] = PrisonerID;

Prisoners[listCount3, 1] = firstname;

Prisoners[listCount3, 2] = surname;

Prisoners[listCount3, 3] = gender;

Prisoners[listCount3, 4] = DOB;

Prisoners[listCount3, 5] = pConviction;

Prisoners[listCount3, 6] = "NONE";

listCount3++;

}

public void setPrisonerWing()

{

string PrisonerIDChoice = "";

string PrisonerWingChoice = "";

Console.WriteLine("Input Prisoner ID: ");

PrisonerIDChoice = Console.ReadLine();

Console.WriteLine("Assign Warden to wing: ");

PrisonerWingChoice = Console.ReadLine();

}

public void showPrisonerDetails() //displays all the prisoner details that has been stored into the array

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Prisoner Details: ");

Console.WriteLine(" ");

Console.ForegroundColor = ConsoleColor.White;

int loopCount3 = 0;

rowCount3 = 0;

while (loopCount3 == 0)

{

if (rowCount3 < listCount3)

{

Console.WriteLine("Prisoner's ID: " + Prisoners[rowCount3, 0]);

Console.WriteLine("First Name: " + Prisoners[rowCount3, 1]);

Console.WriteLine("Surname: " + Prisoners[rowCount3, 2]);

Console.WriteLine("Gender: " + Prisoners[rowCount3, 3]);

Console.WriteLine("DOB: " + Prisoners[rowCount3, 4]);

Console.WriteLine("");

Console.WriteLine("Conviction: " + Prisoners[rowCount3, 5]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

rowCount3++;

}

else

{

loopCount3++;

}

}

}

public void assignPrisonerWing() //assigns a prisoner into a wing

{

string PrisonerinputCheck;

int loopForWing = 0;

int loopforAnswer = 0;

string PrisonerWing;

Console.WriteLine("Input Prisoner ID: ");

PrisonerinputCheck = Console.ReadLine();

while (loopForWing == 0)

{

if (x < rowCount3)

{

if (Prisoners[x, 0].Contains(PrisonerinputCheck))

{

Console.WriteLine("First Name: " + Prisoners[x, 1]);

Console.WriteLine("Surname: " + Prisoners[x, 2]);

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~");

while (loopforAnswer == 0)

{

Console.WriteLine("Assign Prisoner to wing (A or B):");

PrisonerWing = Console.ReadLine();

PrisonerWing = PrisonerWing.ToUpper();

if (PrisonerWing == "A" || PrisonerWing == "a")

{

Prisoners[x, 7] = PrisonerWing;

Console.WriteLine("Prisoner Assigned to : " + PrisonerWing);

loopforAnswer++;

x++;

loopForWing++;

}

else if (PrisonerWing == "B" || PrisonerWing == "b")

{

Prisoners[x, 7] = PrisonerWing;

Console.WriteLine("Prisoner Assigned to : " + PrisonerWing);

loopforAnswer++;

loopForWing++;

}

else

{

Console.WriteLine("Please input the appropriate wing.");

Console.WriteLine(" ");

x++;

}

}

}

else

{

Console.WriteLine("Input a valid ID.");

loopForWing++;

}

}

else

{

Console.WriteLine("There are no records available.");

loopForWing++;

}

}

}

public void showPrisonerWing() //displays all the prisoner in a certain wing

{

string PrisonerinputCheck2;

int loopAnswer = 0;

int loopForWing2 = 0;

int x2 = 0;

int found = 0;

while (loopAnswer == 0)

{

Console.WriteLine("Input Wing Area (A or B): ");

PrisonerinputCheck2 = Console.ReadLine();

PrisonerinputCheck2 = PrisonerinputCheck2.ToUpper();

if (PrisonerinputCheck2 == "A" || PrisonerinputCheck2 == "a")

{

while (loopForWing2 == 0)

{

if (x2 < rowCount3)

{

if (Prisoners[x2, 7].Contains(PrisonerinputCheck2))

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Prisoners in Wing " + PrisonerinputCheck2);

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("Prisoner ID: " + Prisoners[x2, 0]);

Console.WriteLine("First Name: " + Prisoners[x2, 1]);

Console.WriteLine("Surname: " + Prisoners[x2, 2]);

Console.WriteLine("Gender: " + Prisoners[x2, 3]);

Console.WriteLine("DOB: " + Prisoners[x2, 4]);

Console.WriteLine("Conviction: " + Prisoners[x2, 5]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

found = 1;

}

x2++;

}

else

{

loopForWing2++;

x2 = 0;

}

if (found == 1)

{

}

else

{

Console.WriteLine("Wing Area doesn't have any details or it is not found.");

loopForWing2++;

}

}

loopAnswer++;

}

else if (PrisonerinputCheck2 == "B" || PrisonerinputCheck2 == "b")

{

while (loopForWing2 == 0)

{

if (x2 < rowCount3)

{

if (Prisoners[x2, 7].Contains(PrisonerinputCheck2))

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Prisoners in Wing " + PrisonerinputCheck2);

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("Prisoner ID: " + Prisoners[x2, 0]);

Console.WriteLine("First Name: " + Prisoners[x2, 1]);

Console.WriteLine("Surname: " + Prisoners[x2, 2]);

Console.WriteLine("Gender: " + Prisoners[x2, 3]);

Console.WriteLine("DOB: " + Prisoners[x2, 4]);

Console.WriteLine("Conviction: " + Prisoners[x2, 5]);

Console.WriteLine("");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

found = 1;

}

x2++;

}

else

{

loopForWing2++;

x2 = 0;

}

if (found == 1)

{

}

else

{

Console.WriteLine("Wing Area doesn't have any details or it is not found.");

}

}

loopAnswer++;

}

else

{

Console.WriteLine("Please input the appropriate wing area.");

Thread.Sleep(1500);

Console.Clear();

}

}

}

}

//main menu class - added to make navigation of the system a lot easier

public class Menu

{

//all the methods in this class outputs a string to the user telling them which buttons to press to progress when using the program

public void displayMenu()

{

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(@" MENU

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Press 1 to Appoint Governor

Press 2 to Appoint Head Wardens

Press 3 to Appoint Wardens

Press 4 to go to Governor's Area

Press 5 to go to Head Warden's Area

Press 6 to go to Warden's Area

Press 7 to get Help

Press 8 to Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

}

}

public void displayGovMenu()

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(@"Governor Area

Press 1 to View Governor Details

Press 2 to View All Head Warden Details

Press 3 to Assign Head Warden to a wing

Press 4 to View Head Wardens in a wing

Press 5 to go Back");

}

public void displayHeadWMenu()

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(@"Head Warden Area

Press 1 to View All Wardens Details

Press 2 to Assign Warden to Wing

Press 3 to View Wardens in a wing

Press 4 to go Back");

}

public void displayWardenMenu()

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(@"Warden Area

Press 1 to Add a prisoner

Press 2 to View all prisoners

Press 3 to Assign a prisoner to a wing

Press 4 to View prisoners in a wing

Press 5 to go Back");

}

}

static void Main(string[] args)

{

int loop = 0;

string userChoice = "";

//creating an object using the classes

Governor newGovernor = new Governor();

HeadWarden newHeadWarden = new HeadWarden();

Warden newWarden = new Warden();

Prisoner newPrisoner = new Prisoner();

Menu mainMenu = new Menu();

mainMenu.displayMenu();

while (loop == 0) //loop to allow the user to see the main menu

{

int loop2 = 0;

userChoice = Console.ReadLine();

{

//asks the user to input the details of the governor that they want to appoint

if (userChoice == "1")

{

Console.Clear();

newGovernor.setGovID();

newGovernor.setfirstName();

newGovernor.setsurName();

newGovernor.setGender();

newGovernor.setDOB();

newGovernor.setAddress();

newGovernor.setContactNo();

newGovernor.addGovtoList();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

Console.Clear();

mainMenu.displayMenu();

}

//asks the user to input the details of the head warden they want to appoint

else if (userChoice == "2")

{

Console.Clear();

newHeadWarden.setHeadWardenID();

newHeadWarden.setfirstName();

newHeadWarden.setsurName();

newHeadWarden.setGender();

newHeadWarden.setDOB();

newHeadWarden.setAddress();

newHeadWarden.setContactNo();

newHeadWarden.addHWardentoList();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

Console.Clear();

mainMenu.displayMenu();

}

//asks the user to input the details of the warden they want to appoint

else if (userChoice == "3")

{

Console.Clear();

newWarden.setWardenID();

newWarden.setfirstName();

newWarden.setsurName();

newWarden.setGender();

newWarden.setDOB();

newWarden.setAddress();

newWarden.setContactNo();

newWarden.addWardentoList();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

Console.Clear();

mainMenu.displayMenu();

}

//displays the governor area sub menu

else if (userChoice == "4")

{

while (loop2 == 0)

{

mainMenu.displayGovMenu();

string userChoice2 = "";

userChoice2 = Console.ReadLine(); //allows the user to choose another option from the sub menu

//displays the governor details upon user input

if (userChoice2 == "1")

{

Console.Clear();

newGovernor.showGovDetails();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

Console.Clear();

mainMenu.displayGovMenu();

//displays all the head warden details upon user input

}

else if (userChoice2 == "2")

{

Console.Clear();

newHeadWarden.showHWardenDetails();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayGovMenu();

}

//allows the user to assign the head warden into a wing

else if (userChoice2 == "3")

{

Console.Clear();

newHeadWarden.assignHWardenWing();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayGovMenu();

//allows the user to view records of head warden in a certain wing

}

else if (userChoice2 == "4")

{

Console.Clear();

newHeadWarden.showHWardenWing();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayGovMenu();

}

//sends the user back to the main menu

else if (userChoice2 == "5")

{

Console.Clear();

mainMenu.displayMenu();

loop2++; //ends the loop sending the user back to the main menu

}

//allows the user to input another option if they input an invalid option

else

{

Console.WriteLine("Invalid Option. Please input one of the options.");

Thread.Sleep (1500);

}

}

}

//displays the head warden area sub menu

else if (userChoice == "5")

{

while (loop2 == 0) {

mainMenu.displayHeadWMenu();

string userChoice3 = "";

userChoice3 = Console.ReadLine(); //allows the user to choose another option from the sub menu

//displays the warden details upon user input

if (userChoice3 == "1")

{

Console.Clear();

newWarden.showWardenDetails();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayHeadWMenu();

//allows the user to assign the head warden into a wing

}

else if (userChoice3 == "2")

{

Console.Clear();

newWarden.assignWardenWing();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

Console.Clear();

mainMenu.displayHeadWMenu();

}

//allows the user to view records of head warden in a certain wing

else if (userChoice3 == "3")

{

Console.Clear();

newWarden.showWardenWing();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

Console.Clear();

mainMenu.displayHeadWMenu();

}

//sends the user back to the main menu

else if (userChoice3 == "4")

{

Console.Clear();

mainMenu.displayMenu();

loop2++; //ends the loop sending the user back to the main menu

}

//allows the user to input another option if they input an invalid option

else

{

Console.WriteLine("Invalid Option. Please input one of the options.");

Thread.Sleep(1500);

}

}

}

//displays the warden area sub menu

else if (userChoice == "6")

{

while (loop2 == 0)

{

mainMenu.displayWardenMenu();

string userChoice4 = "";

userChoice4 = Console.ReadLine();//allows the user to choose another option from the sub menu

//allows the user to add a prisoner

if (userChoice4 == "1")

{

Console.Clear();

newPrisoner.setPrisonerID();

newPrisoner.setfirstName();

newPrisoner.setsurName();

newPrisoner.setGender();

newPrisoner.setDOB();

Console.WriteLine("");

newPrisoner.setPrisonerConviction();

newPrisoner.addPrisonertoList();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayWardenMenu();

}

//displays the prisoner details upon user input

else if (userChoice4 == "2")

{

Console.Clear();

newPrisoner.showPrisonerDetails();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayWardenMenu();

}

//allows the user to assign the prisoner into a wing

else if (userChoice4 == "3")

{

Console.Clear();

newPrisoner.assignPrisonerWing();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayWardenMenu();

}

//allows the user to view records of prisoners in a certain wing

else if (userChoice4 == "4")

{

Console.Clear();

newPrisoner.showPrisonerWing();

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayWardenMenu();

}

//sends the user back to the main menu

else if (userChoice4 == "5")

{

Console.Clear();

loop2++; //ends the loop sending the user back to the main menu

mainMenu.displayMenu();

}

else //allows the user to input another option if they input an invalid option

{

Console.WriteLine("Invalid Option. Please input one of the options.");

Thread.Sleep(1500);

}

}

}

//displays the on screen help of the program

else if (userChoice == "7")

{

Console.Clear();

Console.WriteLine("\*\*\*\*\*\*\*\*\*HELP PAGE\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine(@"

How to use the program:

1.) Each option has a number assigned to it.

2.) Input the option that is wanted.

3.) Follow what the program tells you to do.

4.) If there's still any problems contact me:

Email: test@gmail.com

Phone: 07123456789");

Console.WriteLine("\n \n Press the 'Enter' key to continue...");

Console.ReadLine();

mainMenu.displayMenu();

}

//exits the program

else if (userChoice == "8")

{

Environment.Exit(0);

}

else

{

Console.WriteLine("Invalid Option. Please input one of the options.");

}

}

}

}

}

}

# Prison System Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Number** | **Description** | **Test Data** | **Expected Outcome** | **Actual Outcome** | **Action / Notes** |
| 1 | Main menu – appointing options | Enter ‘1’ as a user input | When I input ‘1’ to the system, it should send me to an area where I can appoint a governor. | When I input ‘1’ it sent me to an area where I could appoint a governor. | This test is for the first three options (1,2 & 3) where the user is appointing the personal details (governor, head wardens and wardens) |
| 2 | Main menu – area options | Enter ‘4’ as a user input | When I input ‘4’ to the system, it should send me the governor area – in the governor area there will be another menu that allows the user to select options (similar to the main menu (1-5)). | Results came out as expected. | This test is for the next three options (4,5 & 6) where the user can select which area they want to go to (governor’s area, head warden’s area and warden’s area) |
| 3 | Main menu – help options | Enter ‘7’ as a user input | When I input ‘7’, the system should send me to the help page. | When I inputted ‘7’ it sent me to the help page telling me how to use the program. |  |
| 4 | Main menu – exit option | Enter ‘8’ as user input | When I input ‘8’ it should exit the program. | When I inputted ‘8’ it closed the program. |  |
| 5 | Main menu – wrong options. | Enter ‘test’ as a user input | When I input ‘test’ as a user input, the system should tell me that it is an incorrect option and allow me to input another option again. | Result came out as expected. |  |
| 6 | Appointing details e.g. governor, head wardens and wardens. | Key input ‘1’ | Upon the my input, the system should generate an a unique ID for the person and it should ask the user for the person’s details in this case the governor’s as the user input is ‘1’. | Results were as expected – a unique ID was generated and I was asked to input the user’s details. | This test also applies to the other two options (appointing head wardens and wardens) |
| 7 | Appointing details – inputting the person’s detail | Input random governor details. (First name, surname, gender etc.) | When I input the governor details, it should store the details somewhere (in an array) so it can be called whenever the user wants to view the details. | The results came out as expected and the data inputted was stored in an array. | This test also applies to the other two options in the main menu (appointing head wardens and wardens) |
| 8 | Governor area (sub menu) –valid option | Input ‘5’ as a user input | Upon inputting ‘5’ as a user input, it should send me back to the main menu. | When I inputted 5, it sent me back to the main menu. |  |
| 9 | Governor area (sub menu) – invalid option | Input ‘qwe’ as a user input | When I input the invalid option, it should tell me that it is an invalid option and it should let me input another option. | When I inputted the incorrect option, it told me that it was invalid and allowed me to input another option. |  |
| 10 | Governor area (sub menu) – viewing governor details | Input option ‘1’ | When I input the option ‘1’ it should display the details of the governor that has been inputted.  If there are no governor’s inputted, then it should tell the user that there are no governors available. | Results came out as expected – when there were governor records inputted, it displayed all of them, when there wasn’t any, it told me there wasn’t any. |  |
| 11 | Governor area (sub menu) – viewing head warden details | Input option ‘2’ | When I input the option ‘2’ it should display the details of all the head wardens that has been inputted | Results came out as expected – when there were head warden records inputted, it displayed all of them, when there wasn’t any, it told me there wasn’t any. |  |
| 12 | Governor area (sub menu) – assigning head warden to wing | Input option ‘3’ | When I input ‘3’ it should allow me to assign a head warden to a wing – in here it should ask me for a head warden ID | When I inputted ‘3’, it asked me for the head warden ID and allowed me to assign the record to a wing. |  |
| 13 | Governor area (sub menu) – valid ID | Input the head warden ID ‘H0’ | Upon my input, it should display the head warden’s first name and surname and ask me to assign the person to a wing. | Result came out as expected. |  |
| 14 | Governor area (sub menu) – invalid ID | Input the head warden ID ‘zxc123  ’ | Upon my input, it should tell me that it is an invalid ID and it should let me continue using the program letting me input another ID. | Result came out as expected telling me that it was an invalid ID. However, I changed it so it send the user back to the sub menu instead of letting them input another ID. |  |
| 15 | Governor area (sub menu) – valid Wing | Input ‘A’ as a wing option | Upon inputting ‘A’ as the option, the program should tell me that this record has been assigned to the wing. | When I inputted the valid wing, it told me that the record has been assigned to that wing. |  |
| 16 | Governor area (sub menu) – invalid Wing | Input ‘123zxc’ as a wing option | Upon inputting ‘123zxc’ the program should tell me that the wing doesn’t exist and it should ask me to input another option. | When I inputted the invalid wing, it told me that the wing doesn’t exist and asked me for another user input. |  |
| 17 | Governor area (sub menu) Viewing head wardens in a wing | Input ‘5’ as an option | When I input the option ‘5’ it send to a page where I can view head wardens in a certain wing, in here it should ask me what wing I want to view. | Result came out as expected. |  |
| 18 | Governor area (sub menu) – viewing head warden in a wing (valid input) | Input ‘A’ as a wing area option | Upon inputting the correct option, it should tell the user that it is that wing and display all the records in that wing. | Result came out as expected, it only showed the records assigned to that wing. |  |
| 19 | Governor area (sub menu) – viewing head warden in a wing (invalid input) | Input ‘asd’ as a wing area option | Upon inputting the incorrect option, it should tell the user that it is an invalid wing and allow the user to input another option. | When I inputted the incorrect option, it told me it was an invalid wing and allowed me to input another option. |  |
| 20 | Head warden’s area (sub menu) –valid option | Input ‘4’ as a user input | Upon inputting ‘4’ as a user input, it should send me back to the main menu. | Results came out as expected |  |
| 21 | Head warden’s area (sub menu) – invalid option | Input ‘asd’ as a user input | When I input the invalid option, it should tell me that it is an invalid option and it should let me input another option. | Results were as expected. |  |
| 22 | Head warden’s area (sub menu) – viewing all warden details | Input option ‘1’ | When I input the option ‘1’ it should display the details of all the warden that has been inputted.  If there are no warden’s inputted, then it should tell the user that there are no wardens available. | Results came out as expected – when there were warden records inputted, it displayed all of them, when there wasn’t any, it told me there wasn’t any. |  |
| 23 | Head warden’s area (sub menu) – assigning wardens to wing | Input option ‘2’ | When I input ‘2’ it should allow me to assign a warden to a wing – in here it should ask me for a warden ID | When I inputted ‘2’, it asked me for the head warden ID and allowed me to assign the record to a wing. |  |
| 24 | Head warden’s area (sub menu) – valid ID | Input the head warden ID ‘W0’ | Upon my correct input, it should display the warden’s first name and surname and ask me to assign the person to a wing. | Result came out as expected. |  |
| 25 | Head warden’s area (sub menu) – invalid ID | Input the head warden ID ‘zxc123  ’ | Upon my incorrect input, it should tell me that it is an invalid ID and it should let me continue using the program letting me input another ID. | Result came out as expected telling me that it was an invalid ID. However, I changed it so it send the user back to the sub menu instead of letting them input another ID. |  |
| 26 | Head warden’s area (sub menu) – valid Wing | Input ‘A’ as a wing option | Upon inputting ‘A’ as the option, the program should tell me that this record has been assigned to the wing. | When I inputted the valid wing, it told me that the record has been assigned to that wing. |  |
| 27 | Head warden’s area (sub menu) – invalid Wing | Input ‘zxc123’ as a wing option | Upon inputting ‘123zxc’ the program should tell me that the wing doesn’t exist and it should ask me to input another option. | Result came out as expected – the program told me that the wing doesn’t exist when I inputted the invalid option and allowed me to put another input. |  |
| 28 | Head’s area (sub menu) Viewing wardens in a wing | Input ‘5’ as an option | When I input the option ‘5’ it send to a page where I can view warden in a certain wing, in here it should ask me what wing I want to view. | Result came out as expected – inputting ‘5’ sent me to an area where I can view wardens in a wing. |  |
| 29 | Head warden’s area (sub menu) – viewing warden in a wing (valid input) | Input ‘A’ as a wing area option | Upon inputting the correct option, it should tell the user that it is that wing and display all the records in that wing. | Result came out as expected |  |
| 30 | Head warden’s area (sub menu) – viewing warden in a wing (invalid input) | Input ‘asd’ as a wing area option | Upon inputting the invalid option, the program should tell the user that the wing doesn’t exist and it should ask for another input. | Results came out as expected |  |
| 31 | Warden’s area (sub menu) –valid option | Input ‘5’ as a user input | Upon inputting ‘5’ as a user input, it should send me back to the main menu. | When I inputted ‘5’ it sent me back to the main menu |  |
| 32 | Warden’s area (sub menu) – invalid option | Input ‘zxc12’ as a user input | When I input the invalid option, it should tell me that it is an invalid option and it should let me input another option. | Results came out as expected, the program told me that the input was invalid and I had to do another input, |  |
| 33 | Warden’s area (sub menu) – appoint a prisoner | Input option ‘1’ | When I input the option ‘1’, it should send me to an area where I can input the prisoner’s details. | Results were as expected – a unique ID was generated and I was asked to input the user’s details. |  |
| 34 | Warden’s area (sub menu) inputting prisoner details. | Input random prisoner details. (First name, surname, gender etc.) | When I input the prisoner details, it should store the details somewhere (in an array) so it can be called whenever the user wants to view the details. | The results came out as expected and the data inputted was stored in an array. |  |
| 35 | Warden’s area (sub menu) – viewing all prisoners | Input option ‘2’ | When I input the option ‘2’ it should display the details of all the prisoners that has been inputted.  If there are no warden’s inputted, then it should tell the user that there are no wardens available. | Results came out as expected – when there were prisoner records inputted, it displayed all of them, when there wasn’t any, it told me there wasn’t any. |  |
| 36 | Warden’s area (sub menu) – assigning a prisoner to wing | Input option ‘3’ | When I input ‘3’ it should allow me to assign a prisoner to a wing – in here it should ask me to input a prisoner ID | Results came out as expected. |  |
| 37 | Warden’s area (sub menu) – valid ID | Input the head warden ID ‘P0’ | Upon my correct input, it should display the prisoner’s first name and surname and ask me to assign the person to a wing. | When I inputted the correct option, it allowed me to assign a prisoner to a wing and display the names as expected. |  |
| 38 | Warden’s area (sub menu) – invalid ID | Input the head warden ID ‘zxc123  ’ | Upon my incorrect input, it should tell me that it is an invalid ID and it should let me continue using the program letting me input another ID. | Result came out as expected telling me that it was an invalid ID. However, I changed it so it send the user back to the sub menu instead of letting them input another ID. |  |
| 39 | Warden’s area (sub menu) Viewing prisoners in a wing | Input ‘4’ as an option | When I input the option ‘4’ it send to a page where I can view prisoners in a certain wing, in here it should ask me what wing I want to view. | Result came out as expected. |  |
| 40 | Warden’s area (sub menu) – valid Wing | Input ‘A’ as a wing option | Upon inputting ‘A’ as the option, the program should show all the records assigned to that wing. | When I inputted ‘A’ it showed all the records assigned to that wing. |  |
| 41 | Warden’s area (sub menu) – invalid Wing | Input ‘zxc123’ as a wing option | Upon inputting ‘123zxc’ the program should tell me that the wing doesn’t exist and it should ask me to input another option. | When I inputted an invalid option, it told me that the wing doesn’t exist and asked me to input another appropriate wing. |  |

# Test Results Analysis

Test 1 – Choosing and inputting the ‘appoint’ options

In this test, I expected that when I inputted an appoint option, in this case ‘1’ it would send me to an area where I can appoint a governor. Upon testing it, the actual outcome was what was expected, it allowed me to appoint a governor by inputting their personal details. Their first name, surname, gender, date of birth, address, contact number.

Test 2 – Choosing and inputting the ‘area’ options

The actual outcome of the second test was also what I expected the outcome to be. Upon debugging my program, when I chose an area option in this case ‘4’ which is the governor’s area, it sent me to that area allowing me to choose what the governor can do such as view the governor details, view head warden details, appoint a head warden to a wing, etc.

Test 3 – Choosing help option

Test three was displaying the help page once the option was chosen. As expected, the actual outcome was the same – upon choosing this option, the help page was shown to the user.

Task 4 – Choosing the exit option

The fourth test in my test plan was to allow the user to exit once they have chosen this option. As expected, the program closes and exits which what is wanted.

Test 5 – Inputting a wrong option.

Test 5 is a test that uses an abnormal data, I have used ‘test’ as an example input. I expected that when I input ‘test’ as a user input, the system should tell me that it is an incorrect option and allow me to input another option again. As expected the actual result was what was expected. The program told me that it as an incorrect option and it allowed me to input another option again.

Test 6 & 33 – Appointing details (governor, head warden, warden, and prisoner)

This analysis applies to both tests are they have the same expected results and similar test data – the user would need to input an option and once they have inputted the option it is expected that the program generates and unique ID and asks the user to input the personal details of that person. As expected, once the option is chosen the program generates a unique ID for each person, the unique ID is the first letter of their role e.g. P for prisoner and a number (P0). After the generation the user is then asked to input personal details.

Task 7 & 34 – Inputting the details (governor, head warden, warden and prisoner)

Just like the previous test, this analysis applies to both tests as they have the same expected results and similar test data – the user would need input the personal details and it should be stored somewhere else so that it can be viewed whenever the user wants it. As expected, once the user has inputted all of their personal details, all of the data that they have inputted gets store to a specific array. Each class have their own array therefore prisoner data gets stored in a prisoner array. One thing to note about this test is that every field of personal detail accepts any input as the data type is set to a string. This means that data validation is quite poor and anything can just be inputted and get stored into an array so that it can be easily called. However, one way that this can be improved is by ensuring that there is data validation and that each field has a proper data type set to it so it can only accept certain things. For example, the age field can be set into an integer so that it only accept numbers.

Test 8, 20 & 31 – Inputting a valid option in the submenu.

In these tests, I expected that when I input a valid option in each sub menu it will do what the option says and will not cause any errors. I have decided that for all these tests, I input the option to go back to the main menu. As expected, when I input the option that sends the user to the main menu, it sends them there where they can choose an option again.

Test 9, 21 & 33 – Inputting an invalid option in the submenu.

In these tests, I expected that when I input an invalid option in each sub menu, it will tell the user that it is an invalid option and it will ask the user to input another option. As expected, when I input the invalid option it tells the user that it is an invalid option and asks the user to input another option.

Test 10, 11, 22 & 35 – Viewing all people in a wing (governor, head wardens, wardens, and prisoners)

In these tests, the results were as expected. When I inputted the option to view all the people in a wing, it displayed all of them if there were records available. Moreover, when there weren’t any records available, the program told me that no person has been inputted.

Test 12 & 23 & 36 – Assigning a person to a wing (head warden and warden)

In these tests, I expected then upon inputting the option, it will send me to an area where I can assign the person to a wing. As expected, when I inputted the option, it sent me to the assigning page where I input their unique ID and assign them to a wing.

Test 13, 24 & 37 – Inputting a valid ID when assigning a wing (head warden, warden and prisoner)

I expected that in these tests, when I inputted the correct option, for example “P0”, it will display that person’s first name and surname and then it will move on to the next step of the assigning process and ask me to input which wing the user wants the person to be in. The results were as expected.

Test 14, 25 & 38 - Inputting an invalid ID when assigning a wing (head warden, warden and prisoner)

In these three test, when I input the invalid ID in these tests, I expected that the program will tell me that it is an invalid wing and it will ask me to input another ID, hopefully a valid one this time. When I tested the program, to begin with it came out as expected it tells me that the ID is invalid. However, I have changed it so that it sends the user back to the sub menu where they can choose another option again instead of asking the user to input another ID.

Test 15, 26 & 39 – Viewing records in a wing (head wardens, wardens and prisoners)

All of these tests had actual outcomes which were the same as the expected outcome. Upon inputting the option, allowed me to view records in a wing. This was done by the program asking me what wing I want to view and I input which wing I wanted. After my input, the program then displays all the records in that wing.

Test 16, 27 & 40 – Inputting a valid wing (head wardens, wardens and prisoners)

In these tests, the actual outcome was similar to the expected outcome. When I inputted a correct wing ‘A’, it displayed all the records that is in that wing. Although to begin with, it displayed an empty record at the start but I have now amended that so no empty records are display and stored.

Test 17, 28 & 41 – Inputting an invalid wing (head wardens, wardens and prisoners)

When I first programmed these tests, it used to display several invalid lines and it used to be how many records there were stored. I have now changed it so that the actual outcome is the same as the expected outcome. Upon inputting the incorrect wing, it tells me to input an appropriate wing and allows me to input another wing again

# Evaluation

Implementing the design work was a simple task but to begin with gave me troubles. To begin with, when I was creating the class responsibility cards I have included both the responsibilities of the class within and out of the system. For example, when I created the warden class, I mentioned how it could display the prisoner details but I also mentioned how they can physically carry out cell inspection – this wasn’t how the CRC was supposed to be as the class responsibility cards was supposed to show the responsibilities of the class within the system, not outside of it. Because of this mistake that I have done, it made the creation of the of the class diagrams more tedious and challenging for me as I had to create properties and methods in a class which did not entirely make sense for the system. However, I have amended my designs to try and fit it to what they were supposed to do; I have left the outside responsibilities in my CRC but I have highlighted them and changed my class diagram to suit the amended version making the program a lot easier to do. The properties and methods in my class diagram only includes the responsibilities in the system now. This means that when I moved on to program my solution, it was a lot more manageable and easier to do.

Programming the solution had its ups and downs. To begin with, I had to create the classes to fit the solution. This was quite an easy task but a repetitive one, I created a person class in which the ‘person’ classes such as the governor, head warden, warden and prisoners. Although it was quite easy, it still took me some time as I had to program all the methods that I wanted in each class such as retrieving their name, age, etc. This was an easy task to do as all it required was outputting what was wanted and then asking for a user input. However, I had a huge problem when I created these, to begin with I had different data types for specific details, for example with the age I had it as an integer because it is a number. However, it proved to be quite difficult storing different types into an array therefore I changed all of the data types back into a string. Although this was a good solution, it wasn’t really the best solution as it meant that there is no proper data validation in within my program. If I were to improve my program, this is one thing that I could do – ensuring that there is proper data validation when inputting details as opposed to every single detail accepting a string. Apart from this problem programming the rest of the classes were quite easy this was because most of the classes inherited from the person class so the other classes did not need a lot of methods.

Once I created the person class, I moved on to create the ‘person’ classes which include the governor, head warden, warden and prisoner classes. All of these classes are quite similar, all of them needed me to inherit the person class which was a very easy thing to do. After inheriting the person class, this mean that I could use all the methods in the person class as well as in this one. I then programmed each person class to have their unique identifier. I did this by having a variable that was incrementing every time a new record is added which wasn’t hard to do.

Just when I thought I was cruising through the creation of these personal classes, I moved on to storing the details into list. This proved to be a daunting task and I was not able to do it as it was quite difficult to create a multi-dimensional lists. Therefore as a solution to this, I instead created a multi-dimensional array where I could store the details that user has inputted. Each person class have their own array to store in and doing this was very time consuming. In the end, I was able to store the details and output it to the user which is what was wanted by the client. Although it is a good solution, using this solution has a massive problem, it only stores the user’s input when the program is running – therefore when the program is closed all the records that are stored in each array are deleted. In the end, I decided to stick to the solution because of time constraints. However, one thing that could drastically improve the solution is by storing these inputted data into an external file. It could be using an Excel file or a Word document, in my opinion it doesn’t matter as long as it gets save externally and it can be accessed whenever the user wants it as opposed to the current solution that I currently have which wipes all the data once the program is closed.

When I was creating the classes and following my designs, I also noticed that I did not need the ‘Wing’ class and the ‘Cell’ class – this is because when I was programming the system, I found that the classes that I have mentioned were not required to create the solution that was needed by the client. Therefore I removed these classes and amended my program and instead added a different class which is the main menu class.

After creating the classes that I had in my designs, I then started programming the main menu classes of the prison system, it was quite an easy thing to do. It mostly consisted of my creating lines of codes that outputs to the user to show them which button to press to do a specific thing; the main menu was done using the same line of code. The main menu was not included in my screen designs but I have added it as a new class for the system – this is because having a main menu allows the user to easily navigate the system.

Once I have created all the classes, I then moved on to the main part of the solution which is using the classes and methods and putting them together to create the solution. This was quite an easy task though it was a repetitive one. All I had to do was create an object from the class and then call the methods into the object for the program to do something. This part of the programming stage was quite simple and did not give me a lot of trouble but it was quite time consuming – this was because I had to use a lot of conditional statements (Ifs and else ifs) because it is a command line program therefore I had to provide the user several options to navigate through and use it. But since, I am quite confident with looping and conditional statements I found it quite easy to the main part of the program.

During the creation of the application, I also did some testing – I met several errors when creating the program but they were mostly logical errors which I was able to fix by trying a different solution to the problem. Some of the errors that I encountered were the loops that I have done not properly working but I have now fixed them and loops in my program are what I expected.

Overall, the creation of the prison system solution went well but it was quite a repetitive task and there were some problems that I have encountered. The actual outcome of the solution followed some aspects that I have mentioned in my design work but there I have also missed several aspects that I had in my design. One of the things that I have missed out when I created the program were some of the classes that I have mentioned in the design work such as the Wing and Cell class. However, even without these classes, it still follows the requirement specification that has been created which is a strength of the application. It allows users to input and appoint details of people such as governor, wardens and prisoners to a basic level, being able to store these details and display them so long as the program is running. Moreover, I have also added a main menu class to make navigation for the user a lot easier which is another upside to the application.

However, my application also has several downsides – for one, although it follows the requirement specification and allows the user to appoint and view people details, this is only to the basic level and it only works when the program is running. . When the program is closed, you would have to input all the details of the person to be able to view them again otherwise no record would be shown. This is a massive weakness to the application and one of the aspects of the program that needs to be done differently. As mentioned above there are still various improvements that could have been made because the application has a lot of weaknesses. For one, the way I store my data, at the moment as mentioned above, I am storing it to an array – doing it this way does what is required, it stores it and allows the program to call it whenever. However this only works when the program is constantly running; when the program is closed every data in the array would disappear and the user would have to re-input the data to view it again. This is not a very user-friendly application. Another weakness that the application currently have is that there are no proper data validation.

Thus there are several future improvements that needs to be done to the application to make it more useable and functional. To begin with, as a solution to the array problem, it would be better to the details of what the user has inputted to an external file such as Excel so that in can be accessed whenever when the user opens the application, as opposed to the current application that I currently have, where the user would need to input every single record if the application is closed. Furthermore, variables would need to be properly validated, for example date would have a proper data type of ‘DateTime’ as opposed to having a string that it currently has. By having proper data types, there will be proper consistency in the application it will be more accurate than the current one. In conclusion, I am still quite happy with the solution and considering it was my first time programming an object oriented program I think I did quite well.