

15.2

By Adam Chu



Contents

[Plan 1](#_Toc11789987)

[UML 3](#_Toc11789988)

[Test 3](#_Toc11789989)

[Screenshots 5](#_Toc11789990)

[M2 6](#_Toc11789991)

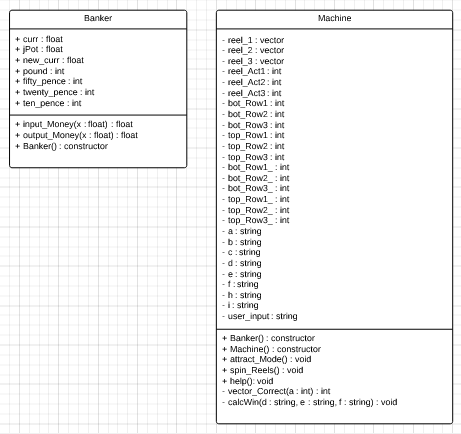
[D1 6](#_Toc11789992)

[D2 7](#_Toc11789993)

# Plan

|  |  |  |  |
| --- | --- | --- | --- |
|  | Arm | | Activates the Wheels if money is >= 10 and money / 10 = int |
| Coin Input | | Allows the user to add money to the machine |
| Display – For Current Winnings & Inputted Money | | Showing Current Monetary value stored inside the program |
| Output Port | | Where Coins are Released for Collection |
| Nudge & Hold buttons | | Allows your program to run methods Nudge or Hold |
| Jackpot number | | Total value of the Current Jackpot |
| 3 Reels | | They Rotate a Certain Number of times based on the program |
| Board with winning values | | Shows you the ways to win money on the fruit machine |
| Eject Button for money | | Releases the currently stored money value. |
| Demo Mode | | Shows pre-programmed effects like a One Armed Peacock, strutting its feathers asking for more money |
| **Machine Class:** | |  | |
| * Spin\_Reels- Method | | Activates the OAB | |
| * Attract Mode | | This initially makes the Shell before the Spin Reel is activated | |
| * Vector\_Correct | | Stops the top Row and Bottom Row from Over extending the Vectors | |
| * calcWin | | Checks if you have won anything and inplements the jPot value  And runs On screen Help | |
| * nudge\_No – Property | | Holds the total number of nudges that can be used | |
| * Reel1, Reel2, Reel3 | | Vectors which hold the words for each Reel | |
| * A,B,C,D,E,F,G,H,I,J | | A-I are the Values printed for each reel and J is what is used to decide which option after the On Screen Help. | |
| * Reel\_Act1 (-3) * Top\_Row (-3) * Bot\_Row (-3) * Top\_Row\_ (-3) * Bot\_Row\_ (-3) | | All are Ints which are between 0-6, Reel\_Act is created with RAND % 7  Bottom Row is Reel Act +1  Top Row is Reel Act -1  Top\_Row\_ is Top\_Row put through Vector\_Correct | |
| **Banker Class:** | |  | |
| * input\_Money - Method | | This is the method that changes the money input to the machine into a set currency value | |
| * output\_Money - Method | | This is the method that breaks down the money into a series of change | |
| * curr | | Money inside the OAB’s memory | |
| * jPot | | JackPot for the 7 7 7 roll, this will start at £5.00 | |

# UML



The idea behind my UML is that the Money Class will interact with the machine class.

I have used 3 arrays so that each reel is different and helps to stop the pseudo-randomness of using Rand(). I have decided to use a a-i system similarly to a t9 phone. This will use 3 rows so that it feels more like an actual OAB like the one shown above.

# Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | What Am I Testing: | Result | | Screenshot |
| 1 | Does the Help function start when the program starts | Yes | Yes | 4 |
| 2 | What happens when you input negative money into the system (i.e. 10p’s = -1) | Yes | The code will not change the money value and tell you this | 5 |
| 3 | Does the Code realise you have runout of money and start the Input function? | Yes | Yes the If statement calls the input money method and the While(true) that encapsulates this  Restarts the OAB when the currency is >=.10 | 1 |
| 4 | Is Spin Reels responsible for assigning variables Reel\_Act? | Yes | Using a seeded Pseudo-Random Sequence, this allows the user to use the MOD function (%) and get the Remainder between that and a number 7.  This takes a pseudo-random number and MOD 7 which gives you 0-6 | Code Line 52-55 |
| 5 | Does calc\_Win, increment the jPot Value and runs the On Screen Help activate | Yes | This will increment the jPot by 0.10 and is shown when the next set is shown  After the system is shown in shell, it shows the on screen help | 2 |
| 6 | Does Vector\_Correct return values based on the size of top and bottom row | Yes | It sets 7 to 0 and it changes -1 to 6  And returns the new values with an else. if which with return value between 0-6 | Line 145 - 155 |
| 7 | Does the input money prompt ask for all the common coins sequentially | Yes | It asks the user 10p’s 20p’s 50p’s £1’s  To make it more user friendly it will ask them as individual questions instead of the Cin >> 10p >> 20p >> etc. | 1 |
| 8 | Does the output money correctly calculate how much money is in the machine and output 10p’s accordingly | Yes | It will say that it has x \* 10p’s | 3 |

## Screenshots

|  |  |
| --- | --- |
|  | This the screenshot showing the Input Money In which it will ask the user to quantify which coins are being inputted into itself |
|  | This is the Cout that is run when the user uses ‘output\_Money’ |
|  | This is the Game Screen at the end of one cycle, in which the on-screen help is activated  It also shows the 3 rows working and line up properly |
|  | When the program is created the system calls help instead of activating immediately activating Input Money, also the jPot Value is set to £5 at the start of the system |
|  | This shows the systems response to when the user inputs negative money it will show an error statement and not change the users currency value. |
|  | This is a screenshot show the user when you win. You get a win statement and showing the amount you win as well as running the on screen help method. |

## 

## M2

During the first stage of Abstraction I had 4 classes that I wanted to implement these were: Button, Banker, Reel & Machine. These would have different methods based on each class:

* I wanted to have a separate Reels class defining each reel and creating a function called spin\_Reels(),
* In the Machine Class I wanted what created the UI and decided if you won or not,
* In the Button Class I was trying to create methods for Hold, Nudge, autoplay as well as help as this class was meant to be the class which is responsible for player input and output
* The last class was meant to be Banker which is responsible for money like it is now which would input money, export money as well as creating and controlling the JackPot Values.

In the during the second stage of abstraction I had decided to merge the Reel, Machine Classes & Button due to Reel having 1 function and 3 properties. Then I decided that I did not need to have a button class which would be responsible for Hold and Nudge as well as the help function, I had decided that I did not need to have a Hold and Nudge inside this project.

The £ symbol is not inside the ASCII character set. And I did not have time to figure out how to change it to UTF-8. This means when dealing with Pounds and Pence I had to not have a denomination, this makes the game screen less professional. I could have changed it to use USD which is supported by the ASCII Character set but doing this would have forced me to recreate the input money system which uses the GDP coins like £, 10p, 20p etc.

# D1

To convert my program to be a graphical application the first thing I would have to do is to remove the iostream elements as well as remove the system elements. Then I would have to find libraries that allow for graphical things to be created and edited as sprites like using SFML (Simple Fast Multimedia Library). This would require me to learn how to use a library to show and change custom sprites for each. But unlike using Procedural programming you have the benefits of Encapsulation and Inheritance as you could create class called sprite and use inheritance to create each item as an inherited class. And have encapsulation to make each instance separate, with a sprite for each item in a vector. But otherwise using OOP I wouldn’t have to recreate the rest of the code like the randomisation I could just use the vectors to decide which sprite is shown (Reduce all of the other Alpha Values).

Also I could use Objects to create ‘sprites’ like I used for 22.2 as it would allow me to change and edit properties of an Instance, after it is drawn to the shell unlike using Procedural programming where once an ‘graph’ is drawn to the screen it becomes un-editable.

Then you have the display which I quite like seeing as it took me a fair amount of time to create as I had to find sub-libaries of the iostream which Cin and Cout are derived from which would allow to force 2 decimal places when Couting a currency value. This would be a lot nicer if the £ was included with ASCII though and due to me not knowing the USD (Coins when creating this and not wanting to deal with 5p’s) made the shell feel unfinished and unprofessional. I feel like its easier to reuse functions with a similar purpose for the Input and Output Money functions I created one of them and change the latter to fulfil its purpose as all of the variables I am using were already initialized and prepared.

# D2

For the two classes that I have created they have the names of machine and banker. The name Banker is quite descript so it is very obvious what it is responsible for which is money, the other class name machine is fine as a class name as in context of the project it makes sense but I could have called it Fruitmachine to make it obvious.

Most of my variable names are fairly descript with reel\_Act1, reel\_Act2 & reel\_Act3 being the generated numbers and having top\_Row1 & bot\_Row1 following a similar pattern in naming, after the top\_Row1 is entered into vector\_Correct it is reassigned to top\_Row1\_. This reassignment is not necessary but this was due to my previous plan of having it take the old position and add the new modifier to create a new number, and I kept the \_ variables.

I do have some bad variable names which are the strings a,b,c,d,e,f,g,h,i. These were created so that when I was editing the UI it was easier to make the text line up and the border match up as well this is due to the shortening of: reel\_1[top\_Row1\_] to a. In hindsight this makes my code harder to understand and worse in terms of upgradability as if someone else was to edit this code they would need a data dictionary whereas if the variable names were completely descript it could be self-explanatory.

I think that the nature of having my methods and properties being split over multiple classes would allow me to add more functions for example: adding another function for Banker which could be save money & jPot to a local file, this method could use the variables inside the class Banker like curr & jPot, and be included with an instance of banker. I do think that the use of vectors would allow me to assign them with different items if I were to create a different OAB with this as a framework, it could have a different sized reels and have different items on it by changing the MOD operation and the calcWin method.

I am quite pleased with the system which allows the user to input money in the form of coins even-though that in its current state it’s too constrictive, its slows down the players experience as it could have been a normal cin with the user inputting a float. I like this method as it asks the user to quantify which coins are being inputted to the system but, after getting feedback from various people they have asked that the questions during the input money be less non-descript and I have split it up so that each coin is its own input.

Before it was cin >> 10ps, 20ps, 50ps, £1 and after changing it to:

How many 10ps are you Inputting? Answer

How many 20ps are you Inputting? Answer

Etc.

I am also quite pleased with the method which allows for each row to have the 1 above and 1 below itself, with the modulus operation it would stop the vectors from overextending. As vectors start at 0 and in my case end at 6, the use of Mod 7 takes a number / 7 and gives you the remainder of 0 - 6. The method vector\_Correct takes the row checks if it > 6 and < 0 and gives you the inverse if that is true, this is required as my top row is the number post Mod 7 and then - 1 and the reverse is true for the bottom row.

I am quite pleased with the game-screen as it looks professional with a created border which does not move or change position, even when the text is displayed for each row or when the money goes from an integer to a number with a decimal place. This is due to me using a subsidiary library to iostream called iomanip (io manipulation) this allows me to force two decimal places. To help with the professionalism of the UI I have added underlining for the JackPot and Money which are displayed on the screen and added blank lines inside of the border which separate the reels from the text displayed above.