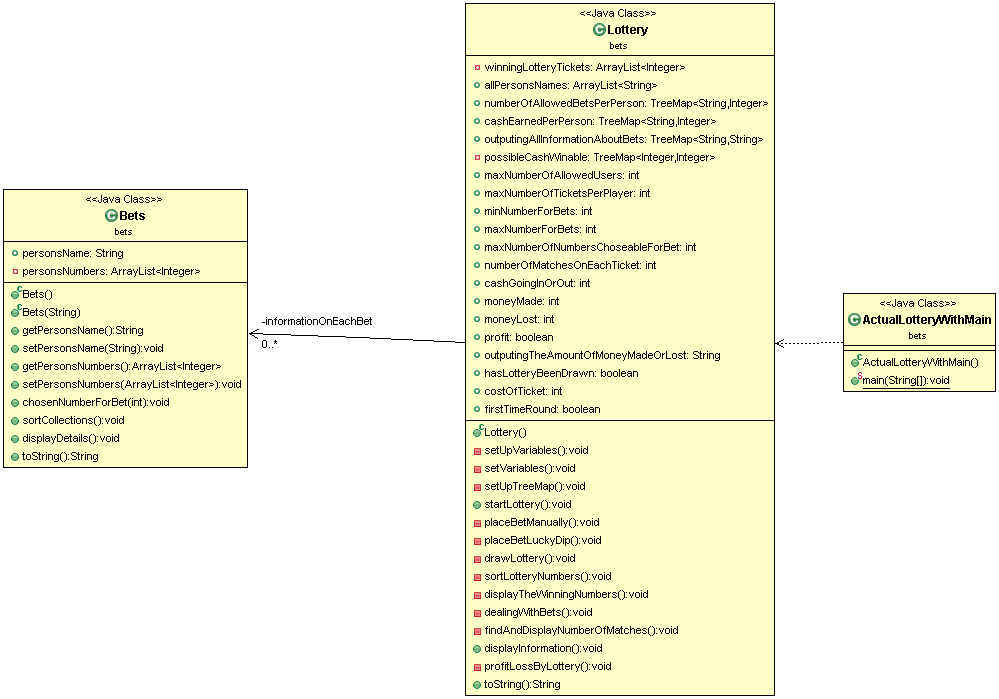
Class Diagram



Classes structure and explanation

My main class is called ActualLotteryWithMain so that it was distinguishable from the lottery class. This class has a simple task of setting up a new lottery and starting it. Once it has finished the whole inside of the lottery class it will output the final figures of how much money was made, lost and if it made profit or lost money. My Bets class has the simple task of keeping track of the person's name and the numbers that they have chosen, it is fairly short as it has little tasks but is called from the Lottery Class for each bet that is created by the user, it also displays the bets that a person has placed once they are finished. My Lottery Class is the main focus of the programming and contains most of the code. This was done to keep the program simpler as i often found myself calling to this in the Lottery class that contained main, this allowed for the program to be a lot shorter and be less likely to have errors. The main task of this Class is to allow the user to chose between three options, they can enter numbers for a bet manually, do a lucky dip or draw the lottery. If the user asks to manually enter the lottery it will allow them to choose 6 numbers between 1 and 50, if they chose lucky dip they get 6 numbers randomly, and if they chose to draw the lottery early it will work forward through the program and find out what the winning numbers are. If the there have been 20 bets it will automatically draw the lottery. Once the lottery has been drawn it will randomly generate the winning numbers making sure there are no duplicates. It will sort them going from lowest to highest as this is easier to read to the human eye than it is having it be random numbers. It will then display the winning numbers. Then the program will compare each person's bet numbers to the winning numbers and track the amount of matches that there are and display to the person how much they have won over all of their tickets if they have more than one. Once this is finished the program will calculate the amount of money flow(in, out,profit/loss), it will then end the program and allow main to continue and display the money flow.

Duplicate numbers

The simplest way I could find of stopping a user entering duplicate numbers was using saving the numbers in an arraylist and using a .contain. This allows for entering the number that the user enters into to be compared to all of the numbers that they have previously entered. The example from my code is

**if**(nextBet.getPersonsNumbers().contains(numberchosen)== **false**){

checkingForDuplicates = **false**;

nextBet.chosenNumberForBet(numberchosen);

} **else**{

checkingForDuplicates = **true**;

}

As this shows it using this system it is easy to compare all of the numbers that have been generated or entered to all of the previous ones that have been entered by the users. if it equals false it will add the number to the array list in the other object, however if it equals true it will ask the user it will ask the user to re-enter the there number and continue checking in the next section(inside a while loop) of code to ensure that the number entered is not a duplicate, the same is done with the lucky dip and drawing the lottery however the user is not prompted off this.

Extensibility

I believe that my program is very extensible as all of the required changes to the program are saved as variables at the beginning of the program. If you wished to change the number of maximum allowed users it, range for bets, amount of numbers choosable per bet or cost of ticket it is possible to change with a simple change to a single number under the setvariables. It is also change the amount of winnable cash and the amount of tickets needed to get that cash amount by following the structure shown in setuptreemap(). The program is also designed around being simple which is why i chose to put the majority of the code inside the lottery class as i found myself calling it almost all the time from the class i had that included main, this allows for the program to be much simpler and easy to read meaning anyone should be able to read the code and understand what it is doing with basic java knowledge.

Advanced Features

I was able to complete both of the advanced features by allowing only three bets to be placed by one person(assuming name is unique) and would output at the end the individual tickets, at the very end display how many matches they got on multiple tickets and how much they earned total.

**Limit number of bets by one person**: i was able to only allow one person to place a maximum of three bets with the use of a treemap. if it was the first time around the loop it would place the person's name as the key inside the treemap with 1 being the value entered. every time after the first loop it would check the if the person's name was contained as a key in the treemap. if it was not found it would just add it to the tree map, if it was found it would get the value and check how many bets had been placed from this. if it found a value that was larger than the maximum amount of tickets it would force the player to chose another name. if it was still less than three it would just add the name and the amount of bets already placed and add another to it to the treemap. this allows for constant checking and will never allow a user to place more bets that they should be allowed. however it does not check for capitalisation therefor as a change to the future versions this should be changed so that all characters are lowercase and therefor the names will be truly unique.

**Display winnings by person:** i was able to show the winnings of each user if they had more than one or more tickets with the use of treemaps. this was stored during the creation of the bets and if they had a bet placed it was ignored and if they didn't it was added to allPersonsNames(which is an arraylist). during the comparing stage and finding out how many matching tickets the user had on each bet it it would call a TreeMaps. this allowed me to track the ticket result in a treemap called outputingAllInformationAboutBets, this stores all of the results of the tickets. another treemap called cashEarnedPerPerson tracks the total amount that a person has won. both of these allow me to track all of the information that i will use to display. once i get to the display part i used a for loop that would loop around for the size of the amount of names(found from allPeronsNames), this would allow it to get the person's name from the arraylist using the index(loop count) and then use the treemaps to get the information for each of the names and display it.