I chose to write a Banking application to demonstrate the utilization of different classes and inheritance as well as practicing switch cases. This program computes a bank account for a user to manipulate. The first type of input the user employs is the options they can do with their bank account. The second type of input is the amount of money, which outputs a different value of balance in the user’s bank account depending on the option they chose with the first input. The program has two separate classes, the BankAccount (line 30) and Menu (line 86) which extends from BankAccount. The BankAccount class sets the variables for the Name and ID of the bank account, as well as the balance and previous transaction. The functions contained within BankAccount are crucial to any sort of banking application for the user to utilize, such as: depositing to their overall balance, withdrawing from their overall balance, and displaying the previous transaction made. The try/catch methods are used on the depositing (line 38) and withdrawal (line 54) function; this is to ensure the user is not inputting a negative number or a random character into the program when depositing/withdrawing an amount of money from the balance. If the user does so, the illegal argument exception is thrown and the user can try again. The get previous transaction function is used to display to the user what transaction they just did to their bank account previously. The Menu class which extends from BankAccount first uses a constructor for the name and ID of the user’s bank account. From there, the showMenu function is utilized. The showMenu function in the Menu class first displays a welcome message as well as options the user can choose from to act on their bank account. Their options being, A: check balance, B: deposit, C: withdraw, D: see previous transaction, and E: exit. A do while loop (line 109) is then used to continuously prompt the user to enter an option, which is then interpreted into different switch cases. The first case, Case A, just outputs the user’s current balance. Case B asks the user to enter an amount to deposit, which is scanned and then used as the variable for which the deposit function from BankAccount class is called. Case C is similar, the only difference being asking the user to enter an amount to withdraw, which is also scanned and then used as the variable for which the withdraw function from BankAccount class is called. Case D is used to output the user’s previous transaction, calling from the getPreviousTranscation function from BankAccount class, displaying how much they deposited or withdrew from their overall balance. The final case, case E, acts as an exit and if chosen terminates the do/while loop. The user is given the message “thanks for banking with us!” to inform that they have successfully exited the bank menu and ended the program. The main class first creates an object for Menu, and provides the name and ID for those variables. The showMenu function is then called by the object that begins the whole banking application, and once exited and terminated a recursive function factorial (line 12) is used in the overall Banking class to show the user a factorial number of a specific integer. Overall, I’ve learned how useful it can be to create different classes for organizational purposes, and how inheritance can be utilized to extend the functions between said classes. I hoped to use recursion in a more meaningful way, however it was difficult to implement using the constructor variables like name and ID number within it. The factorial function did educate me on how effective recursion can be on repetitive tasks with slight modification.

Public class Banking

* Static int factorial()
* Public static void main()
* Int num
* Int result

Class Menu extends BankAccount

* Menu() constructor
* Void showMenu()
* Void deposit()
* Void withdraw()
* Void getPreviousTranscation()
* Int balance
* Int previousTransaction
* String name
* String ID
* Char option
* String cname
* String cid

Class BankAccount

* Void deposit()
* Void withdraw()
* Void getPreviousTranscation()
* Int balance
* Int previousTransaction
* String name
* String ID

Sources:

Darshan Mehta: Illegal Argument exception for try/catch function catching negative numbers a user inserts-

<https://stackoverflow.com/questions/35806637/java-include-negative-values-in-catch-block-for-double-parsedouble>

w3schools: Java recursion example-

<https://www.w3schools.com/java/java_recursion.asp>