Montgomery College
CMSC 203
Assignment 6 Design

1) Write the pseudo code for the methods of addTicket() and getMoviePassTickets() based on the Assignment 6 Description given to you. Refer to the **Pseudocode Guideline** on how to write Pseudocode.

2) Create the UML diagram for classes in this assignment.

Pseudocode Guideline

Pseudocode is code written for human understanding not a compiler. You can think of pseudocode as "English code," code that can be understood by anyone (not just a computer scientist). Pseudocode is not language specific, which means that given a block of pseudocode, you could convert it to Java, Python, C++, or whatever language you so desire.

Pseudocode will be important to your future in Computer Science. Typically pseudocode is used to write a high-level outline of an algorithm.

As you may already know, an algorithm is a series of steps that a program takes to complete a specific task. The algorithms can get very complicated without a detailed plan, so writing pseudocode before actually coding will be very beneficial.

How to Write Pseudocode

There are no concrete rules that dictate how to write pseudocode, however, there are commonly accepted standards. A reader should be able to follow the pseudocode and hand-simulate (run through the code using paper and pencil) what is going to happen at each step. After writing pseudocode, you should be able to easily convert your pseudocode into any programming language you like.

We use indentation to delineate blocks of code, so it is clear which lines are inside of which method (function), loop, etc. Indentation is crucial to writing pseudocode. Java may not care if you don't indent inside your **if** statements, but a human reader would be completely lost without indentation cues.

Remember: Human comprehension is the whole point of pseudocode. So, what does pseudocode look like?

Finding the Fibonacci numbers till n:

Pseudocode	Real Code in Java

Declare an integer variable called n
Declare an integer variable sum.
Declare an integer variable f1
Declare an integer variable f2
If n is less than 2
sum =n
else
set sum to 0
set f1 and f2 to 1
repeat n times
sum = f1 + f2
f2 = f1
f1 = sum
end loop
print sum
	`int n,k, f1, f2, sum;`
	`if ( n < 2 )`
	`sum =n;`
	`else`
	`{`
	`sum=0;`
	`f1 = f2 = 1;`
	
	`for(k=2; k<n; k++)`
	`{`
	`sum = f1 + f2;`
	`f2 = f1;`
	`f1 = sum;`
	`}`
	`}`
	`System.out.println("Fibonacci of number " + n + " is "+ sum);`

Remember that pseudocode is not language specific so we are not looking for "almost Java" code, but instead, we are looking for a strong understanding of the algorithm at hand.

addTicket()

If the type is "Child": create an instance of the Child class, set the price equal to the result of calculateTicketPrice(), add the Child into the ArrayList, and return the result of getPrice().

If the type is "Adult": create an instance of the Adult class, set the price equal to the result of calculateTicketPrice(), add the Adult into the ArrayList, and return the result of getPrice().

If the type is "Employee": Create an instance of the Employee class with the provided data. If that employee has visited less than twice, the price of the ticket is set to 0. Otherwise, the price is set to the result of calculateTicketPrice(). Add the ticket to the ArrayList and return the result of getPrice().

If the type is "MoviePass": Create an instance of the MoviePass Class. The default price will be calculated on the spot. If the person has not visited at all, then the price is set to 9.99. Otherwise, if the person has not seen any movies today and the feature is NONE, then the price will be set to 0. If none of this works, then the price will be the default price. Add the ticket to the ArrayList and return the result of getPrice().

getMoviePassTickets()

Create an ArrayList<Ticket> passes and an ArrayList<String> result. Step through the ArrayList of all tickets and add each index that is an instance of MoviePass into passes. Afterwards, use the method sortById() to sort passes by id. Finally, step through passes, adding the result of toString() of each member into result, and return result.