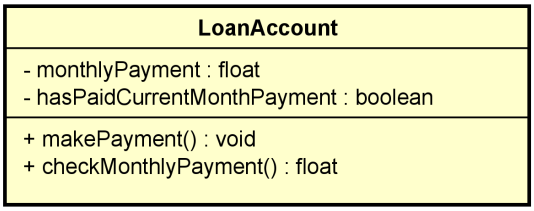
CSE 460
Software Analysis and Design
(Fall 2022)

Homework #2
Assigned: August September 5, 11:59 pm
Due: September 16, 11:59 pm

Posting ID 9816-379

1) (a)



The monthlyPayment is a floating-point attribute that will be used to store the amount of monthly payment.

The hasPaidCurrentMonthPayment is a Boolean stating whether has the owner paid the currently monthly payment or not.

The makePayment will provide the transaction process which will deduct the monthly payment from the user bank account or his credit card and then verified that the payment has gone through successfully. If the payment went through successfully then the makePayment will then set hasPaidCurrentMonthPayment to be true. If the payment is not successful, the makePayment will rollback to the original state of the system.

The checkMonthlyPayment will query for the owner's monthly payment through the database then print it onto the user's screen.

(b) The 2 pre-conditions for makePayments() method is that the owner must already be authenticated and the owner have not make the loan payment for the current month yet.

The post-condition method for makePayments() method is that the hasPaidCurrentMonthPayment Boolean attribute must be true after executing makePayments() method if it executes successfully.

The 2 pre-conditions for checkMonthlyPayment() method is that the owner must already be registered in the database for loan and the owner must be already be authenticated.

The post-condition for checkMonthlyPayment() method is that the current monthly payment amount must be returned to the caller and then perhaps be printed on screen for the owner to see.

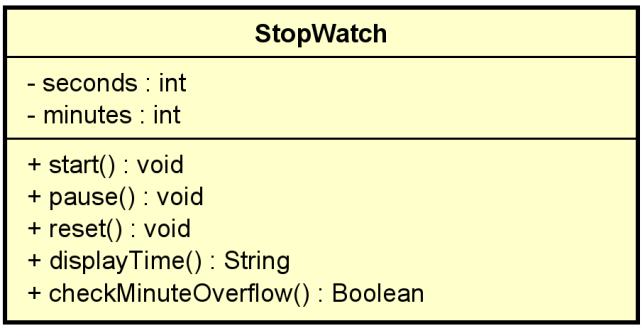
2) (a) With encapsulation of the monthlyPayment and hasPaidCurrentMonthPayment attribute, these attribute cannot be altered without the use of makePayment() method and checkMonthlyPayment() method.

Another benefit of encapsulation is that the code for the LoanAccount class will looks cleaner and more flexible. This is because we can change the code to be read-only or write-only through the getter and setter methods which in this case is makePayment() method and checkMonthlyPayment() method.

(b) One benefit of the code modularity for LoanAccount class is that the methods are separated into independent modules which do not rely on any other methods but depends on only its own attributes.

(c) The importance of both encapsulation and modularity in the LoanAccount class is that the code will be more maintainable and flexible. It is more maintainable since the methods are independent of each other. It is flexible since the attributes are being encapsulated such that if we want to change it, we can only use the getters and setters to do it. Thus it is flexible in the sense that we only need to change the getters and setters when we want to add or remove functionality from the LoanAccount class.

3) (a)



(b) The active methods are start(), pause() checkHourOverflow() and reset().

start() is an active method since it must start the execution flow of incrementing seconds and minutes in a timely manner once it is called and the process must remain running unless pause() method is called.

pause() is an active method since it must pause the increment of the time to effectively pause the stopwatch. It can only do so by altering the flow of the execution of the program by generating an interrupt to pause the increment of seconds and minutes through.

checkMinuteOverflow() is an active method since it must pause the execution of the incrementing of seconds and minute permanently when the time is already at 60 minutes. As the time displayed

on the screen must not exceed 60 minutes. So if the time is currently 60 minuites, the start() method should not be able to further increment time even if it is called.

reset() is an active method. When the time is 60 minutes, the reset() method is the only way to allow the start() method to increment the minutes and seconds again.

The passive method is displayTime().

displayTime() is just a function that updates the current minutes and seconds to the display when it is called. It does not have the ability to change flow of execution of the program.