PROJECT (a general description)

CS589; Fall 2014
Due Date: **December 8, 2014**
Late project: **50% penalty**
After **December 12, 2014** the project will not be accepted.

Object-Oriented and Model-Based Testing

The goal of this project is to test *GasPump* class that exhibits state behavior specified by the EFSM model. The source code of the class *GasPump* is provided in a separate file.

Description of the *GasPump* class:

The following operations are supported by the *GasPump* class:

GasPump()	//constructor
int Activate (int a, int b)	// the gas pump is activated where *a* represents the price of // Regular gas; *b* represents the price of Super gas
int Start()	//start the transaction
int PayCredit()	// pay for gas by a credit card
int Reject()	// credit card is rejected
int Cancel()	// cancel the transaction
int Approved()	// credit card is approved
int PayCash(int c)	// pay for gas by cash, where *c* represents prepaid cash
int Regular()	// Regular gas is selected
int Super()	// Super gas is selected
int StartPump()	// start pumping gas
int PumpGallon()	// one gallon of gas is disposed
int StopPump()	// stop pumping gas
int NoReceipt()	// no receipt
int Receipt()	// receipt is printed
int TurnOff()	// gas pump is turned off

Unless stated differently, each method (operation) returns 1 when the operation is successfully completed; otherwise, zero (0) value is returned.

The *GasPump* class is a state-based class that is used to control a simple gas pump. Users can pay by cash or a credit card. The gas pump disposes two types of gasoline: Regular and Super. The price of each type of gasoline is provided when the gas pump is activated. The detailed behavior of the *GasPump* class is specified by the EFSM model that is provided in a separate file.

TESTING

In this project the goal is to test the provided implementation (source code) of the *GasPump* class. In order to test the *GasPump* class, you are supposed to implement a testing environment that should contain a class test driver to execute test cases. The following testing methods should be used:

1. Model-Based Testing. Use the provided EFSM model to test the *GasPump* class. Design test cases for the *GasPump* class so that all 2-transition sequences testing criterion (all transition-pairs) is satisfied based on the provided EFSM, i.e., all 2-transition sequences are exercised during testing.
2. Design a set of additional test cases so each default (ghost) transition in the EFSM is tested.
3. Use multiple-condition testing to design additional test cases to test predicates of conditional-statements in operations. Notice that if a predicate contains only a simple condition, the multiple-condition testing is equivalent to the branch testing for this predicate.
4. Execute all test cases designed in steps 1, 2, and 3. For each test case, determine the correctness/incorrectness of the test results. If for a given test case the results are incorrect (test failed), identify the cause of incorrectness (a defect) in the source code of the *GasPump* class.

In the testing environment, you may need to introduce testing-oriented methods (in the *GasPump* class) that will be used to watch "internal states" of the *GasPump* object in order to determine the correctness/incorrectness of the results for some test cases.

Note: As a tester, you are not supposed to modify the logic (source code) of any operation of the *GasPump* class. In addition, notice that the source code under test may contain defects.

Sample test case:

Test #1: Activate(4,5), Start(), PayCredit(), Approved(), Super(), StartPump(), PumpGallon(), StopPump(), Receipt(), TurnOff()

Notice when the EFSM model is “executed” on this test (sequence of events), the following sequence of transitions are traversed: T_0, T_1, T_2, T_4, T_8, T_9, T_{10}, T_{11}, T_{13}, T_{15}

The detailed description for the project report and deliverables will be presented later on.