PROJECT (a general description)

CS589; Fall 2024
Due Date: **November 29, 2024**
Late project: **50% penalty**
After **December 5, 2024,** 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 (float a, float d) // the gas pump is activated where a represents the
                                // price of Regular gas; d represents the price of Diesel gas
int Credit()                    // 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 Cash(float c)               // pay for gas by cash, where c represents prepaid cash
int Regular()                   // Regular gas is selected
int Diesel()                    // Diesel gas is selected
int Start()                     // start pumping gas
int Pump()                      // one Gallon of gas is disposed
int Stop()                      // 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, a 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 with a credit card. The gas pump disposes two types of gasoline: Regular and Diesel. 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. Notice that the EFSM model specifies the expected behavior of the *GasPump* class.

TESTING

In this project, the goal is to test the provided implementation (source code) of the *GasPump* class. 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. Identify all **default transitions** in each state (including *Start* state). Design test cases that "execute" all identified default transitions.
3. Use **multiple-condition** testing to design additional test cases to test predicates of conditional-statements in operations/methods. 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. It is assumed that the provided EFSM represents the expected/correct behavior of the *GasPump* class. 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 need to introduce testing-oriented methods (in the *GasPump* class) that will be used to watch the "internal states" of the *GasPump* object to determine the correctness/incorrectness of the results for test cases.

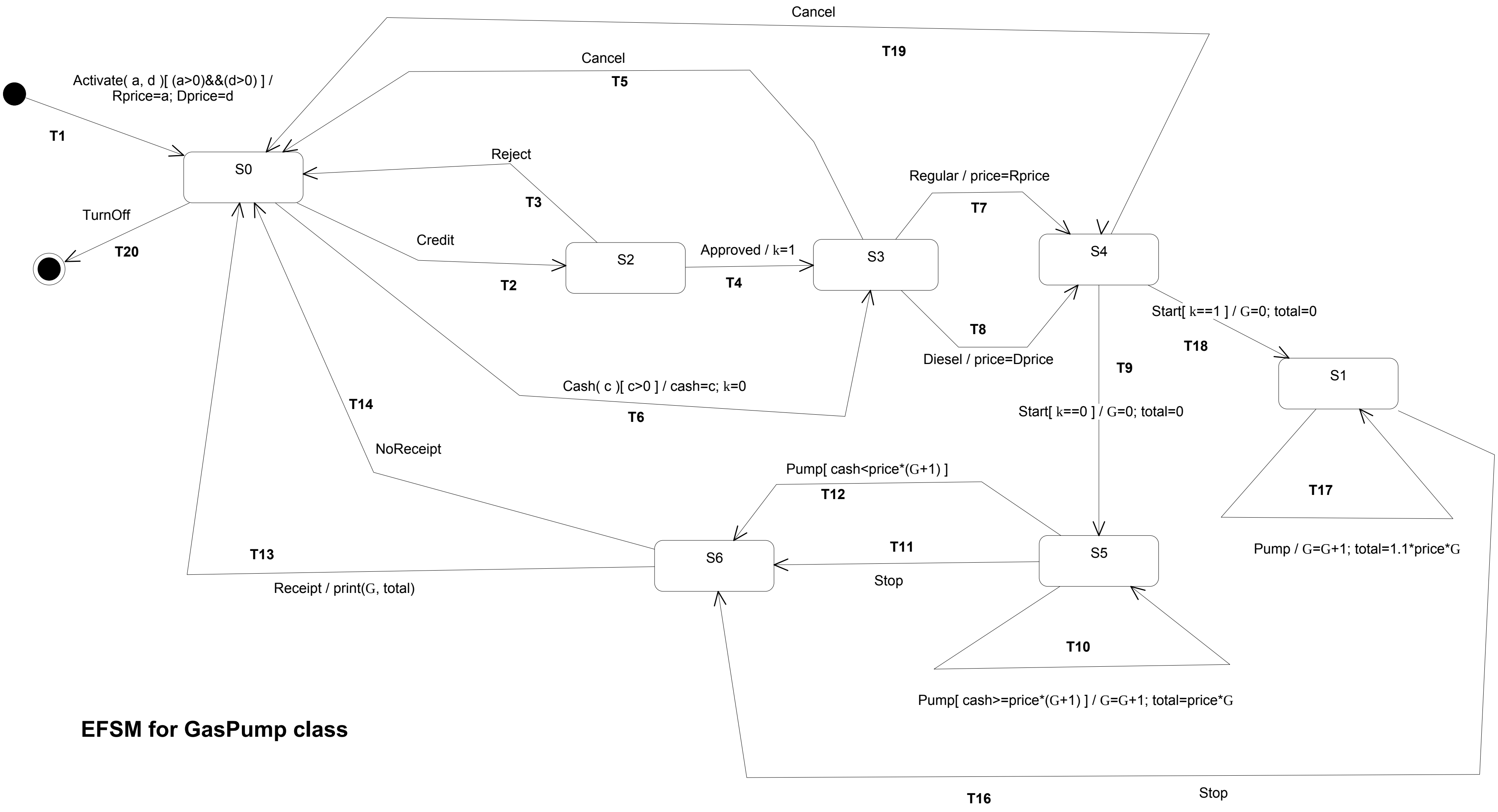
Note: As a tester, you are **NOT** supposed to modify the logic (source code) of any operation/method of the *GasPump* class. In addition, notice that the source code under test may contain defects.

Sample test case:

Test #1: Activate(4.5, 3.2), Credit(), Approved(), Diesel(), Start(), Pump(), Stop, Receipt(), TurnOff()

Notice when the EFSM model is "executed" on this test (sequence of events), the following sequence of transitions are traversed: T_1, T_2, T_4, T_8, T_{18}, T_{17}, T_{16}, T_{13}, T_{20}.

A detailed description of the project report and deliverables will be presented later on.


Activate(a, d)[(a>0)&&(d>0)] /
Rprice=a; Dprice=d
T1
S0
TurnOff
T20
Cancel
T19
Cancel
T5
Reject
T3
Credit
T2
S2
Approved / k=1
T4
S3
Regular / price=Rprice
T7
S4
T8
Diesel / price=Dprice
Cash(c)[c>0] / cash=c; k=0
T6
Start[k==1] / G=0; total=0
T18
S1
T9
Start[k==0] / G=0; total=0
T14
NoReceipt
Pump[cash<price*(G+1)]
T12
T17
Pump / G=G+1; total=1.1*price*G
T11
S6
S5
T13
Stop
Receipt / print(G, total)
T10
Pump[cash>=price*(G+1)] / G=G+1; total=price*G
T16
Stop

EFSM for GasPump class