Model-Based Testing Assignment

Pedro Tavares
`up201708899@fe.up.pt`

Raquel Dias
`up201700419@fe.up.pt`

15 December 2017

Note: You should have `Java 8+` and `Maven 3.5.0+` installed on your local machine.

Before we start: Make sure that you are at the root assignment folder. In order to open MISTA, execute the following command: `java -jar MISTA.jar`. Go to `File->Open`, navigate to `assignment->src->test->java->tvvs`, and select the `VendingMachine.xmid` file. Then, go to `Test->Options` and uncheck the *Add object reference to accessors* option. In order to edit the assignment source code and execute the generated tests, open/import the `assignment` folder with you favorite IDE.

Given the Petri Net defined on Figure **??**:

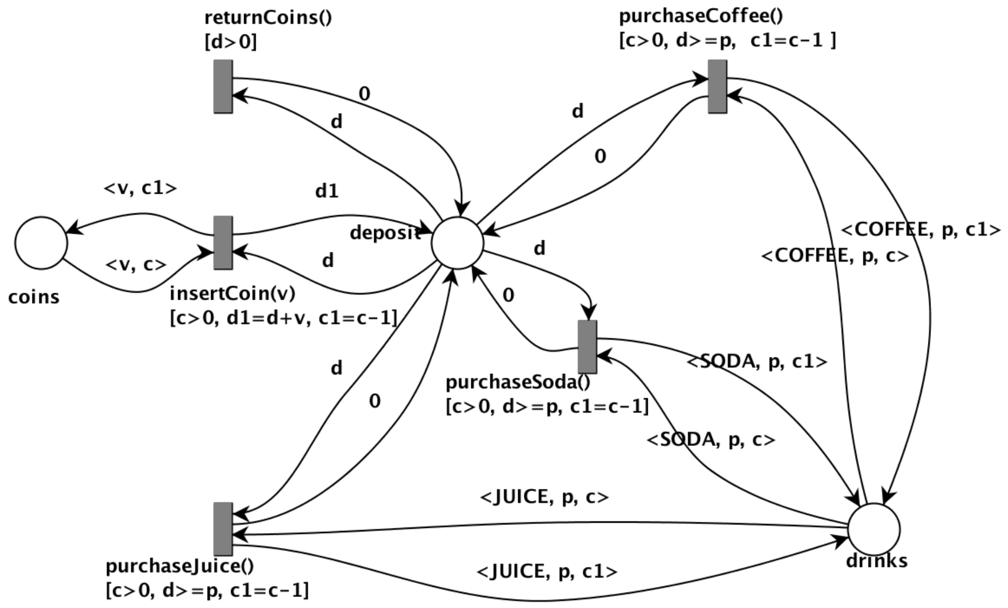

Figure 1: Petri Net that represents a vending machine system.

1. Implement the following methods on `VendindMachine` class:

   ```
   void insertCoin(Coin coin);
   void returnCoins();
   void purchase(String drink);
   ```

2. With MISTA opened, go to `MIM -> Methods` and map the Model-Level Events to the actual implementation code:

Model-Level Event	**Implementation Code**
`insertCoin(1)`	?
`insertCoin(5)`	?
`insertCoin(10)`	?
`insertCoin(25)`	?
`insertCoin(100)`	?
`returnCoins()`	?
`purchaseCoffee()`	?
`purchaseJuice()`	?
`purchaseSoda()`	?

Table 1: Empty mapping from Model-Level Event to Implementation Code

3. With the setup defined on Listing **??**, generate test cases based on the Reachability Tree coverage criteria (Select `Reachability Tree`, `Java`, and `JUnit`. Finally, go to `Test->Generate Test Code`). A file should be created inside the test folder. Run the generated tests with JUnit. **Note[1]**: before running the test generation, try to guess the number of tests that will cover the criteria. Compare the number and share your thoughts. **Note[2]**: The tests should pass. If they're failing, look for inconsistencies in both model and implementation. If you find any, fix it and re-generate the tests. Remember, a wrongly designed model can lead to unreliable test results.

   ```
   INIT coins(5,1), coins(10,1), coins(25,1),
   coins(100,1), deposit(0), drinks(COFFEE,35,1),
   drinks(JUICE,110,1), drinks(SODA,105,1)
   ```

Listing 1: This setup allow a maximum of four coins per test case, one per coin type (5, 10, 25, and 100). Deposit starts at 0. Each drink has it own price.

4. Change the setup values in order to accept more coins on each test scenario. Again, try to guess the number of test cases, obverse the test generation, execute the tests, and share your thoughts.

5. Repeat the process for other coverage criteria: Transition, State and Depth coverage.