<u>Problem #1 Model-based testing of the VendingMachine class:</u>

<u>We have the following pairs:</u>

<u>For idle:</u>
In: T1,T2,T3,T4,T6,T8,T9,T10,T11,T12,T13,T14,T15
Out: T2,T3,T4,T5,T6,T7
13*6 = 78 pairs
<u>For coins inserted:</u>
In: T7,T19,T20,T21,T23
Out: T10,T11,T12,T19,T20,T21,T22,T24,T25
5*9 = 45 pairs
<u>For sugar:</u>
In: T16,T17,T18,T22
Out: T13,T14,T15,T16,T17,T18,T23,T26,T27
4*9 = 36 pairs
<u>For no small cups:</u>
In: T25,T27,T28
Out: T9,T28
6 pairs
<u>For no large_cups:</u>
In: T24,T26,T29
Out: T8,T29
6 pairs

We have a total of 171 pairs, to simplify the document the pairs will be named with the tables presented in the next pages with the associated test for each pair.

For idle:

Pair #	Value	Test #	Pair #	Value	Test #	Pair #	Value	Test #	Pair #	Value	Test #
1	T1,T2	1	21	T4,T4	4	41	T9,T6	37	61	T13,T2	23
2	T1,T3	2	22	T4,T5	1	42	T9,T7	38	62	T13,T3	24
3	T1,T4	4	23	T4,T6	2	43	T10,T2	38	63	T13,T4	25
4	T1,T5	3	24	T4,T7	11	44	T10,T3	11	64	T13,T5	22
5	***T1,T6***	////	25	T6,T2	6	45	T10,T4	1	65	T13,T6	26
6	***T1,T7***	////	26	T6,T3	4	46	T10,T5	40	66	T13,T7	40
7	T2,T2	1	27	T6,T4	4	47	T10,T6	7	67	T14,T2	18
8	T2,T3	1	28	T6,T5	2	48	T10,T7	52	68	T14,T3	19
9	T2,T4	1	29	T6,T6	6	49	T11,T2	36	69	T14,T4	21
10	T2,T5	4	30	T6,T7	8	50	T11,T3	39	70	T14,T5	52
11	T2,T6	4	31	T8,T2	30	51	T11,T4	38	71	T14,T6	20
12	T2,T7	39	32	T8,T3	29	52	T11,T5	10	72	T14,T7	40
13	T3,T2	4	33	T8,T4	31	53	T11,T6	39	73	T15,T2	13
14	T3,T3	1	34	T8,T5	8	54	T11,T7	8	74	T15,T3	14
15	T3,T4	1	35	T8,T6	28	55	T12,T2	39	75	T15,T4	16
16	T3,T5	5	36	T8,T7	38	56	T12,T3	39	76	T15,T5	12
17	T3,T6	4	37	T9,T2	33	57	T12,T4	39	77	T15,T6	15
18	T3,T7	38	38	T9,T3	34	58	T12,T5	9	78	T15,T7	40
19	T4,T2	1	39	T9,T4	35	59	T12,T6	8			
20	T4,T3	4	40	T9,T5	32	60	T12,T7	17			

For coins inserted:

Pair #	Value	Test #	Pair #	Value	Test #	Pair #	Value	Test #
79	T7,T10	40	99	T20,T21	9	167	***T7,T12***	////
80	***T7,T11***	////	100	T20,T22	9	168	T19,T12	8
81	T7,T19	8	101	T20,T24	40	169	T20,T12	39
82	T7,T20	52	102	T20,T25	40	170	***T21,T12***	////
83	T7,T21	12	103	T21,T10	38	171	T23,T12	9
84	T7,T22	10	104	T21,T11	36			
85	***T7,T24***	////	105	T21,T19	11			
86	***T7,T25***	////	106	T21,T20	11			
87	T19,T10	52	107	T21,T21	10			
88	***T19,T11***	////	108	T21,T22	9			
89	T19,T19	8	109	***T21,T24***	////			
90	T19,T20	11	110	T21,T25	32			
91	T19,T21	11	111	T23,T10	10			
92	T19,T22	9	112	T23,T11	10			
93	T19,T24	8	113	T23,T19	9			
94	***T19,T25***	////	114	T23,T20	52			
95	T20,T10	11	115	T23,T21	9			
96	T20,T11	38	116	T23,T22	9			
97	T20,T19	11	117	T23,T24	41			
98	T20,T20	9	118	T23,T25	41			

For sugar:

Pair #	Value	Test #	Pair #	Value	Test #
119	T16,T13	42	139	***T18,T15***	////
120	T16,T14	27	140	T18,T16	27
121	T16,T15	13	141	T18,T17	15
122	T16,T16	13	142	T18,T18	15
123	T16,T17	27	143	T18,T23	38
124	T16,T18	27	144	T18,T26	31
125	T16,T23	27	145	***T18,T27***	////
126	T16,T26	37	146	T22,T13	23
127	T16,T27	37	147	T22,T14	17
128	***T17,T13***	////	148	T22,T15	12
129	T17,T14	43	149	T22,T16	13
130	T17,T15	14	150	T22,T17	10
131	T17,T16	27	151	T22,T18	15
132	T17,T17	14	152	T22,T23	9
133	T17,T18	43	153	T22,T26	22
134	T17,T23	10	154	T22,T27	34
135	***T17,T26***	////			
136	T17,T27	35			
137	T18,T13	40			
138	T18,T14	16			

For no small cups:

Pair #	Value	Test #
155	T25,T9	32
156	T25,T28	33
157	T27,T9	34
158	T27,T28	35
159	T28,T9	33
160	T28,T28	33

For no large_cups:

Pair #	Value	Test #
161	T24,T8	8
162	T24,T29	29
163	T26,T8	30
164	T26,T29	31
165	T29,T8	29
166	T29,T29	29

Impossible transitions:

The pair ***T1,T6*** is impossible because when a vending machine is created, we have 0+25<0 required to do T6 which is always false.

The pair ***T1,T7*** is impossible because when a vending machine is created, we have price == 0 required to do T7 which is always false.

The pair ***T7,T11*** is impossible because after doing T7, s == 0 which is incompatible with the test s == 2 in T11.

The pair ***T7,T12*** is impossible because after doing T7, s == 0 which is incompatible with the test s == 1 in T12.

The pair ***T7,T24*** is impossible because after doing T7, s == 0 which is incompatible with the test s == 1 in T24.

The pair ***T7,T25*** is impossible because after doing T7, s == 0 which is incompatible with the test s == 2 in T25.

The pair ***T19,11*** is impossible because after doing T19, s == 1 which is incompatible with the test s == 2 in T11.

The pair ***T19,25*** is impossible because after doing T19, s == 1 which is incompatible with the test s == 2 in T25.

The pair ***T21,24*** is impossible because after doing T21, s == 2 which is incompatible with the test s == 1 in T24.

The pair ***T21,T12*** is impossible because after doing T21, s == 2 which is incompatible with the test s == 1 in T12.

The pair ***T17,T13*** is impossible because after doing T17, s == 2 which is incompatible with the test s == 1 in T13.

The pair ***T17,T26*** is impossible because after doing T17, s == 2 which is incompatible with the test s == 1 in T26.

The pair ***T18,T15*** is impossible because after doing T18, s == 1 which is incompatible with the test s == 2 in T15.

The pair ***T18,T27*** is impossible because after doing T18, s == 1 which is incompatible with the test s == 2 in T27.

Problem #2 Testing default (ghost) transitions of the VendingMachine class:

We have to test for each state every possible events:
coin()
small_cup()
large_cup()
sugar()
tea()
insert_large_cups(int n)
insert_small_cups(int n)
set_price(int p)
cancel()
dispose()

For idle we have to test additionally:
small_cup()
large_cup()
sugar()
tea()
insert_large_cups(int n)[n<=0]
insert_small_cups(int n)[n<=0]
set_price(int p)[p<=0]
cancel()

All are executed in test 44.

For coins inserted we have to test additionally:
tea()[!((k1>1)&&(s==1))&&!((k1>1)&&(s==2))&&!((k1==1)&&(s==1))&&!((k1==1)&&(s==2))] //
Not T11/T12/T24/T25
insert_large_cups(int n)
insert_small_cups(int n)
set_price(int p)
dispose()
All are executed in test 45.

For sugar we have to test additionally:
tea()[!((k1>1)&&(s==1))&&!((k1>1)&&(s==2))&&!((k1==1)&&(s==1))&&!((k1==1)&&(s==2))] //
Not T13/T15/T26/T27
insert_large_cups(int n)
insert_small_cups(int n)
set_price(int p)
dispose()

All are executed in test 46.

<u>For no small cups we have to test additionally:</u>
small_cup()
large_cup()
sugar()
tea()
insert_large_cups(int n)[n<=0]
insert_small_cups(int n)[n<=0]
set_price(int p)
cancel()
dispose()

All are executed in test 47.

<u>For no large_cups we have to test additionally:</u>
small_cup()
large_cup()
sugar()
tea()
insert_large_cups(int n)[n<=0]
insert_small_cups(int n)[n<=0]
set_price(int p)
cancel()
dispose()

All are executed in test 48.

Problem #3 Multiple-condition testing:

For the coin method:

The true branch of if(x == 1) is tested in test #1 while the false branch is tested on test #8.

t+25 >= price	price > 0	Test #
True	True	7
True	False	5
False	True	2
False	False	Impossible, t is a positive integer if we add 25 to this integer, he still greater or equals to 25 which is always true if the price is equals or less than 0.

The true branch of if(t + 25 < price) is tested in test #2 while the false branch is tested in the test #5.

x > 1	x < 6	Test #
True	True	9
True	False	Impossible, there is not state such that x >= 6 (we can have x = 6 after calling dispose, but this is not supposed to be a state where we are supposed to be able anything)
False	True	Impossible, the only state where x>1 is false is state 1, but not being in state 1 is required to reach this test
False	False	Impossible, there is not state such that x >= 6 (we can have x = 6 after calling dispose, but this is not supposed to be a state where we are supposed to be able anything)

For the small_cup method:

x == 2	x == 3	Test #
True	True	Impossible, x cannot have two different values
True	False	9
False	True	12
False	False	44

For the large_cup method:

x == 2	x == 3	Test #
True	True	Impossible, x cannot have two different values
True	False	9
False	True	15
False	False	44

For the sugar method:

x == 2	x == 3	Test #
True	True	Impossible, x cannot have two different values
True	False	9
False	True	9
False	False	44

Both true and false branches of the if(x==2) else are tested in test #9.

For the tea method:

If #1:

x == 2	x == 3	Test #
True	True	Impossible, x cannot have two different values
True	False	8
False	True	12
False	False	44

If #2:

x == 2	k1 > 1	s == 2	Test #
True	True	True	10
True	True	False	8
True	False	True	32
True	False	False	28
False	True	True	12
False	True	False	22
False	False	True	34
False	False	False	30

If #3:

x == 2	k > 1	s == 1	Test #
True	True	True	8
True	True	False	38
True	False	True	8
True	False	False	32
False	True	True	22
False	True	False	12
False	False	True	30
False	False	False	34

If #4:

x == 2	k == 1	s == 1	Test #
True	True	True	8
True	True	False	32
True	False	True	49
True	False	False	38
False	True	True	30
False	True	False	34
False	False	True	22
False	False	False	12

If #5:

x == 2	k1 == 1	s == 2	Test #
True	True	True	32
True	True	False	30
True	False	True	40
True	False	False	37
False	True	True	36
False	True	False	42
False	False	True	12
False	False	False	22

If #6:

x == 3	k1 == 1	s == 2	Test #
True	True	True	34
True	True	False	30
True	False	True	12
True	False	False	22
False	True	True	Impossible see below
False	True	False	54
False	False	True	53
False	False	False	53

The 4 previous test are not executable by using the state machine, because x == 3 false means x == 2 if x== 2 due to the **if #5** intercept the true/true case by ending with return.

If #7:

x == 3	k == 1	s == 1	Test #
True	True	True	30
True	True	False	40
True	False	True	22
True	False	False	12
False	True	True	Impossible see below
False	True	False	54
False	False	True	53
False	False	False	53

The 4 previous test are not executable by using the state machine, because x == 3 false means x == 2, if x== 2 due to the **if #4** intercept the true/true case by ending with return.

If #8:

x == 3	k1 > 1	s == 2	Test #
True	True	True	12
True	True	False	22
True	False	True	50
True	False	False	42
False	True	True	Impossible see below
False	True	False	55
False	False	True	53
False	False	False	53

The 4 previous test are not executable by using the state machine, because x == 3 false means x == 2, if x== 2 due to the **if #2** intercept the true/true case by ending with return.

If #9:

x == 3	k > 1	s == 1	Test #
True	True	True	22
True	True	False	50
True	False	True	51
True	False	False	51
False	True	True	Impossible see below
False	True	False	55
False	False	True	53
False	False	False	53

The 4 previous test are not executable by using the state machine, because x == 3 false means x == 2, if x== 2 due to the **if #3** intercept the true/true case by ending with return.

For the insert_large_cups method:

x == 1	n > 0	Test #
True	True	1
True	False	44
False	True	45
False	False	45

x == 5	n > 0	Test #
True	True	28
True	False	48
False	True	45
False	False	45

For the insert_small_cups method:

x == 1	n > 0	Test #
True	True	1
True	False	44
False	True	45
False	False	45

x == 4	n > 0	Test #
True	True	32
True	False	47
False	True	45
False	False	45

For the set_price method:

x == 1	p > 0	Test #
True	True	1
True	False	44
False	True	45
False	False	45

For the cancel method:

x == 2	x == 3	Test #
True	True	Impossible, x cannot have two different values
True	False	7
False	True	17
False	False	44

For the dispose method:
The true branch of the test x == 1 is covered in test #1.
The false branch of the test x == 1 is covered in test #45.

Problem #4 A Test Suit and the results of its execution:

The test suite is:

Test#1: insert_large_cups 2 insert_small_cups 3 insert_small_cups 3 set_price 25 insert_large_cups 2 set_price 25 dispose
Test#2: insert_small_cups 3 set_price 100 coin dispose
Test#3: dispose
Test#4: set_price 100 set_price 75 insert_small_cups 3 insert_large_cups 2 coin set_price 100 insert_large_cups 2 dispose
Test#5: insert_large_cups 2 insert_small_cups 3 dispose
Test#6: set_price 100 coin coin insert_large_cups 2 dispose
Test#7: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin cancel dispose
Test#8: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin large_cup large_cup tea coin coin large_cup tea insert_large_cups 1 dispose
Test#9: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin coin coin small_cup sugar sugar coin sugar sugar small_cup sugar sugar large_cup sugar sugar sugar sugar tea dispose
Test#10: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin small_cup small_cup sugar sugar cancel coin coin sugar small_cup sugar tea dispose
Test#11: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin small_cup coin large_cup small_cup large_cup coin cancel set_price 25 coin large_cup cancel insert_small_cups 1 dispose
Test#12: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin small_cup sugar tea dispose
Test#13: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin small_cup sugar coin coin tea insert_large_cups 1 dispose
Test#14: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin small_cup sugar small_cup small_cup tea insert_small_cups 1 dispose
Test#15: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin small_cup sugar large_cup small_cup tea coin dispose
Test#16: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin small_cup sugar tea set_price 25 dispose
Test#17: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin sugar cancel dispose
Test#18: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin sugar cancel insert_large_cups 1 dispose
Test#19: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin sugar cancel insert_small_cups 1 dispose
Test#20: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin sugar cancel coin dispose
Test#21: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin sugar cancel set_price 25 dispose
Test#22: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin large_cup sugar tea dispose

Test#23: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin large_cup sugar tea insert_large_cups 2 dispose
Test#24: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin large_cup sugar tea insert_small_cups 2 dispose
Test#25: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin large_cup sugar tea set_price 25 dispose
Test#26: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin large_cup sugar tea coin dispose
Test#27: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin large_cup sugar coin small_cup coin large_cup coin sugar sugar coin cancel dispose
Test#28: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin large_cup tea insert_large_cups 1 coin dispose
Test#29: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin large_cup tea coin coin insert_large_cups 1 insert_small_cups 1 dispose
Test#30: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin large_cup sugar tea insert_large_cups 1 insert_large_cups 1 dispose
Test#31: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin sugar large_cup tea coin coin insert_large_cups 1 set_price 25 dispose
Test#32: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin small_cup tea insert_small_cups 1 dispose
Test#33: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin small_cup tea coin coin insert_small_cups 1 insert_small_cups 1 dispose
Test#34: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin small_cup sugar tea insert_small_cups 1 insert_small_cups 1 dispose
Test#35: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin sugar small_cup tea coin coin insert_small_cups 1 set_price 25 dispose
Test#36: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin small_cup tea insert_large_cups 1 coin coin small_cup sugar tea insert_small_cups 2 dispose
Test#37: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin small_cup sugar coin tea insert_small_cups 2 coin coin large_cup sugar coin tea insert_large_cups 1 dispose
Test#38: insert_large_cups 2 insert_small_cups 1 set_price 25 coin small_cup tea insert_small_cups 2 coin large_cup tea insert_large_cups 3 coin small_cup coin tea set_price 25 insert_small_cups 1 coin small_cup cancel insert_large_cups 1 dispose
Test#39: insert_large_cups 10 insert_small_cups 10 set_price 25 coin small_cup tea insert_small_cups 1 set_price 50 coin coin small_cup tea coin coin large_cup tea coin coin large_cup coin tea set_price 25 coin large_cup tea insert_large_cups 1 coin large_cup tea coin large_cup tea insert_small_cups 1 dispose
Test#40: insert_large_cups 1 insert_small_cups 1 set_price 25 coin large_cup coin tea insert_large_cups 2 coin small_cup coin tea insert_small_cups 2 coin sugar cancel coin sugar large_cup tea coin sugar small_cup tea coin cancel dispose
Test#41: insert_large_cups 1 insert_small_cups 1 set_price 25 coin small_cup sugar sugar tea insert_small_cups 1 coin large_cup sugar sugar tea insert_large_cups 1 dispose
Test#42: insert_large_cups 2 insert_small_cups 1 set_price 25 coin sugar large_cup cancel coin sugar large_cup sugar sugar coin tea dispose

```
Test#43: insert_large_cups 2 insert_small_cups 2 set_price 25 coin sugar small_cup cancel coin
sugar small_cup large_cup small_cup sugar sugar coin tea dispose
Test#44: small_cup large_cup sugar tea insert_large_cups 0 insert_small_cups 0 set_price 0
cancel dispose
Test#45: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin tea insert_large_cups -
1 insert_small_cups -1 set_price -1 insert_large_cups 1 insert_small_cups 1 set_price 1 dispose
cancel dispose
Test#46: insert_large_cups 2 insert_small_cups 2 set_price 50 coin coin sugar tea
insert_large_cups -1 insert_small_cups -1 set_price -1 insert_large_cups 1 insert_small_cups 1
set_price 1 dispose sugar cancel dispose
Test#47: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin small_cup tea
small_cup large_cup sugar tea insert_large_cups -1 insert_small_cups -1 set_price -1
insert_large_cups 1 set_price 1 cancel dispose insert_small_cups 1 dispose
Test#48: insert_large_cups 1 insert_small_cups 1 set_price 50 coin coin large_cup tea
small_cup large_cup sugar tea insert_large_cups -1 insert_small_cups -1 set_price -1
insert_small_cups 1 set_price 1 cancel dispose insert_large_cups 1 dispose
Test#49: insert_small_cups 1 set_price 25 coin large_cup tea cancel dispose
Test#50: insert_large_cups 2 set_price 25 coin small_cup sugar tea cancel dispose
Test#51: set_price 25 coin sugar small_cup tea large_cup tea insert_large_cups 2 coin sugar
small_cup tea cancel dispose
Test#52: insert_small_cups 2 set_price 25 insert_large_cups 2 coin coin sugar sugar coin
large_cup cancel coin sugar cancel dispose
Test#53: set_price 25 coin tea small_cup tea large_cup tea cancel dispose
Test#54: set_price 25 insert_large_cups 1 insert_small_cups 1 coin tea cancel dispose
Test#55: set_price 25 insert_large_cups 25 insert_small_cups 25 coin tea cancel dispose
$$ $$
```

This is the output produced by the execution of the test suite in the test driver is for the TS.txt:

```
Begin of the execution of Test#1:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 3 executed and return 1.
insert_small_cups with parameter 3 executed and return 1.
set_price with parameter 25 executed and return 1.
insert_large_cups with parameter 2 executed and return 1.
set_price with parameter 25 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#1.

Begin of the execution of Test#2:
insert_small_cups with parameter 3 executed and return 1.
set_price with parameter 100 executed and return 1.
coin was executed and return 1.
```

```
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#2.

Begin of the execution of Test#3:
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#3.

Begin of the execution of Test#4:
set_price with parameter 100 executed and return 1.
set_price with parameter 75 executed and return 1.
insert_small_cups with parameter 3 executed and return 1.
insert_large_cups with parameter 2 executed and return 1.
coin was executed and return 1.
set_price with parameter 100 executed and return 1.
insert_large_cups with parameter 2 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#4.

Begin of the execution of Test#5:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 3 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#5.

Begin of the execution of Test#6:
set_price with parameter 100 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
insert_large_cups with parameter 2 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#6.

Begin of the execution of Test#7:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
RETURN COINS
```

```
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#7.

Begin of the execution of Test#8:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#8.

Begin of the execution of Test#9:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
RETURN COIN
coin was executed and return 1.
RETURN COIN
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#9.

Begin of the execution of Test#10:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#10.

Begin of the execution of Test#11:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
RETURN COIN
coin was executed and return 1.
```

```
large_cup was executed and return 1.
small_cup was executed and return 1.
large_cup was executed and return 1.
RETURN COIN
coin was executed and return 1.
RETURN COINS
cancel was executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
RETURN COINS
cancel was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#11.

Begin of the execution of Test#12:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#12.

Begin of the execution of Test#13:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
```

```
tea was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#13.

Begin of the execution of Test#14:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#14.

Begin of the execution of Test#15:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
large_cup was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
coin was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#15.

Begin of the execution of Test#16:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
```

```
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
set_price with parameter 25 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#16.

Begin of the execution of Test#17:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#17.

Begin of the execution of Test#18:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#18.

Begin of the execution of Test#19:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
```

```
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#19.

Begin of the execution of Test#20:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
coin was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#20.

Begin of the execution of Test#21:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
set_price with parameter 25 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#21.

Begin of the execution of Test#22:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
```

```
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#22.

Begin of the execution of Test#23:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
insert_large_cups with parameter 2 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#23.

Begin of the execution of Test#24:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#24.

Begin of the execution of Test#25:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
```

```
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
set_price with parameter 25 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#25.

Begin of the execution of Test#26:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
coin was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#26.

Begin of the execution of Test#27:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
small_cup was executed and return 1.
RETURN COIN
coin was executed and return 1.
large_cup was executed and return 1.
RETURN COIN
coin was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
RETURN COINS
cancel was executed and return 1.
```

```
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#27.

Begin of the execution of Test#28:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
coin was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#28.

Begin of the execution of Test#29:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
RETURN COIN
coin was executed and return 1.
RETURN COIN
coin was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#29.

Begin of the execution of Test#30:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
```

```
large_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#30.

Begin of the execution of Test#31:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
RETURN COIN
coin was executed and return 1.
RETURN COIN
coin was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
set_price with parameter 25 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#31.

Begin of the execution of Test#32:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#32.
```

```
Begin of the execution of Test#33:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
RETURN COIN
coin was executed and return 1.
RETURN COIN
coin was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#33.

Begin of the execution of Test#34:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#34.

Begin of the execution of Test#35:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
```

```
tea was executed and return 1.
RETURN COIN
coin was executed and return 1.
RETURN COIN
coin was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 25 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#35.

Begin of the execution of Test#36:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#36.

Begin of the execution of Test#37:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
```

```
insert_small_cups with parameter 2 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#37.

Begin of the execution of Test#38:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
insert_large_cups with parameter 3 executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
set_price with parameter 25 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
RETURN COINS
cancel was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#38.
```

```
Begin of the execution of Test#39:
insert_large_cups with parameter 10 executed and return 1.
insert_small_cups with parameter 10 executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
```

```
dispose was executed and return 1.
End of the successful execution of Test#39.

Begin of the execution of Test#40:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
insert_large_cups with parameter 2 executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
coin was executed and return 1.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#40.

Begin of the execution of Test#41:
insert_large_cups with parameter 1 executed and return 1.
```

```
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#41.

Begin of the execution of Test#42:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
large_cup was executed and return 1.
RETURN COINS
cancel was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
large_cup was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE LARGE CUP OF TEA WITH SUGAR
tea was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#42.

Begin of the execution of Test#43:
insert_large_cups with parameter 2 executed and return 1.
```

```
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
RETURN COINS
cancel was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
large_cup was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
DISPOSE SMALL CUP OF TEA WITH SUGAR
tea was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#43.

Begin of the execution of Test#44:
small_cup was executed and return 0.
large_cup was executed and return 0.
sugar was executed and return 0.
tea was executed and return 0.
insert_large_cups with parameter 0 executed and return 0.
insert_small_cups with parameter 0 executed and return 0.
set_price with parameter 0 executed and return 0.
cancel was executed and return 0.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#44.

Begin of the execution of Test#45:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
tea was executed and return 0.
insert_large_cups with parameter -1 executed and return 0.
insert_small_cups with parameter -1 executed and return 0.
```

```
set_price with parameter -1 executed and return 0.
insert_large_cups with parameter 1 executed and return 0.
insert_small_cups with parameter 1 executed and return 0.
set_price with parameter 1 executed and return 0.
dispose was executed and return 0.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#45.

Begin of the execution of Test#46:
insert_large_cups with parameter 2 executed and return 1.
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
tea was executed and return 0.
insert_large_cups with parameter -1 executed and return 0.
insert_small_cups with parameter -1 executed and return 0.
set_price with parameter -1 executed and return 0.
insert_large_cups with parameter 1 executed and return 0.
insert_small_cups with parameter 1 executed and return 0.
set_price with parameter 1 executed and return 0.
dispose was executed and return 0.
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#46.

Begin of the execution of Test#47:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
DISPOSE SMALL CUP OF TEA
tea was executed and return 1.
small_cup was executed and return 0.
large_cup was executed and return 0.
```

```
sugar was executed and return 0.
tea was executed and return 0.
insert_large_cups with parameter -1 executed and return 0.
insert_small_cups with parameter -1 executed and return 0.
set_price with parameter -1 executed and return 0.
insert_large_cups with parameter 1 executed and return 0.
set_price with parameter 1 executed and return 0.
cancel was executed and return 0.
dispose was executed and return 0.
insert_small_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#47.

Begin of the execution of Test#48:
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 50 executed and return 1.
coin was executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
DISPOSE LARGE CUP OF TEA
tea was executed and return 1.
small_cup was executed and return 0.
large_cup was executed and return 0.
sugar was executed and return 0.
tea was executed and return 0.
insert_large_cups with parameter -1 executed and return 0.
insert_small_cups with parameter -1 executed and return 0.
set_price with parameter -1 executed and return 0.
insert_small_cups with parameter 1 executed and return 0.
set_price with parameter 1 executed and return 0.
cancel was executed and return 0.
dispose was executed and return 0.
insert_large_cups with parameter 1 executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#48.

Begin of the execution of Test#49:
insert_small_cups with parameter 1 executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
large_cup was executed and return 1.
```

```
tea was executed and return 0.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#49.

Begin of the execution of Test#50:
insert_large_cups with parameter 2 executed and return 1.
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
small_cup was executed and return 1.
sugar was executed and return 1.
tea was executed and return 0.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#50.

Begin of the execution of Test#51:
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
tea was executed and return 0.
large_cup was executed and return 1.
tea was executed and return 0.
insert_large_cups with parameter 2 executed and return 0.
RETURN COIN
coin was executed and return 1.
sugar was executed and return 1.
small_cup was executed and return 1.
tea was executed and return 0.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#51.

Begin of the execution of Test#52:
insert_small_cups with parameter 2 executed and return 1.
set_price with parameter 25 executed and return 1.
insert_large_cups with parameter 2 executed and return 1.
```

```
coin was executed and return 1.
RETURN COIN
coin was executed and return 1.
sugar was executed and return 1.
sugar was executed and return 1.
RETURN COIN
coin was executed and return 1.
large_cup was executed and return 1.
RETURN COINS
cancel was executed and return 1.
coin was executed and return 1.
sugar was executed and return 1.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#52.

Begin of the execution of Test#53:
set_price with parameter 25 executed and return 1.
coin was executed and return 1.
tea was executed and return 0.
small_cup was executed and return 1.
tea was executed and return 0.
large_cup was executed and return 1.
tea was executed and return 0.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#53.

Begin of the execution of Test#54:
set_price with parameter 25 executed and return 1.
insert_large_cups with parameter 1 executed and return 1.
insert_small_cups with parameter 1 executed and return 1.
coin was executed and return 1.
tea was executed and return 0.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#54.
```

```
Begin of the execution of Test#55:
set_price with parameter 25 executed and return 1.
insert_large_cups with parameter 25 executed and return 1.
insert_small_cups with parameter 25 executed and return 1.
coin was executed and return 1.
tea was executed and return 0.
RETURN COINS
cancel was executed and return 1.
SHUT DOWN
dispose was executed and return 1.
End of the successful execution of Test#55.
```

The output of the automated test suite is:

```
[TestNG] Running:
  Command line suite

SHUT DOWN
The test1 passed!
SHUT DOWN
The test2 passed!
SHUT DOWN
The test3 passed!
SHUT DOWN
The test4 passed!
SHUT DOWN
The test5 passed!
SHUT DOWN
The test6 passed!
RETURN COINS
SHUT DOWN
The test7 passed!
DISPOSE LARGE CUP OF TEA
DISPOSE LARGE CUP OF TEA
SHUT DOWN
The test8 passed!
RETURN COIN
RETURN COIN
RETURN COIN
DISPOSE LARGE CUP OF TEA
SHUT DOWN
The test9 passed!
RETURN COINS
DISPOSE SMALL CUP OF TEA
```

```
SHUT DOWN
The test10 passed!
RETURN COIN
RETURN COIN
RETURN COINS
RETURN COINS
SHUT DOWN
The test11 passed!
DISPOSE SMALL CUP OF TEA WITH SUGAR
SHUT DOWN
The test12 passed!
RETURN COIN
RETURN COIN
DISPOSE SMALL CUP OF TEA WITH SUGAR
SHUT DOWN
The test13 passed!
DISPOSE SMALL CUP OF TEA WITH SUGAR
SHUT DOWN
The test14 passed!
DISPOSE SMALL CUP OF TEA WITH SUGAR
SHUT DOWN
The test15 passed!
DISPOSE SMALL CUP OF TEA WITH SUGAR
SHUT DOWN
The test16 passed!
RETURN COINS
SHUT DOWN
The test17 passed!
RETURN COINS
SHUT DOWN
The test18 passed!
RETURN COINS
SHUT DOWN
The test19 passed!
RETURN COINS
SHUT DOWN
The test20 passed!
RETURN COINS
SHUT DOWN
The test21 passed!
DISPOSE LARGE CUP OF TEA WITH SUGAR
SHUT DOWN
The test22 passed!
DISPOSE LARGE CUP OF TEA WITH SUGAR
```

```
SHUT DOWN
The test23 passed!
DISPOSE LARGE CUP OF TEA WITH SUGAR
SHUT DOWN
The test24 passed!
DISPOSE LARGE CUP OF TEA WITH SUGAR
SHUT DOWN
The test25 passed!
DISPOSE LARGE CUP OF TEA WITH SUGAR
SHUT DOWN
The test26 passed!
RETURN COIN
RETURN COIN
RETURN COIN
RETURN COIN
RETURN COINS
SHUT DOWN
The test27 passed!
DISPOSE LARGE CUP OF TEA
SHUT DOWN
The test28 passed!
DISPOSE LARGE CUP OF TEA
RETURN COIN
RETURN COIN
SHUT DOWN
The test29 passed!
DISPOSE LARGE CUP OF TEA WITH SUGAR
SHUT DOWN
The test30 passed!
DISPOSE LARGE CUP OF TEA WITH SUGAR
RETURN COIN
RETURN COIN
SHUT DOWN
The test31 passed!
DISPOSE SMALL CUP OF TEA
SHUT DOWN
The test32 passed!
DISPOSE SMALL CUP OF TEA
RETURN COIN
RETURN COIN
SHUT DOWN
The test33 passed!
DISPOSE SMALL CUP OF TEA WITH SUGAR
SHUT DOWN
```

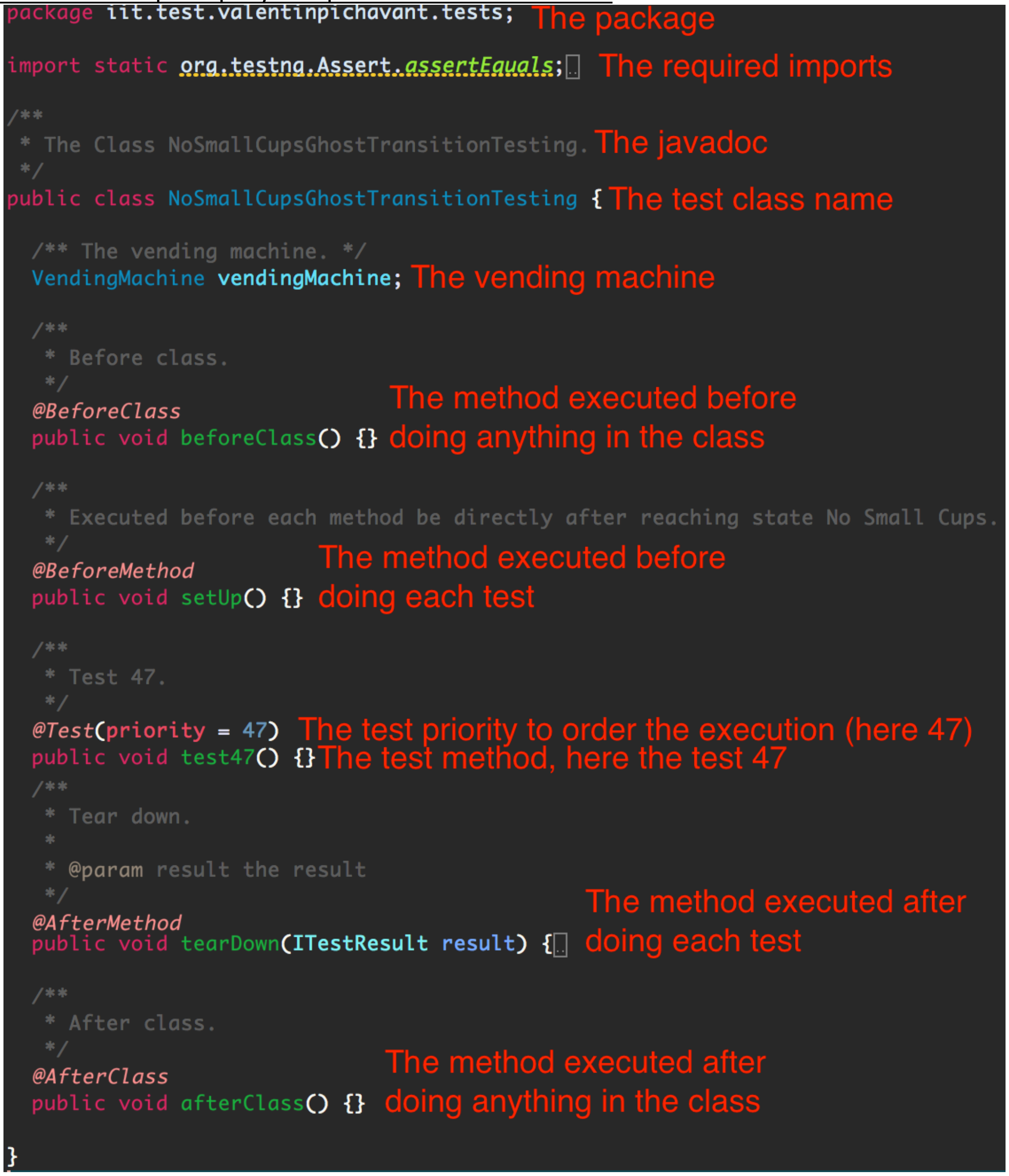
```
The test34 passed!
DISPOSE SMALL CUP OF TEA WITH SUGAR
RETURN COIN
RETURN COIN
SHUT DOWN
The test35 passed!
DISPOSE SMALL CUP OF TEA
DISPOSE SMALL CUP OF TEA WITH SUGAR
SHUT DOWN
The test36 passed!
RETURN COIN
DISPOSE SMALL CUP OF TEA WITH SUGAR
RETURN COIN
DISPOSE LARGE CUP OF TEA WITH SUGAR
SHUT DOWN
The test37 passed!
DISPOSE SMALL CUP OF TEA
DISPOSE LARGE CUP OF TEA
RETURN COIN
DISPOSE SMALL CUP OF TEA
RETURN COINS
SHUT DOWN
The test38 passed!
DISPOSE SMALL CUP OF TEA
DISPOSE SMALL CUP OF TEA
DISPOSE LARGE CUP OF TEA
RETURN COIN
DISPOSE LARGE CUP OF TEA
DISPOSE LARGE CUP OF TEA
DISPOSE LARGE CUP OF TEA
DISPOSE LARGE CUP OF TEA
SHUT DOWN
The test39 passed!
RETURN COIN
DISPOSE LARGE CUP OF TEA
RETURN COIN
DISPOSE SMALL CUP OF TEA
RETURN COINS
DISPOSE LARGE CUP OF TEA WITH SUGAR
DISPOSE SMALL CUP OF TEA WITH SUGAR
RETURN COINS
SHUT DOWN
The test40 passed!
DISPOSE SMALL CUP OF TEA
DISPOSE LARGE CUP OF TEA
SHUT DOWN
The test41 passed!
RETURN COINS
RETURN COIN
DISPOSE LARGE CUP OF TEA WITH SUGAR
SHUT DOWN
The test42 passed!
RETURN COINS
RETURN COIN
DISPOSE SMALL CUP OF TEA WITH SUGAR
SHUT DOWN
The test43 passed!
SHUT DOWN
The test44 passed!
RETURN COINS
SHUT DOWN
The test45 passed!
RETURN COINS
SHUT DOWN
The test46 passed!
DISPOSE SMALL CUP OF TEA
SHUT DOWN
The test47 passed!
DISPOSE LARGE CUP OF TEA
SHUT DOWN
The test48 passed!
RETURN COINS
SHUT DOWN
The test49 passed!
RETURN COINS
SHUT DOWN
The test50 passed!
RETURN COIN
RETURN COINS
SHUT DOWN
The test51 passed!
RETURN COIN
RETURN COIN
RETURN COINS
RETURN COINS
SHUT DOWN
The test52 passed!
RETURN COINS
```

```
SHUT DOWN
The test53 passed!
RETURN COINS
SHUT DOWN
The test54 passed!
RETURN COINS
SHUT DOWN
The test55 passed!

===============================================
Command line suite
Total tests run: 55, Failures: 0, Skips: 0
===============================================
```

Problem #5 Conclusions:

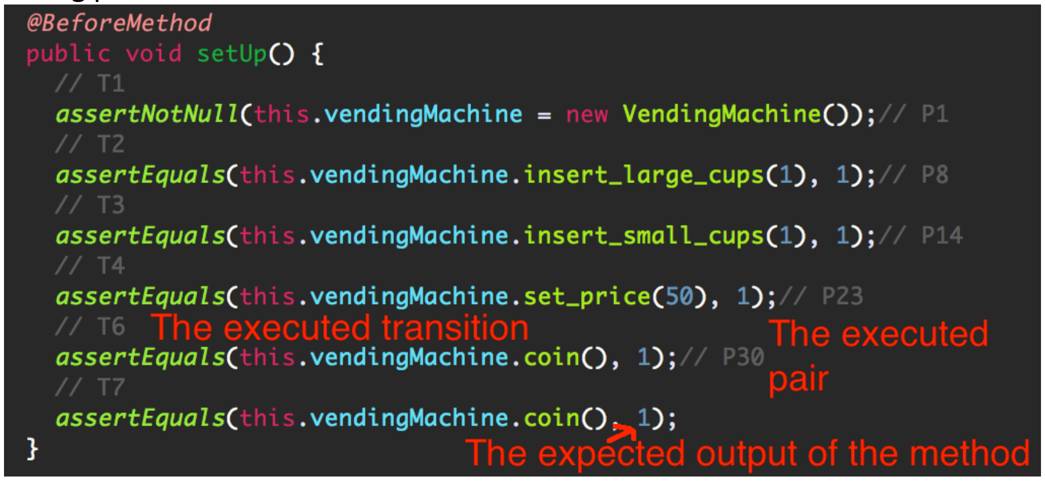
To implement the test environment, I have used the test framework TestNG to make all tests automated. With that it’s possible to executed more easily. It also prove that the test passed because using assertEquals(test1, test2) we can prove that a test is passed or not. If test1 == test2 the test is passed, else the test is failed. For example, assertEquals(this.vendingMachine.insert_large_cups(1), 1; proves that insert_large_cups(1) was executed and return 1 if the test passed, else it return 0.

Each test class is principally composed of those elements:



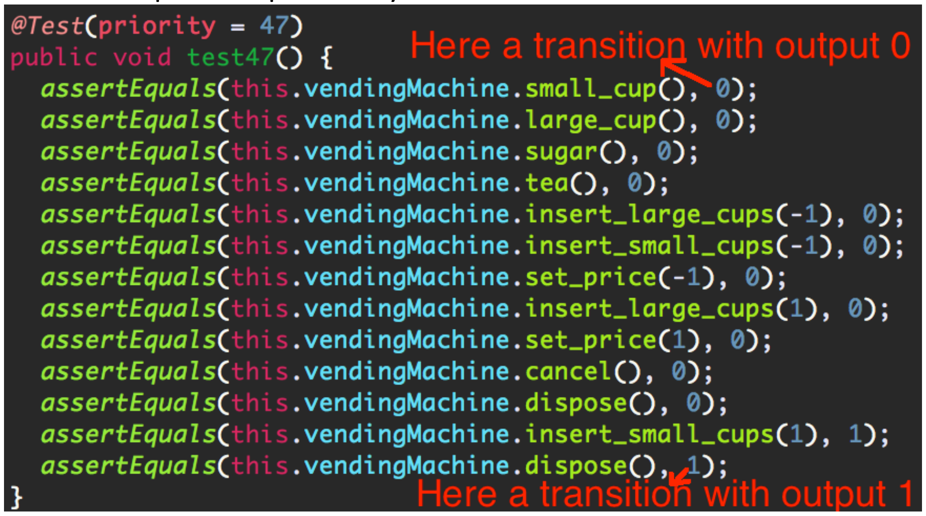
```
package iit.test.valentinpichavant.tests; The package

import static org.testng.Assert.assertEquals; The required imports

/**
 * The Class NoSmallCupsGhostTransitionTesting. The javadoc
 */
public class NoSmallCupsGhostTransitionTesting { The test class name

  /** The vending machine. */
  VendingMachine vendingMachine; The vending machine

  /**
   * Before class.
   */                             The method executed before
  @BeforeClass
  public void beforeClass() {} doing anything in the class

  /**
   * Executed before each method be directly after reaching state No Small Cups.
   */
  @BeforeMethod             The method executed before
  public void setUp() {} doing each test

  /**
   * Test 47.
   */
  @Test(priority = 47) The test priority to order the execution (here 47)
  public void test47() {}The test method, here the test 47
  /**
   * Tear down.
   *
   * @param result the result
   */                                            The method executed after
  @AfterMethod
  public void tearDown(ITestResult result) { doing each test

  /**
   * After class.
   */                          The method executed after
  @AfterClass
  public void afterClass() {} doing anything in the class

}
```

The setUp method is executed before each test method, this allow to avoid redundancy in the test methods, here we proceed the set up required to be in the Coins Inserted state with a price of 50, one large cup and one small cup in the vending machine. The transition executed and the corresponding pair also.



```
@BeforeMethod
public void setUp() {
  // T1
  assertNotNull(this.vendingMachine = new VendingMachine());// P1
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P8
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P14
  // T4
  assertEquals(this.vendingMachine.set_price(50), 1);// P23
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P30
  // T7
  assertEquals(this.vendingMachine.coin(), 1);
}
```

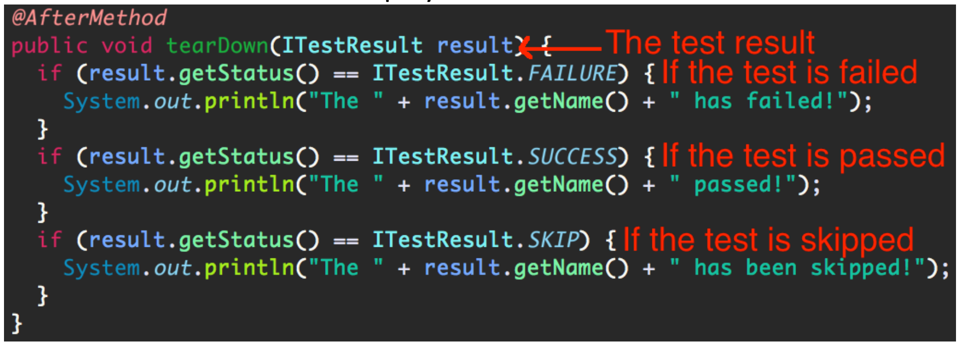
This is a sample of automated test. We can notice that the first method is small_cup because all the methods in the setUp where previously executed.



```
@Test(priority = 47)
public void test47() {
  assertEquals(this.vendingMachine.small_cup(), 0);
  assertEquals(this.vendingMachine.large_cup(), 0);
  assertEquals(this.vendingMachine.sugar(), 0);
  assertEquals(this.vendingMachine.tea(), 0);
  assertEquals(this.vendingMachine.insert_large_cups(-1), 0);
  assertEquals(this.vendingMachine.insert_small_cups(-1), 0);
  assertEquals(this.vendingMachine.set_price(-1), 0);
  assertEquals(this.vendingMachine.insert_large_cups(1), 0);
  assertEquals(this.vendingMachine.set_price(1), 0);
  assertEquals(this.vendingMachine.cancel(), 0);
  assertEquals(this.vendingMachine.dispose(), 0);
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

Here a transition with output 0
Here a transition with output 1

Here we have the teardown method which is called after each test, we can analyze the test result for each test. With that we can display on the screen the result of each test.



```
@AfterMethod
public void tearDown(ITestResult result) {
  if (result.getStatus() == ITestResult.FAILURE) {
    System.out.println("The " + result.getName() + " has failed!");
  }
  if (result.getStatus() == ITestResult.SUCCESS) {
    System.out.println("The " + result.getName() + " passed!");
  }
  if (result.getStatus() == ITestResult.SKIP) {
    System.out.println("The " + result.getName() + " has been skipped!");
  }
}
```

The tests result could be:
Failure if output from the method called is different from the expected output.
Success if output from the method called is equals from the expected output.
Skip if the test has neither fail neither succeeded, this could be caused by an infinite loop or an exception. That why there is a difference between that and the TS.txt, we can say the expected output in this version.
In the test program, neither beforeClass nor afterClass where implemented.

To create the test driver, I have created three classes:
ApplicationCore.java
This class will wrap the VendingMachine class into a new class with methods created to analyze the VendingMachine class. Most of the methods consist of calling the vending machine method over a vending machine object. As it was required to not change the class VendingMachine.java source code, I had to create a method to display the private attribute of the class without modify the class, for that I have used Java reflexivity to modify only for display the privacy of the attributes. First, we retrieve all the fields with getPrivateFields then for each field, we make them accessible with field.setAccessible(true), we display the value and finally we get back the field non-accessible.

JavaTestDriverApplication.java
This goal of this class is to create the application, create the panel in which the items will be displayed, and the multiple interaction with the user will be done.
They are 6 methods in this class:

The method main:
This method is responsible for launching the application.

The constructor:
This is responsible for launching the initialization of the application.

The method initialize:
This method is responsible for initializing the application, we have several part in it:
1. Create the frame and configure it
2. Create a panel inside the frame, configure it and add it to the frame
3. Create multiple buttons, for each
    a. We create a new button
    b. We add the action when the button is pressed
    c. We define his position
    d. And we add the button to the panel
4. We initialize the the applicationCore attribute.

The method launchTests:
This method is responsible for creating a list of all the test classes who need to be executed.

The method launchTestsThread:
This method is responsible for creating a Thread responsible for executing all the test classes who need to be executed.

The method launchTestsThread:
This method is responsible for creating a Thread responsible for executing all the test classes who need to be executed.

The method doTest:
This method is for launching a test class using the framework TestNG.

ExecuteTS.java

This goal of this class is to parse and execute the TS.txt file.
They are 2 methods in this class:

The constructor:

When you create a new ExecuteTS, this will try to open the TS.txt file if it's available in the same folder, if the file is found, it will decompose each line of the TS.txt in a List of lines and then call the executeTestSuite method to do the tests.

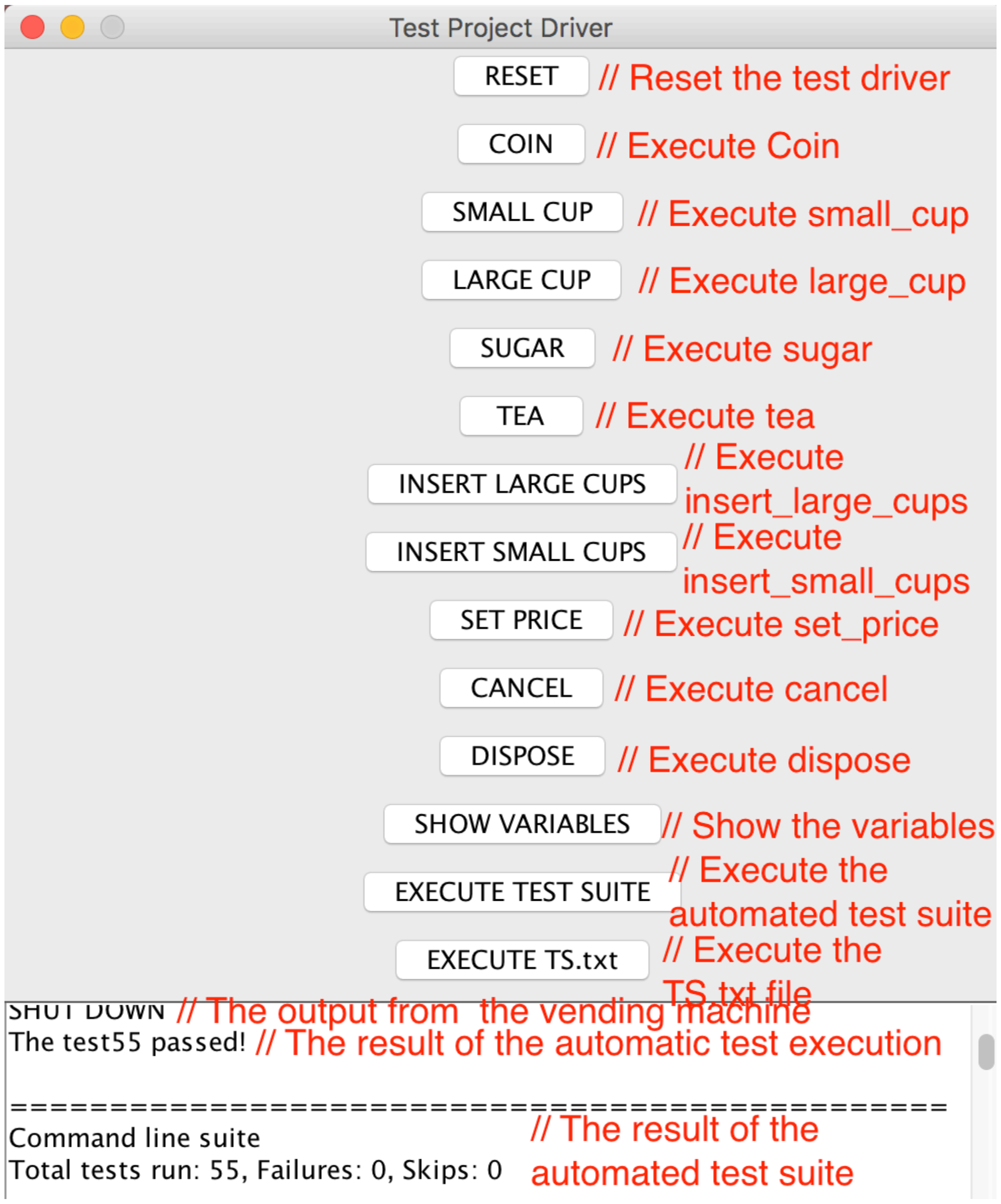
The method executeTestSuite:

This method is responsible for executing each test find in the TS.txt, for each line which contains a test, this will try to execute the test by calling each method on a vending machine created for the test. This will display on screen the output produced. If there is a bug in the test, the test will fail.

The source code is documented for more information. For the transition-pair testing, each transition/pair executed is written as comment in the source code.
During the test, I find several possible bugs:

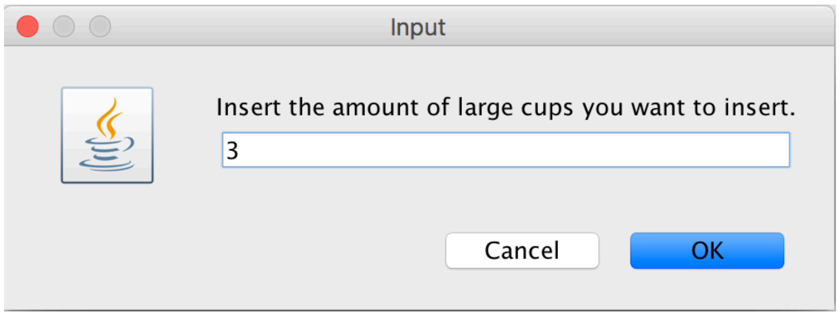
- The variables names should be more explicit like price.
- We have some useless tests, for instance, in we test if x == 2 || x == 3 and after we test again if test x == 2 or x == 3.
- In tea we have many redundant test, for instance in each test we test the current state, and the #9 if, we have a missing else to follow the else/if structure.
- The state machine created by the specification, in the source, we have only display of what need to be done while "return coin" is executed.
- We can go to the state "coins inserted" without having inserted any cup, this will force the user to cancel, it would be more logical to block the entry into this state by addind that at least one cup is inserted into the vending machine.

This is the main view of the Test Project Driver:


Test Project Driver
RESET // Reset the test driver
COIN // Execute Coin
SMALL CUP // Execute small_cup
LARGE CUP // Execute large_cup
SUGAR // Execute sugar
TEA // Execute tea
INSERT LARGE CUPS // Execute insert_large_cups
INSERT SMALL CUPS // Execute insert_small_cups
SET PRICE // Execute set_price
CANCEL // Execute cancel
DISPOSE // Execute dispose
SHOW VARIABLES // Show the variables
EXECUTE TEST SUITE // Execute the automated test suite
EXECUTE TS.txt // Execute the TS.txt file

```
SHUT DOWN // The output from  the vending machine
The test55 passed! // The result of the automatic test execution

===============================================
Command line suite                    // The result of the
Total tests run: 55, Failures: 0, Skips: 0   automated test suite
```

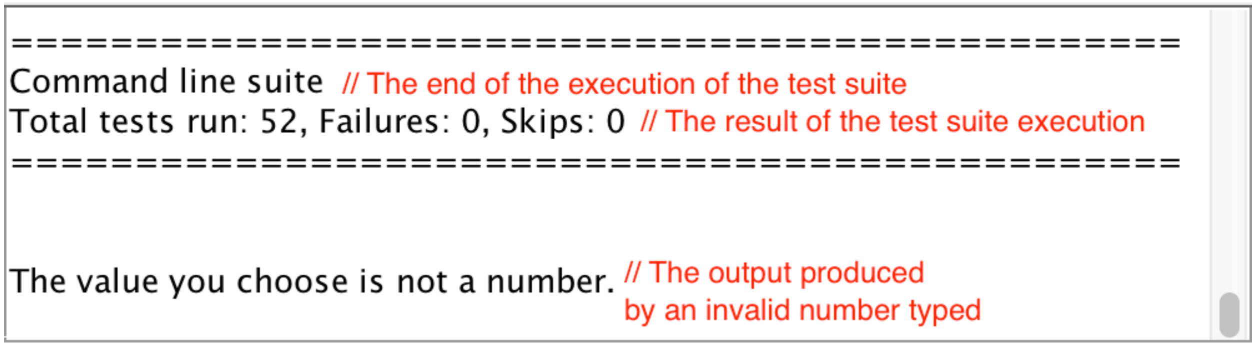
This is the input displayed after a click on the insert large cups button:



Insert the amount of large cups you want to insert.

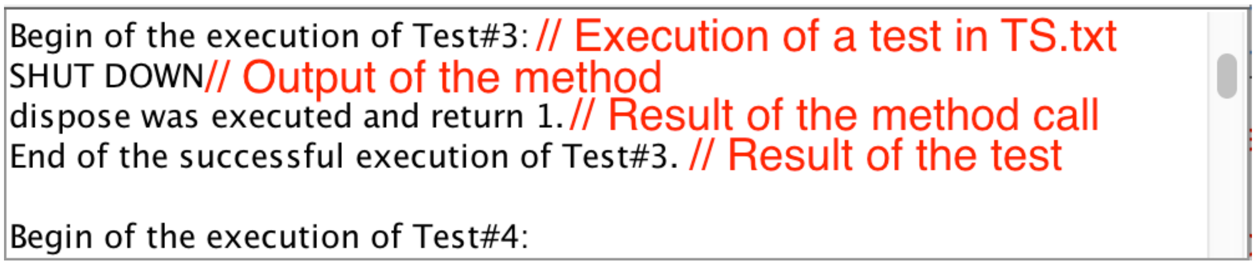
3

This is the output produced at the end of the execution of the test suite and what it's displayed if the output chosen in the previous input was not a number or cancelled.



```
===============================================
Command line suite  // The end of the execution of the test suite
Total tests run: 52, Failures: 0, Skips: 0  // The result of the test suite execution
===============================================


The value you choose is not a number. // The output produced
                                      by an invalid number typed
```

We have the same for the execution of the non-automated test suite with the TS.txt:



```
Begin of the execution of Test#3: // Execution of a test in TS.txt
SHUT DOWN// Output of the method
dispose was executed and return 1.// Result of the method call
End of the successful execution of Test#3. // Result of the test

Begin of the execution of Test#4:
```

Problem #6:

ApplicationCore.java:

```
package iit.test.valentinpichavant;

import java.lang.reflect.Field;
import java.lang.reflect.Modifier;
import java.util.ArrayList;
import java.util.List;

/**
 * The Class ApplicationCore which contains the vending machin for the test driver.
 */
public class ApplicationCore {

 /** The vending machine. */
 VendingMachine vendingMachine;

 /**
  * Instantiates a new application core.
  */
 public ApplicationCore() {
  super();
  this.vendingMachine = new VendingMachine();
 }

 /**
  * Reset the vending machine.
  */
 public void reset() {
  this.vendingMachine = new VendingMachine();
 }

 /**
  * Coin.
  *
  * @return the result of coin
  */
 public int coin() {
  return this.vendingMachine.coin();
 }
```

```
/**
 * Small cup.
 *
 * @return the result of  small cup
 */
public int smallCup() {
  return this.vendingMachine.small_cup();
}

/**
 * Large cup.
 *
 * @return the result of large cup
 */
public int largeCup() {
  return this.vendingMachine.large_cup();
}

/**
 * Sugar.
 *
 * @return the result of sugar
 */
public int sugar() {
  return this.vendingMachine.sugar();
}

/**
 * Tea.
 *
 * @return the result of tea
 */
public int tea() {
  return this.vendingMachine.tea();
}

/**
 * Insert large cups.
 *
 * @param amount the amount of large cups to be added
 * @return the result of insert large cups
 */
public int insertLargeCups(int amount) {
  return this.vendingMachine.insert_large_cups(amount);
}

/**
 * Insert small cups.
 *
 * @param amount the amount of small cups to be added
 * @return the result of insert small cups
 */
public int insertSmallCups(int amount) {
  return this.vendingMachine.insert_small_cups(amount);
}

/**
 * Sets the price.
 *
 * @param price the price
 * @return the result of set price
 */
public int setPrice(int price) {
  return this.vendingMachine.set_price(price);
}

/**
 * Cancel.
 *
 * @return the result of cancel
 */
public int cancel() {
  return this.vendingMachine.cancel();
}

/**
 * Dispose.
 *
 * @return the result of dispose
 */
public int dispose() {
  return this.vendingMachine.dispose();
}

/**
 * Show variables.
 *
 * @return the string contains all of the private variables
 */
public String showVariables() {
  final StringBuilder sb = new StringBuilder();
  final List<Field> fields = getPrivateFields(VendingMachine.class);
  for (final Field field : fields) {
    field.setAccessible(true);
    switch (field.getType().toString()) {
      case "int":
        try {
          sb.append("The attribute: " + field.getName() + " is type " + field.getType()
              + " with a value of " + field.getInt(this.vendingMachine));
          sb.append("\n");
        } catch (final IllegalArgumentException e) {
          e.printStackTrace();
        } catch (final IllegalAccessException e) {
          e.printStackTrace();
        }
        break;
      default:
        try {
          sb.append("The field: " + field.getName() + " is type " + field.getType());
          sb.append("\n");
        } catch (final IllegalArgumentException e) {
          e.printStackTrace();
        }
        break;
    }
    field.setAccessible(false);
  }
  return sb.toString();
}

/**
 * Gets the private fields of a class.
 *
 * @param theClass the class to be analysed
 * @return the private fields
 */
private static List<Field> getPrivateFields(Class<?> theClass) {
  final List<Field> privateFields = new ArrayList<>();
  final Field[] fields = theClass.getDeclaredFields();
  for (final Field field : fields) {
    if (Modifier.isPrivate(field.getModifiers())) {
      privateFields.add(field);
      }
    }
    return privateFields;
  }

}
```

JavaTestDriverApplication.java:

```
package iit.test.valentinpichavant;

import org.testng.TestNG;
import org.testng.annotations.Test;

import java.awt.EventQueue;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextPane;
import javax.swing.ScrollPaneConstants;

import iit.test.valentinpichavant.tests.AdditionalBranchTesting;
import iit.test.valentinpichavant.tests.AdditionalTesting;
import iit.test.valentinpichavant.tests.CoinsInsertedGhostTransitionTesting;
import iit.test.valentinpichavant.tests.CoinsInsertedTesting;
import iit.test.valentinpichavant.tests.IdleGhostTransitionTesting;
import iit.test.valentinpichavant.tests.IdleTesting;
import iit.test.valentinpichavant.tests.NoLargeCupsGhostTransitionTesting;
import iit.test.valentinpichavant.tests.NoLargeCupsTesting;
import iit.test.valentinpichavant.tests.NoSmallCupsGhostTransitionTesting;
import iit.test.valentinpichavant.tests.NoSmallCupsTesting;
import iit.test.valentinpichavant.tests.SugarGhostTransitionTesting;
import iit.test.valentinpichavant.tests.SugarTesting;

/**
 * The Class JavaTestDriverApplication which is responsible for creating the application.
 */
```

```
public class JavaTestDriverApplication {

  /** The frame of the test project driver. */
  private JFrame frmTestProjectDriver;

  /** The application core. */
  private ApplicationCore applicationCore;

  /** The text field containing the results of the outputs. */
  private final JTextPane txtpnText = new JTextPane();

  /**
   * The main method.
   *
   * @param args the arguments
   */
  public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
      @Override
      public void run() {
        try {
          final JavaTestDriverApplication window = new JavaTestDriverApplication();
          window.frmTestProjectDriver.setVisible(true);
        } catch (final Exception e) {
          e.printStackTrace();
        }
      }
    });
  }

  /**
   * Instantiates a new java test driver application.
   */
  public JavaTestDriverApplication() {
    this.initialize();
  }

  /**
   * Initialize the test driver application.
   */
  private void initialize() {
    //Create the frame
    this.frmTestProjectDriver = new JFrame();
    this.frmTestProjectDriver.setResizable(false);
    this.frmTestProjectDriver.setTitle("Test Project Driver ");
    this.frmTestProjectDriver.setBounds(0, 0, 500, 600);
    this.frmTestProjectDriver.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.frmTestProjectDriver.getContentPane()
        .setLayout(new BoxLayout(this.frmTestProjectDriver.getContentPane(),
BoxLayout.X_AXIS));

    //Create the panel to be added on the frame
    final JPanel panel = new JPanel();
    this.frmTestProjectDriver.getContentPane().add(panel);
    //Create the layout of the panel
    final GridBagLayout gbl_panel = new GridBagLayout();
    gbl_panel.columnWidths = new int[] {150, 150, 150};
    gbl_panel.rowHeights = new int[] {30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 80};
    gbl_panel.columnWeights = new double[] {1.0, 0.0, 0.0};
    gbl_panel.rowWeights =
        new double[] {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0};
    panel.setLayout(gbl_panel);

    //Create and add the button reset which is responsible for reseting the Application Core on
click
    final JButton btnReset = new JButton("RESET");
    btnReset.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        JavaTestDriverApplication.this.applicationCore = new ApplicationCore();
        JavaTestDriverApplication.this.txtpnText.setText("");
      }
    });
    final GridBagConstraints gbc_btnReset = new GridBagConstraints();
    gbc_btnReset.insets = new Insets(0, 0, 5, 5);
    gbc_btnReset.gridx = 1;
    gbc_btnReset.gridy = 0;
    panel.add(btnReset, gbc_btnReset);

    //Create and add the button COIN to call the coin method over the applicationCore
    final JButton btnCoin = new JButton("COIN");
    btnCoin.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        JavaTestDriverApplication.this.txtpnText.setText(
            JavaTestDriverApplication.this.txtpnText.getText() + "The  result of call of coin is "
                + JavaTestDriverApplication.this.applicationCore.coin() + ".\n");
      }
    });
    final GridBagConstraints gbc_btnCoin = new GridBagConstraints();
    gbc_btnCoin.insets = new Insets(0, 0, 5, 5);
    gbc_btnCoin.gridx = 1;
    gbc_btnCoin.gridy = 1;
    panel.add(btnCoin, gbc_btnCoin);

    //Create and add the button SMALL CUP to call the smallCup method over the
applicationCore
    final JButton btnSmallCup = new JButton("SMALL CUP");
    btnSmallCup.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        JavaTestDriverApplication.this.txtpnText
            .setText(JavaTestDriverApplication.this.txtpnText.getText()
                + "The result of the call of small_cup is "
                + JavaTestDriverApplication.this.applicationCore.smallCup() + ".\n");
      }
    });
    final GridBagConstraints gbc_btnSmallCup = new GridBagConstraints();
    gbc_btnSmallCup.insets = new Insets(0, 0, 5, 5);
    gbc_btnSmallCup.gridx = 1;
    gbc_btnSmallCup.gridy = 2;
    panel.add(btnSmallCup, gbc_btnSmallCup);

    //Create and add the button LARGE CUP to call the largeCup method over the
applicationCore
    final JButton btnLargeCup = new JButton("LARGE CUP");
    btnLargeCup.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        JavaTestDriverApplication.this.txtpnText
            .setText(JavaTestDriverApplication.this.txtpnText.getText()
                + "The result of the call of large_cup is "
                + JavaTestDriverApplication.this.applicationCore.largeCup() + ".\n");
      }
    });
    final GridBagConstraints gbc_btnLargeCup = new GridBagConstraints();
    gbc_btnLargeCup.insets = new Insets(0, 0, 5, 5);
    gbc_btnLargeCup.gridx = 1;
    gbc_btnLargeCup.gridy = 3;
    panel.add(btnLargeCup, gbc_btnLargeCup);

    //Create and add the button SUGAR to call the sugar method over the applicationCore
```

```
final JButton btnSugar = new JButton("SUGAR");
btnSugar.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
    JavaTestDriverApplication.this.txtpnText
        .setText(JavaTestDriverApplication.this.txtpnText.getText()
            + "The result of the call of sugar is "
            + JavaTestDriverApplication.this.applicationCore.sugar() + ".\n");
  }
});
final GridBagConstraints gbc_btnSugar = new GridBagConstraints();
gbc_btnSugar.insets = new Insets(0, 0, 5, 5);
gbc_btnSugar.gridx = 1;
gbc_btnSugar.gridy = 4;
panel.add(btnSugar, gbc_btnSugar);

//Create and add the button TEA to call the tea method over the applicationCore
final JButton btnTea = new JButton("TEA");
btnTea.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
    JavaTestDriverApplication.this.txtpnText.setText(
        JavaTestDriverApplication.this.txtpnText.getText() + "The result of the call of tea is "
            + JavaTestDriverApplication.this.applicationCore.tea() + ".\n");
  }
});
final GridBagConstraints gbc_btnTea = new GridBagConstraints();
gbc_btnTea.insets = new Insets(0, 0, 5, 5);
gbc_btnTea.gridx = 1;
gbc_btnTea.gridy = 5;
panel.add(btnTea, gbc_btnTea);

//Create and add the button INSERT LARGE CUPS to call the inser_large_cups method over
the applicationCore after asking for the value of cups
final JButton btnInsertLargeCups = new JButton("INSERT LARGE CUPS");
btnInsertLargeCups.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
    final String amountString = JOptionPane.showInputDialog(panel,
        "Insert the amount of large cups you want to insert.", null);
    int amount;
    try {
      amount = Integer.parseInt(amountString);
      JavaTestDriverApplication.this.txtpnText
```

```
          .setText(JavaTestDriverApplication.this.txtpnText.getText()
            + "The result of the call of insert_large_cups with value " + amount + " is "
            + JavaTestDriverApplication.this.applicationCore.insertLargeCups(amount) + ".\n");
      } catch (final NumberFormatException nfe) {
        nfe.printStackTrace();
        JavaTestDriverApplication.this.txtpnText
          .setText(JavaTestDriverApplication.this.txtpnText.getText()
            + "The value you choose is not a number.\n");
      }
    }
  });
  final GridBagConstraints gbc_btnInsertLargeCups = new GridBagConstraints();
  gbc_btnInsertLargeCups.insets = new Insets(0, 0, 5, 5);
  gbc_btnInsertLargeCups.gridx = 1;
  gbc_btnInsertLargeCups.gridy = 6;
  panel.add(btnInsertLargeCups, gbc_btnInsertLargeCups);

  //Create and add the button INSERT SMALL CUPS to call the inser_small_cups method over
the applicationCore after asking for the value of cups
  final JButton btnInsertSmallCups = new JButton("INSERT SMALL CUPS");
  btnInsertSmallCups.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      final String amountString = JOptionPane.showInputDialog(panel,
          "Insert the amount of small cups you want to insert.", null);
      int amount;
      try {
        amount = Integer.parseInt(amountString);
        JavaTestDriverApplication.this.txtpnText
          .setText(JavaTestDriverApplication.this.txtpnText.getText()
            + "The result of the call of insert_small_cups with value " + amount + " is "
            + JavaTestDriverApplication.this.applicationCore.insertSmallCups(amount) + ".\n");
      } catch (final NumberFormatException nfe) {
        nfe.printStackTrace();
        JavaTestDriverApplication.this.txtpnText
          .setText(JavaTestDriverApplication.this.txtpnText.getText()
            + "The value you choose is not a number.\n");
      }
    }
  });
  final GridBagConstraints gbc_btnInsertSmallCups = new GridBagConstraints();
  gbc_btnInsertSmallCups.insets = new Insets(0, 0, 5, 5);
  gbc_btnInsertSmallCups.gridx = 1;
  gbc_btnInsertSmallCups.gridy = 7;
  panel.add(btnInsertSmallCups, gbc_btnInsertSmallCups);

  //Create and add the button SET PRICE to call the set_price method over the applicationCore
after asking for the value of price
  final JButton btnSetPrice = new JButton("SET PRICE");
  btnSetPrice.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      final String priceString =
        JOptionPane.showInputDialog(panel, "Insert the price you want.", null);
      int price;
      try {
        price = Integer.parseInt(priceString);
        JavaTestDriverApplication.this.txtpnText
          .setText(JavaTestDriverApplication.this.txtpnText.getText()
            + "The result of the call of set_price with value " + price + " is "
            + JavaTestDriverApplication.this.applicationCore.setPrice(price) + ".\n");
      } catch (final NumberFormatException nfe) {
        nfe.printStackTrace();
        JavaTestDriverApplication.this.txtpnText
          .setText(JavaTestDriverApplication.this.txtpnText.getText()
            + "The value you choose is not a number.\n");
      }
    }
  });
  final GridBagConstraints gbc_btnSetPrice = new GridBagConstraints();
  gbc_btnSetPrice.insets = new Insets(0, 0, 5, 5);
  gbc_btnSetPrice.gridx = 1;
  gbc_btnSetPrice.gridy = 8;
  panel.add(btnSetPrice, gbc_btnSetPrice);

  //Create and add the button CANCEL to call the cancel method over the applicationCore
  final JButton btnCancel = new JButton("CANCEL");
  btnCancel.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      JavaTestDriverApplication.this.txtpnText
        .setText(JavaTestDriverApplication.this.txtpnText.getText()
          + "The result of the call of cancel is "
          + JavaTestDriverApplication.this.applicationCore.cancel() + ".\n");
    }
  });
  final GridBagConstraints gbc_btnCancel = new GridBagConstraints();
  gbc_btnCancel.insets = new Insets(0, 0, 5, 5);
  gbc_btnCancel.gridx = 1;
  gbc_btnCancel.gridy = 9;
  panel.add(btnCancel, gbc_btnCancel);

  //Create and add the button DISPOSE to call the dispose method over the applicationCore
  final JButton btnDispose = new JButton("DISPOSE");
  btnDispose.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      JavaTestDriverApplication.this.txtpnText
          .setText(JavaTestDriverApplication.this.txtpnText.getText()
              + "The result of the call of dispose is "
              + JavaTestDriverApplication.this.applicationCore.dispose() + ".\n");
    }
  });
  final GridBagConstraints gbc_btnDispose = new GridBagConstraints();
  gbc_btnDispose.insets = new Insets(0, 0, 5, 5);
  gbc_btnDispose.gridx = 1;
  gbc_btnDispose.gridy = 10;
  panel.add(btnDispose, gbc_btnDispose);

  //Create and add the button SHOW VARIABLES to call the show_variables method over the
applicationCore
  final JButton btnShowVariables = new JButton("SHOW VARIABLES");
  btnShowVariables.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      JavaTestDriverApplication.this.txtpnText
          .setText(JavaTestDriverApplication.this.txtpnText.getText()
              + JavaTestDriverApplication.this.applicationCore.showVariables());
    }
  });
  final GridBagConstraints gbc_btnShowVariables = new GridBagConstraints();
  gbc_btnShowVariables.insets = new Insets(0, 0, 5, 5);
  gbc_btnShowVariables.gridx = 1;
  gbc_btnShowVariables.gridy = 11;
  panel.add(btnShowVariables, gbc_btnShowVariables);

  //Create and add the button EXECUTE TEST SUITE to call the method launchTests to launch
the automated tests
  final JButton btnExecuteAutomatedTest = new JButton("EXECUTE TEST SUITE");
  btnExecuteAutomatedTest.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      JavaTestDriverApplication.this.launchTests();
    }
  });
  final GridBagConstraints gbc_btnExecuteAutomatedTest = new GridBagConstraints();
  gbc_btnExecuteAutomatedTest.insets = new Insets(0, 0, 5, 5);
  gbc_btnExecuteAutomatedTest.gridx = 1;
  gbc_btnExecuteAutomatedTest.gridy = 12;
  panel.add(btnExecuteAutomatedTest, gbc_btnExecuteAutomatedTest);

  //Create and add the button EXECUTE TS.txt to be able to execute the TS.txt file in the same
folder
  JButton btnExecuteTstxt = new JButton("EXECUTE TS.txt");
  btnExecuteTstxt.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      //Save the current system output
      final PrintStream origOut = System.out;
      //Create a new outputStream
      final ByteArrayOutputStream baos = new ByteArrayOutputStream();
      final PrintStream out = new PrintStream(baos);
      //Change the java output to the new one to save all the out generated by the tests
execution
      System.setOut(out);
      //Run the tests
      new ExecuteTS();
      //Update the text field
      txtpnText.setText(txtpnText.getText() + new String(baos.toByteArray(),
StandardCharsets.UTF_8) + "\n");
      //Get back to the original system output
      System.setOut(origOut);
    }
  });
  GridBagConstraints gbc_btnExecuteTstxt = new GridBagConstraints();
  gbc_btnExecuteTstxt.insets = new Insets(0, 0, 5, 5);
  gbc_btnExecuteTstxt.gridx = 1;
  gbc_btnExecuteTstxt.gridy = 13;
  panel.add(btnExecuteTstxt, gbc_btnExecuteTstxt);

  //Initialize and add the text field which will display the output of the test driver
  this.txtpnText.setText("");
  this.txtpnText.setEditable(false);
  final GridBagConstraints gbc_txtpnText = new GridBagConstraints();
  gbc_txtpnText.fill = GridBagConstraints.BOTH;
  gbc_txtpnText.insets = new Insets(0, 0, 0, 0);
  gbc_txtpnText.gridwidth = 3;
```

```
  gbc_txtpnText.gridx = 0;
  gbc_txtpnText.gridy = 14;
  final JScrollPane scroll = new JScrollPane(this.txtpnText);
  scroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
  panel.add(scroll, gbc_txtpnText);

  //Create the applicationCore
  this.applicationCore = new ApplicationCore();
 }

 /**
  * Launch the tests.
  */
 private void launchTests() {
  //Create the list containing all tests classes
  final List<Class<?>> testClasses = new ArrayList<>();
  //Adding all tests classes to the list
  testClasses.add(IdleTesting.class);
  testClasses.add(CoinsInsertedTesting.class);
  testClasses.add(SugarTesting.class);
  testClasses.add(NoLargeCupsTesting.class);
  testClasses.add(NoSmallCupsTesting.class);
  testClasses.add(AdditionalTesting.class);
  testClasses.add(IdleGhostTransitionTesting.class);
  testClasses.add(CoinsInsertedGhostTransitionTesting.class);
  testClasses.add(SugarGhostTransitionTesting.class);
  testClasses.add(NoLargeCupsGhostTransitionTesting.class);
  testClasses.add(NoSmallCupsGhostTransitionTesting.class);
  testClasses.add(AdditionalBranchTesting.class);
  //Create a new thread to execute the tests
  this.launchTestsThread(testClasses);
 }

 /**
  * Launch a thread to do testing.
  *
  * @param testClass the test classes list
  */
 private void launchTestsThread(final List<Class<?>> testClass) {
  //Start the thread of tests
  new Thread(new Runnable() {
   @Override
   public void run() {
    JavaTestDriverApplication.this.doTests(testClass);
```

```
     }
    }).start();
  }

  /**
   * Do the tests.
   *
   * @param testClasses the test classes list
   */
  @Test
  private void doTests(List<Class<?>> testClasses) {
    //Create the class needed to execute the tests
    final TestNG testng = new TestNG();
    //Define the tests
    testng.setTestClasses(testClasses.toArray(new Class[testClasses.size()]));
    //Save the current system output
    final PrintStream origOut = System.out;
    //Create a new outputStream
    final ByteArrayOutputStream baos = new ByteArrayOutputStream();
    final PrintStream out = new PrintStream(baos);
    //Change the java output to the new one to save all the out generated by the tests execution
    System.setOut(out);
    //Run the tests
    testng.run();
    //Update the text field
    this.txtpnText.setText(
        this.txtpnText.getText() + new String(baos.toByteArray(), StandardCharsets.UTF_8) + "\n");
    //Get back to the original system output
    System.setOut(origOut);
  }
}
```

VendingMachine.java:

```
package iit.test.valentinpichavant;

/**
 * The Class VendingMachine.
 */
public class VendingMachine
{

   /** The current state. */
   private int x;

   /** The current price. */
   private int price;

   /** The current number of large cups. */
   private int k;

   /** The current number of small cups. */
   private int k1;

   /** The the current value inserted. */
   private int t;

   /** The size of the cup choosen, 1 if large, 2 if small, nothing otherwise. */
   private int s;

   /**
    * Instantiates a new vending machine.
    */
   public VendingMachine()
   {
      k1 = 0;
      k = 0;
      t = 0;
      price = 0;
      x = 1;
   }
```

```
/**
 * Coin.
 *
 * @return the result of coin, 1 if transition executed else 0
 */
public final int coin()
{
   if (x == 1)
   {
      if ((t + 25 >= price) && (price > 0))
      {
         s = 0;
         t = 0;
         x = 2;
         return 1;
      }
      else if (t + 25 < price)
      {
         t = t + 25;
         return 1;
      }
   }
   else if ((x > 1) && (x < 6))
   {
      System.out.print("RETURN COIN");
      System.out.print("\n");
      return 1;
   }
   return 0;
}
```

```
/**
 * Small cup.
 *
 * @return the result of small_cup, 1 if transition executed else 0
 */
public final int small_cup()
{
   if ((x == 2) || (x == 3))
   {
      s = 2;
      return 1;
   }
   return 0;
}

/**
 * Large cup.
 *
 * @return the result of large_cup, 1 if transition executed else 0
 */
public final int large_cup()
{
   if ((x == 2) || (x == 3))
   {
      s = 1;
      return 1;
   }
   return 0;
}
```

```
/**
 * Sugar.
 *
 * @return the result of sugar, 1 if transition executed else 0
 */
public final int sugar()
{
   if ((x == 2) || (x == 3))
   {
      if (x == 2)
      {
         x = 3;
      }
      else
      {
         x = 2;
      }
      return 1;
   }
   return 0;
}
```

```
/**
 * Tea.
 *
 * @return the result of tea, 1 if transition executed else 0
 */
public final int tea()
{
   if ((x == 2) || (x == 3))
   {
      if ((x == 2) && (k1 > 1) && (s == 2))
      {
         System.out.print("DISPOSE SMALL CUP OF TEA");
         System.out.print("\n");
         k1 = k1 - 1;
         x = 1;
         return 1;
      }
      else if ((x == 2) && (k > 1) && (s == 1))
      {
         System.out.print("DISPOSE LARGE CUP OF TEA");
         System.out.print("\n");
         k = k - 1;
         x = 1;
         return 1;
      }
      else if ((x == 2) && (k == 1) && (s == 1))
      {
         System.out.print("DISPOSE LARGE CUP OF TEA");
         System.out.print("\n");
         k = k - 1;
         x = 5;
         return 1;
      }
      else if ((x == 2) && (k1 == 1) && (s == 2))
      {
         System.out.print("DISPOSE SMALL CUP OF TEA");
         System.out.print("\n");
         k1 = k1 - 1;
         x = 4;
         return 1;
      }
      else if ((x == 3) && (k1 == 1) && (s == 2))
      {
         System.out.print("DISPOSE SMALL CUP OF TEA WITH SUGAR");
```

```
            System.out.print("\n");
            k1 = k1 - 1;
            x = 4;
            return 1;
        }
        else if ((x == 3) && (k == 1) && (s == 1))
        {
            System.out.print("DISPOSE LARGE CUP OF TEA WITH SUGAR");
            System.out.print("\n");
            k = k - 1;
            x = 5;
            return 1;
        }
        if ((x == 3) && (k1 > 1) && (s == 2))
        {
            System.out.print("DISPOSE SMALL CUP OF TEA WITH SUGAR");
            System.out.print("\n");
            k1 = k1 - 1;
            x = 1;
            return 1;
        }
        else if ((x == 3) && (k > 1) && (s == 1))
        {
            System.out.print("DISPOSE LARGE CUP OF TEA WITH SUGAR");
            System.out.print("\n");
            k = k - 1;
            x = 1;
            return 1;
        }
        return 0;
    }
    return 0;
}
```

```
/**
 * Insert large cups.
 *
 * @param n the amount of large cups to be added
 * @return the result of insert_large_cups, 1 if transition executed else 0
 */
public final int insert_large_cups(int n)
{
   if ((x == 1) && (n > 0))
   {
      k = k + n;
      return 1;
   }
   else if ((x == 5) && (n > 0))
   {
      k = n;
      x = 1;
      return 1;
   }
   return 0;
}

/**
 * Insert small cups.
 *
 * @param n the amount of small cups to be added
 * @return the result of small_cup, 1 if transition executed else 0
 */
public final int insert_small_cups(int n)
{
   if ((x == 1) && (n > 0))
   {
      k1 = k1 + n;
      return 1;
   }
   else if ((x == 4) && (n > 0))
   {
      k1 = n;
      x = 1;
      return 1;
   }
   return 0;
}
```

```
/**
 * Sets the price.
 *
 * @param p the new price for the vending machine
 * @return the result of set_price, 1 if transition executed else 0
 */
public final int set_price(int p)
{
   if ((x == 1) && (p > 0))
   {
      price = p;
      return 1;
   }
   return 0;
}

/**
 * Cancel.
 *
 * @return the result of cancel, 1 if transition executed else 0
 */
public final int cancel()
{
   if ((x == 2) || (x == 3))
   {
      System.out.print("RETURN COINS");
      System.out.print("\n");
      x = 1;
      return 1;
   }
   return 0;
}
```

```
    /**
     * Dispose.
     *
     * @return the result of dispose, 1 if transition executed else 0
     */
    public final int dispose()
    {
        if ((x == 1))
        {
            System.out.print("SHUT DOWN");
            System.out.print("\n");
            x = 6;
            return 1;
        }
        return 0;
    }
}
```

IdleTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class IdleTesting.
 */
public class IdleTesting {

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Test 1.
   */
  @Test(priority = 1)
  public void test1() {
    VendingMachine vendingMachine;
    // T1
    assertNotNull(vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(vendingMachine.insert_large_cups(2), 1);// P8
    // T3
    assertEquals(vendingMachine.insert_small_cups(3), 1);// P14
    // T3
    assertEquals(vendingMachine.insert_small_cups(3), 1);// P15
    // T4
    assertEquals(vendingMachine.set_price(25), 1);// P19
    // T2
    assertEquals(vendingMachine.insert_large_cups(2), 1);// P7
  // T2
  assertEquals(vendingMachine.insert_large_cups(2), 1);// P9
  // T4
  assertEquals(vendingMachine.set_price(25), 1);// P22
  // T5
  assertEquals(vendingMachine.dispose(), 1);
 }

 /**
  * Test 2.
  */
 @Test(priority = 2)
 public void test2() {
  VendingMachine vendingMachine;
  // T1
  assertNotNull(vendingMachine = new VendingMachine());// P2
  // T3
  assertEquals(vendingMachine.insert_small_cups(3), 1);// P15
  // T4
  assertEquals(vendingMachine.set_price(100), 1);// P23
  // T6
  assertEquals(vendingMachine.coin(), 1);// P28
  // T5
  assertEquals(vendingMachine.dispose(), 1);
 }

 /**
  * Test 3.
  */
 @Test(priority = 3)
 public void test3() {
  VendingMachine vendingMachine;
  // T1
  assertNotNull(vendingMachine = new VendingMachine());// P4
  // T5
  assertEquals(vendingMachine.dispose(), 1);
 }

 /**
  * Test 4.
  */
 @Test(priority = 4)
 public void test4() {
  VendingMachine vendingMachine;
  // T1
  assertNotNull(vendingMachine = new VendingMachine());// P3
  // T4
  assertEquals(vendingMachine.set_price(100), 1);// P21
  // T4
  assertEquals(vendingMachine.set_price(75), 1);// P20
  // T3
  assertEquals(vendingMachine.insert_small_cups(3), 1);// P13
  // T2
  assertEquals(vendingMachine.insert_large_cups(2), 1); // P11
  // T6
  assertEquals(vendingMachine.coin(), 1);// P26
  // T3
  assertEquals(vendingMachine.insert_small_cups(3), 1);// P17
  // T6
  assertEquals(vendingMachine.coin(), 1);// P27
  // T4
  assertEquals(vendingMachine.set_price(100), 1);// P19
  // T2
  assertEquals(vendingMachine.insert_large_cups(2), 1); // P10
  // T5
  assertEquals(vendingMachine.dispose(), 1);
 }

 /**
  * Test 5.
  */
 @Test(priority = 5)
 public void test5() {
  VendingMachine vendingMachine;
  // T1
  assertNotNull(vendingMachine = new VendingMachine());// P1
  // T2
  assertEquals(vendingMachine.insert_large_cups(2), 1); // P8
  // T3
  assertEquals(vendingMachine.insert_small_cups(3), 1);// P16
  // T5
  assertEquals(vendingMachine.dispose(), 1);
 }
 /**
  * Test 6.
  */
 @Test(priority = 6)
 public void test6() {
  VendingMachine vendingMachine;
  // T1
  assertNotNull(vendingMachine = new VendingMachine());// P3
  // T4
  assertEquals(vendingMachine.set_price(100), 1);// P23
  // T6
  assertEquals(vendingMachine.coin(), 1);// P29
  // T6
  assertEquals(vendingMachine.coin(), 1);// P25
  // T2
  assertEquals(vendingMachine.insert_large_cups(2), 1); // P10
  // T5
  assertEquals(vendingMachine.dispose(), 1);
 }

 /**
  * Tear down.
  *
  * @param result the result
  */
 @AfterMethod
 public void tearDown(ITestResult result) {
  if (result.getStatus() == ITestResult.FAILURE) {
   System.out.println("The " + result.getName() + " has failed!");
  }
  if (result.getStatus() == ITestResult.SUCCESS) {
   System.out.println("The " + result.getName() + " passed!");
  }
  if (result.getStatus() == ITestResult.SKIP) {
   System.out.println("The " + result.getName() + " has been skipped!");
  }
 }

 /**
  * After class.
  */
 @AfterClass
 public void afterClass() {}
}
```

CoinsInsertedTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class CoinsInsertedTesting.
 */
public class CoinsInsertedTesting {

  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Executed before each method be directly after reaching state Coin Inserted.
   */
  @BeforeMethod
  public void setUp() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P8
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P14
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);// P23
    // T6
  assertEquals(this.vendingMachine.coin(), 1);// P30
  // T7
  assertEquals(this.vendingMachine.coin(), 1);
 }

 /**
  * Test 7.
  */
 @Test(priority = 7)
 public void test7() {
  // T7 is the last transition due to the setUp()
  // P79
  // T10
  assertEquals(this.vendingMachine.cancel(), 1);// P46
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }

 /**
  * Test 8.
  */
 @Test(priority = 8)
 public void test8() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P89
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P88
  // T11
  assertEquals(this.vendingMachine.tea(), 1);// P54
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P30
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P93
  // T24
  assertEquals(this.vendingMachine.tea(), 1);// P161
  // T8
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P34
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }
```

```
/**
 * Test 9.
 */
@Test(priority = 9)
public void test9() {
 // T7 is the last transition due to the setUp()
 // P82
 // T20
 assertEquals(this.vendingMachine.coin(), 1);// P98
 // T20
 assertEquals(this.vendingMachine.coin(), 1);// P99
 // T21
 assertEquals(this.vendingMachine.small_cup(), 1);// P108
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P152
 // T23
 assertEquals(this.vendingMachine.sugar(), 1);// P114
 // T20
 assertEquals(this.vendingMachine.coin(), 1);// P100
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P152
 // T23
 assertEquals(this.vendingMachine.sugar(), 1);// P115
 // T21
 assertEquals(this.vendingMachine.small_cup(), 1);// P108
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P152
 // T23
 assertEquals(this.vendingMachine.sugar(), 1);// P113
 // T19
 assertEquals(this.vendingMachine.large_cup(), 1);// P92
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P152
 // T23
 assertEquals(this.vendingMachine.sugar(), 1);// P116
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P152
 // T23
 assertEquals(this.vendingMachine.sugar(), 1);// P171
 // T12
 assertEquals(this.vendingMachine.tea(), 1);// P58
 // T5
 assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
/**
 * Test 10.
 */
@Test(priority = 10)
public void test10() {
  // T7 is the last transition due to the setUp()
  // P83
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P107
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P108
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P152
  // T23
  assertEquals(this.vendingMachine.sugar(), 1);// P111
  // T10
  assertEquals(this.vendingMachine.cancel(), 1);// P47
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P30
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P150
  // T17
  assertEquals(this.vendingMachine.small_cup(), 1);// P134
  // T23
  assertEquals(this.vendingMachine.sugar(), 1);// P112
  // T11
  assertEquals(this.vendingMachine.tea(), 1);// P52
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 11.
 */
@Test(priority = 11)
public void test11() {
  // T7 is the last transition due to the setUp()
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P106
  // T20
  assertEquals(this.vendingMachine.coin(), 1);// P97
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P91
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P105
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P90
  // T20
  assertEquals(this.vendingMachine.coin(), 1);// P95
  // T10
  assertEquals(this.vendingMachine.cancel(), 1);// P45
  // T4
  assertEquals(this.vendingMachine.set_price(25), 1);// P24
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P91
  // T10
  assertEquals(this.vendingMachine.cancel(), 1);// P44
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P16
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Tear down.
 *
 * @param result the result
 */
@AfterMethod
public void tearDown(ITestResult result) {
  if (result.getStatus() == ITestResult.FAILURE) {
    System.out.println("The " + result.getName() + " has failed!");
  }
  if (result.getStatus() == ITestResult.SUCCESS) {
    System.out.println("The " + result.getName() + " passed!");
  }
  if (result.getStatus() == ITestResult.SKIP) {
    System.out.println("The " + result.getName() + " has been skipped!");
  }
}
```

```
  /**
   * After class.
   */
  @AfterClass
  public void afterClass() {}
}
```

SugarTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class SugarTesting.
 */
public class SugarTesting {


  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Executed before each method be directly after reaching state Coin Inserted.
   */
  @BeforeMethod
  public void setUp() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P8
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P14
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);// P23
    // T6
    assertEquals(this.vendingMachine.coin(), 1);// P30
    // T7
    assertEquals(this.vendingMachine.coin(), 1);
  }

  /**
   * Test 12.
   */
  @Test(priority = 12)
  public void test12() {
    // T7 is the last transition due to the setUp()
    // P83
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);// P108
    // T22
    assertEquals(this.vendingMachine.sugar(), 1);// P148
    // T15
    assertEquals(this.vendingMachine.tea(), 1);// P76
    // T5
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Test 13.
   */
  @Test(priority = 13)
  public void test13() {
    // T7 is the last transition due to the setUp()
    // P83
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);// P108
    // T22
    assertEquals(this.vendingMachine.sugar(), 1);// P149
    // T16
    assertEquals(this.vendingMachine.coin(), 1);// P122
    // T16
    assertEquals(this.vendingMachine.coin(), 1);// P121
    // T15
    assertEquals(this.vendingMachine.tea(), 1);// P73
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P10
    // T5
    assertEquals(this.vendingMachine.dispose(), 1);
```

```
}

/**
 * Test 14.
 */
@Test(priority = 14)
public void test14() {
 // T7 is the last transition due to the setUp()
 // P83
 // T21
 assertEquals(this.vendingMachine.small_cup(), 1);// P108
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P150
 // T17
 assertEquals(this.vendingMachine.small_cup(), 1);// P132
 // T17
 assertEquals(this.vendingMachine.small_cup(), 1);// P130
 // T15
 assertEquals(this.vendingMachine.tea(), 1);// P74
 // T3
 assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P16
 // T5
 assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 15.
 */
@Test(priority = 15)
public void test15() {
 // T7 is the last transition due to the setUp()
 // P83
 // T21
 assertEquals(this.vendingMachine.small_cup(), 1);// P108
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P151
 // T18
 assertEquals(this.vendingMachine.large_cup(), 1);// P142
 // T18
 assertEquals(this.vendingMachine.large_cup(), 1);// P141
 // T17
 assertEquals(this.vendingMachine.small_cup(), 1);// P130
 // T15
 assertEquals(this.vendingMachine.tea(), 1);// P77
```

```
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P28
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }

 /**
  * Test 16.
  */
 @Test(priority = 16)
 public void test16() {
  // T7 is the last transition due to the setUp()
  // P83
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P108
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P138
  // T15
  assertEquals(this.vendingMachine.tea(), 1);// P75
  // T4
  assertEquals(this.vendingMachine.set_price(25), 1);// P22
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }

 /**
  * Test 17.
  */
 @Test(priority = 17)
 public void test17() {
  // T7 is the last transition due to the setUp()
  // P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P147
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P60
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }
```

```
/**
 * Test 18.
 */
@Test(priority = 18)
public void test18() {
  // T7 is the last transition due to the setUp()
  // P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P147
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P67
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P10
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 19.
 */
@Test(priority = 19)
public void test19() {
  // T7 is the last transition due to the setUp()
  // P
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P147
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P68
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P16
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 20.
 */
@Test(priority = 20)
public void test20() {
  // T7 is the last transition due to the setUp()
  // P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P147
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P71
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P28
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 21.
 */
@Test(priority = 21)
public void test21() {
  // T7 is the last transition due to the setUp()
  // P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P147
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P69
  // T4
  assertEquals(this.vendingMachine.set_price(25), 1);// P22
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 22.
 */
@Test(priority = 22)
public void test22() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P92
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P153
  // T13
  assertEquals(this.vendingMachine.tea(), 1);// P64
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
/**
 * Test 23.
 */
@Test(priority = 23)
public void test23() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P92
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P146
  // T13
  assertEquals(this.vendingMachine.tea(), 1);// P61
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P10
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 24.
 */
@Test(priority = 24)
public void test24() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P92
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P146
  // T13
  assertEquals(this.vendingMachine.tea(), 1);// P62
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P16
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
/**
 * Test 25.
 */
@Test(priority = 25)
public void test25() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P92
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P146
  // T13
  assertEquals(this.vendingMachine.tea(), 1);// P63
  // T4
  assertEquals(this.vendingMachine.set_price(25), 1);// P22
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 26.
 */
@Test(priority = 26)
public void test26() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P92
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P146
  // T13
  assertEquals(this.vendingMachine.tea(), 1);// P65
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P28
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
/**
 * Test 27.
 */
@Test(priority = 27)
public void test27() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P92
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P149
  // T16
  assertEquals(this.vendingMachine.coin(), 1);// P123
  // T17
  assertEquals(this.vendingMachine.small_cup(), 1);// P131
  // T16
  assertEquals(this.vendingMachine.coin(), 1);// P124
  // T18
  assertEquals(this.vendingMachine.large_cup(), 1);// P140
  // T16
  assertEquals(this.vendingMachine.coin(), 1);// P125
  // T23
  assertEquals(this.vendingMachine.sugar(), 1);// P116
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P149
  // T16
  assertEquals(this.vendingMachine.coin(), 1);// P120
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P60
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
  /**
   * Tear down.
   *
   * @param result the result
   */
  @AfterMethod
  public void tearDown(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
      System.out.println("The " + result.getName() + " has failed!");
    }
    if (result.getStatus() == ITestResult.SUCCESS) {
      System.out.println("The " + result.getName() + " passed!");
    }
    if (result.getStatus() == ITestResult.SKIP) {
      System.out.println("The " + result.getName() + " has been skipped!");
    }
  }

  /**
   * After class.
   */
  @AfterClass
  public void afterClass() {}

}
```

NoSmallCupsTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class NoSmallCupsTesting.
 */
public class NoSmallCupsTesting {


  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Executed before each method be directly after reaching state Coin Inserted.
   */
  @BeforeMethod
  public void setUp() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P8
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P14
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);// P23
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P30
  // T7
  assertEquals(this.vendingMachine.coin(), 1);
 }

 /**
  * Test 32.
  */
 @Test(priority = 32)
 public void test32() {
  // T7 is the last transition due to the setUp()
  // P83
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P110
  // T25
  assertEquals(this.vendingMachine.tea(), 1);// P155
  // T9
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P40
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }

 /**
  * Test 33.
  */
 @Test(priority = 33)
 public void test33() {
  // T7 is the last transition due to the setUp()
  // P83
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P110
  // T25
  assertEquals(this.vendingMachine.tea(), 1);// P156
  // T28
  assertEquals(this.vendingMachine.coin(), 1);// P160
  // T28
  assertEquals(this.vendingMachine.coin(), 1);// P159
  // T9
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P37
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P10
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
```

```
}

/**
 * Test 34.
 */
@Test(priority = 34)
public void test34() {
  // T7 is the last transition due to the setUp()
  // P83
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P107
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P154
  // T27
  assertEquals(this.vendingMachine.tea(), 1);// P157
  // T9
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P38
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P16
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}


/**
 * Test 35.
 */
@Test(priority = 35)
public void test35() {
  // T7 is the last transition due to the setUp()
  // P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P150
  // T17
  assertEquals(this.vendingMachine.small_cup(), 1);// P136
  // T27
  assertEquals(this.vendingMachine.tea(), 1);// P158
  // T28
  assertEquals(this.vendingMachine.coin(), 1);// P160
  // T28
  assertEquals(this.vendingMachine.coin(), 1);// P159
  // T9
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P39
```

```
    // T4
    assertEquals(this.vendingMachine.set_price(25), 1);// P22
    // T5
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Tear down.
   *
   * @param result the result
   */
  @AfterMethod
  public void tearDown(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
      System.out.println("The " + result.getName() + " has failed!");
    }
    if (result.getStatus() == ITestResult.SUCCESS) {
      System.out.println("The " + result.getName() + " passed!");
    }
    if (result.getStatus() == ITestResult.SKIP) {
      System.out.println("The " + result.getName() + " has been skipped!");
    }
  }

  /**
   * After class.
   */
  @AfterClass
  public void afterClass() {}

}
```

NoLargeCupsTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class NoLargeCupsTesting.
 */
public class NoLargeCupsTesting {


  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Executed before each method be directly after reaching state Coin Inserted.
   */
  @BeforeMethod
  public void setUp() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P8
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P14
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);// P23
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P30
  // T7
  assertEquals(this.vendingMachine.coin(), 1);
 }

 /**
  * Test 28.
  */
 @Test(priority = 28)
 public void test28() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P93
  // T24
  assertEquals(this.vendingMachine.tea(), 1);// P161
  // T8
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P35
  // T6
  assertEquals(this.vendingMachine.coin(), 1);// P28
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }

 /**
  * Test 29.
  */
 @Test(priority = 29)
 public void test29() {
  // T7 is the last transition due to the setUp()
  // P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P93
  // T24
  assertEquals(this.vendingMachine.tea(), 1);// P162
  // T29
  assertEquals(this.vendingMachine.coin(), 1);// P166
  // T29
  assertEquals(this.vendingMachine.coin(), 1);// P165
  // T8
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P32
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P16
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }

 /**
  * Test 30.
  */
 @Test(priority = 30)
 public void test30() {
  // T7 is the last transition due to the setUp()
  // P
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P92
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P153
  // T26
  assertEquals(this.vendingMachine.tea(), 1);// P163
  // T8
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P31
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P10
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
 }


 /**
  * Test 31.
  */
 @Test(priority = 31)
 public void test31() {
  // T7 is the last transition due to the setUp()
  // P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P150
  // T18
  assertEquals(this.vendingMachine.large_cup(), 1);// P144
  // T26
  assertEquals(this.vendingMachine.tea(), 1);// P164
  // T29
  assertEquals(this.vendingMachine.coin(), 1);// P166
  // T29
  assertEquals(this.vendingMachine.coin(), 1);// P165
```

```
    // T8
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P33
    // T4
    assertEquals(this.vendingMachine.set_price(25), 1);// P22
    // T5
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Tear down.
   *
   * @param result the result
   */
  @AfterMethod
  public void tearDown(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
      System.out.println("The " + result.getName() + " has failed!");
    }
    if (result.getStatus() == ITestResult.SUCCESS) {
      System.out.println("The " + result.getName() + " passed!");
    }
    if (result.getStatus() == ITestResult.SKIP) {
      System.out.println("The " + result.getName() + " has been skipped!");
    }
  }

  /**
   * After class.
   */
  @AfterClass
  public void afterClass() {}

}
```

AdditionalTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class AdditionalTesting.
 */
public class AdditionalTesting {

  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Test 36.
   */
  @Test(priority = 36)
  public void test36() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P8
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P14
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);// P23
    // T6
    assertEquals(this.vendingMachine.coin(), 1);// P30
    // T7
    assertEquals(this.vendingMachine.coin(), 1);// P83
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);// P104
    // T11
    assertEquals(this.vendingMachine.tea(), 1);// P49
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P11
    // T6
    assertEquals(this.vendingMachine.coin(), 1);// P30
    // T7
    assertEquals(this.vendingMachine.coin(), 1);// P83
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);// P108
    // T22
    assertEquals(this.vendingMachine.sugar(), 1);// P154
    // T27
    assertEquals(this.vendingMachine.tea(), 1);// P157
    // T9
    assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P40
    // T5
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Test 37.
   */
  @Test(priority = 37)
  public void test37() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P8
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P14
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);// P23
    // T6
    assertEquals(this.vendingMachine.coin(), 1);// P30
    // T7
    assertEquals(this.vendingMachine.coin(), 1);// P83
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);// P104
    // T22
    assertEquals(this.vendingMachine.sugar(), 1);// P149
    // T16
    assertEquals(this.vendingMachine.coin(), 1);// P127
    // T27
    assertEquals(this.vendingMachine.tea(), 1);// P157
    // T9
    assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P41
    // T6
    assertEquals(this.vendingMachine.coin(), 1);// P30
    // T7
    assertEquals(this.vendingMachine.coin(), 1);// P81
    // T19
    assertEquals(this.vendingMachine.large_cup(), 1);// P92
    // T22
    assertEquals(this.vendingMachine.sugar(), 1);// P149
    // T16
    assertEquals(this.vendingMachine.coin(), 1);// P126
    // T26
    assertEquals(this.vendingMachine.tea(), 1);// P163
    // T8
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P34
    // T5
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Test 38.
   */
  @Test(priority = 38)
  public void test38() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P8
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P14
    // T4
    assertEquals(this.vendingMachine.set_price(25), 1);// P23
    // T7
    assertEquals(this.vendingMachine.coin(), 1);// P83
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);// P110
    // T25
    assertEquals(this.vendingMachine.tea(), 1);// P155
    // T9
    assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P42
    // T7
    assertEquals(this.vendingMachine.coin(), 1);// P81
    // T19
    assertEquals(this.vendingMachine.large_cup(), 1);// P93
    // T24
    assertEquals(this.vendingMachine.tea(), 1);// P161
    // T8
    assertEquals(this.vendingMachine.insert_large_cups(3), 1);// P36
    // T7
    assertEquals(this.vendingMachine.coin(), 1);// P83
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);// P106
    // T20
    assertEquals(this.vendingMachine.coin(), 1);// P96
    // T11
    assertEquals(this.vendingMachine.tea(), 1);// P51
    // T4
    assertEquals(this.vendingMachine.set_price(25), 1);// P20
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P18
    // T7
    assertEquals(this.vendingMachine.coin(), 1);// P83
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);// P103
    // T10
    assertEquals(this.vendingMachine.cancel(), 1);// P43
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P10
    // T5
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Test 39.
   */
  @Test(priority = 39)
  public void test39() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());// P1
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(10), 1);// P8
```

```
// T3
assertEquals(this.vendingMachine.insert_small_cups(10), 1);// P14
// T4
assertEquals(this.vendingMachine.set_price(25), 1);// P23
// T7
assertEquals(this.vendingMachine.coin(), 1);// P83
// T21
assertEquals(this.vendingMachine.small_cup(), 1);// P104
// T11
assertEquals(this.vendingMachine.tea(), 1);// P50
// T3
assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P15
// T4
assertEquals(this.vendingMachine.set_price(50), 1);// P23
// T6
assertEquals(this.vendingMachine.coin(), 1);// P30
// T7
assertEquals(this.vendingMachine.coin(), 1);// P83
// T21
assertEquals(this.vendingMachine.small_cup(), 1);// P104
// T11
assertEquals(this.vendingMachine.tea(), 1);// P53
// T6
assertEquals(this.vendingMachine.coin(), 1);// P30
// T7
assertEquals(this.vendingMachine.coin(), 1);// P81
// T19
assertEquals(this.vendingMachine.large_cup(), 1);// P168
// T12
assertEquals(this.vendingMachine.tea(), 1);// P59
// T6
assertEquals(this.vendingMachine.coin(), 1);// P30
// T7
assertEquals(this.vendingMachine.coin(), 1);// P81
// T19
assertEquals(this.vendingMachine.large_cup(), 1);// P90
// T20
assertEquals(this.vendingMachine.coin(), 1);// P169
// T12
assertEquals(this.vendingMachine.tea(), 1);// P57
// T4
assertEquals(this.vendingMachine.set_price(25), 1);// P24
// T7
assertEquals(this.vendingMachine.coin(), 1);// P81
```

```
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P168
  // T12
  assertEquals(this.vendingMachine.tea(), 1);// P55
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P12
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P168
  // T12
  assertEquals(this.vendingMachine.tea(), 1);// P60
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P168
  // T12
  assertEquals(this.vendingMachine.tea(), 1);// P56
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P16
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 40.
 */
@Test(priority = 40)
public void test40() {
  // T1
  assertNotNull(this.vendingMachine = new VendingMachine());// P1
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P8
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P14
  // T4
  assertEquals(this.vendingMachine.set_price(25), 1);// P23
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P83
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P90
  // T20
  assertEquals(this.vendingMachine.coin(), 1);// P101
  // T24
  assertEquals(this.vendingMachine.tea(), 1);// P161
  // T8
  assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P36
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P83
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P106
  // T20
  assertEquals(this.vendingMachine.coin(), 1);// P102
  // T25
  assertEquals(this.vendingMachine.tea(), 1);// P155
  // T9
  assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P42
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P147
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P72
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P151
  // T18
  assertEquals(this.vendingMachine.large_cup(), 1);// P137
  // T13
  assertEquals(this.vendingMachine.tea(), 1);// P66
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P150
  // T17
  assertEquals(this.vendingMachine.small_cup(), 1);// P130
  // T15
  assertEquals(this.vendingMachine.tea(), 1);// P78
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P79
  // T10
  assertEquals(this.vendingMachine.cancel(), 1);// P46
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
/**
 * Test 41.
 */
@Test(priority = 41)
public void test41() {
  // T1
  assertNotNull(this.vendingMachine = new VendingMachine());// P1
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P8
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P14
  // T4
  assertEquals(this.vendingMachine.set_price(25), 1);// P23
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P83
  // T21
  assertEquals(this.vendingMachine.small_cup(), 1);// P108
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P152
  // T23
  assertEquals(this.vendingMachine.sugar(), 1);// P118
  // T25
  assertEquals(this.vendingMachine.tea(), 1);// P155
  // T9
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P42
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P81
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P92
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P152
  // T23
  assertEquals(this.vendingMachine.sugar(), 1);// P117
  // T24
  assertEquals(this.vendingMachine.tea(), 1);// P161
  // T8
  assertEquals(this.vendingMachine.insert_large_cups(1), 1);// P34
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
/**
 * Test 42.
 */
@Test(priority = 42)
public void test42() {
  // T1
  assertNotNull(this.vendingMachine = new VendingMachine());// P1
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P8
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(1), 1);// P14
  // T4
  assertEquals(this.vendingMachine.set_price(25), 1);// P23
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P151
  // T18
  assertEquals(this.vendingMachine.large_cup(), 1);// P138
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P72
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P151
  // T18
  assertEquals(this.vendingMachine.large_cup(), 1);// P143
  // T23
  assertEquals(this.vendingMachine.sugar(), 1);// P116
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P149
  // T16
  assertEquals(this.vendingMachine.coin(), 1);// P119
  // T13
  assertEquals(this.vendingMachine.tea(), 1);// P64
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
/**
 * Test 43.
 */
@Test(priority = 43)
public void test43() {
 // T1
 assertNotNull(this.vendingMachine = new VendingMachine());// P1
 // T2
 assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P8
 // T3
 assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P14
 // T4
 assertEquals(this.vendingMachine.set_price(25), 1);// P23
 // T7
 assertEquals(this.vendingMachine.coin(), 1);// P84
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P151
 // T17
 assertEquals(this.vendingMachine.small_cup(), 1);// P129
 // T14
 assertEquals(this.vendingMachine.cancel(), 1);// P72
 // T7
 assertEquals(this.vendingMachine.coin(), 1);// P84
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P150
 // T17
 assertEquals(this.vendingMachine.small_cup(), 1);// P133
 // T18
 assertEquals(this.vendingMachine.large_cup(), 1);// P141
 // T17
 assertEquals(this.vendingMachine.small_cup(), 1);// P134
 // T23
 assertEquals(this.vendingMachine.sugar(), 1);// P116
 // T22
 assertEquals(this.vendingMachine.sugar(), 1);// P149
 // T16
 assertEquals(this.vendingMachine.coin(), 1);// P119
 // T13
 assertEquals(this.vendingMachine.tea(), 1);// P64
 // T5
 assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
/**
 * Test 52.
 */
@Test(priority = 52)
public void test52() {
  // T1
  assertNotNull(this.vendingMachine = new VendingMachine());// P2
  // T3
  assertEquals(this.vendingMachine.insert_small_cups(2), 1);// P15
  // T4
  assertEquals(this.vendingMachine.set_price(25), 1);// P19
  // T2
  assertEquals(this.vendingMachine.insert_large_cups(2), 1);// P6
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P82
  // T20
  assertEquals(this.vendingMachine.coin(), 1);// P100
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P152
  // T23
  assertEquals(this.vendingMachine.sugar(), 1);// P114
  // T20
  assertEquals(this.vendingMachine.coin(), 1);// P97
  // T19
  assertEquals(this.vendingMachine.large_cup(), 1);// P87
  // T10
  assertEquals(this.vendingMachine.cancel(), 1);// P48
  // T7
  assertEquals(this.vendingMachine.coin(), 1);// P84
  // T22
  assertEquals(this.vendingMachine.sugar(), 1);// P147
  // T14
  assertEquals(this.vendingMachine.cancel(), 1);// P70
  // T5
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
  /**
   * Tear down.
   *
   * @param result the result
   */
  @AfterMethod
  public void tearDown(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
      System.out.println("The " + result.getName() + " has failed!");
    }
    if (result.getStatus() == ITestResult.SUCCESS) {
      System.out.println("The " + result.getName() + " passed!");
    }
    if (result.getStatus() == ITestResult.SKIP) {
      System.out.println("The " + result.getName() + " has been skipped!");
    }
  }

  /**
   * After class.
   */
  @AfterClass
  public void afterClass() {}

}
```

IdleGhostTransitionlTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class IdleGhostTransitionTesting.
 */
public class IdleGhostTransitionTesting {

  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Test 44.
   */
  @Test(priority = 44)
  public void test44() {
    assertNotNull(this.vendingMachine = new VendingMachine());
    assertEquals(this.vendingMachine.small_cup(), 0);
    assertEquals(this.vendingMachine.large_cup(), 0);
    assertEquals(this.vendingMachine.sugar(), 0);
    assertEquals(this.vendingMachine.tea(), 0);
    assertEquals(this.vendingMachine.insert_large_cups(0), 0);
    assertEquals(this.vendingMachine.insert_small_cups(0), 0);
    assertEquals(this.vendingMachine.set_price(0), 0);
    assertEquals(this.vendingMachine.cancel(), 0);
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Tear down.
   *
   * @param result the result
   */
  @AfterMethod
  public void tearDown(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
      System.out.println("The " + result.getName() + " has failed!");
    }
    if (result.getStatus() == ITestResult.SUCCESS) {
      System.out.println("The " + result.getName() + " passed!");
    }
    if (result.getStatus() == ITestResult.SKIP) {
      System.out.println("The " + result.getName() + " has been skipped!");
    }
  }

  /**
   * After class.
   */
  @AfterClass
  public void afterClass() {}

}
```

CoinsInsertedGhostTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class CoinsInsertedGhostTransitionTesting.
 */
public class CoinsInsertedGhostTransitionTesting {

  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Executed before each method be directly after reaching state Coin Inserted.
   */
  @BeforeMethod
  public void setUp() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(2), 1);
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(2), 1);
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);
    // T6
    assertEquals(this.vendingMachine.coin(), 1);
    // T7
    assertEquals(this.vendingMachine.coin(), 1);
  }

  /**
   * Test 45.
   */
  @Test(priority = 45)
  public void test45() {
    assertEquals(this.vendingMachine.tea(), 0);
    assertEquals(this.vendingMachine.insert_large_cups(-1), 0);
    assertEquals(this.vendingMachine.insert_small_cups(-1), 0);
    assertEquals(this.vendingMachine.set_price(-1), 0);
    assertEquals(this.vendingMachine.insert_large_cups(1), 0);
    assertEquals(this.vendingMachine.insert_small_cups(1), 0);
    assertEquals(this.vendingMachine.set_price(1), 0);
    assertEquals(this.vendingMachine.dispose(), 0);
    assertEquals(this.vendingMachine.cancel(), 1);
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Tear down.
   *
   * @param result the result
   */
  @AfterMethod
  public void tearDown(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
      System.out.println("The " + result.getName() + " has failed!");
    }
    if (result.getStatus() == ITestResult.SUCCESS) {
      System.out.println("The " + result.getName() + " passed!");
    }
    if (result.getStatus() == ITestResult.SKIP) {
      System.out.println("The " + result.getName() + " has been skipped!");
    }
  }
```

```
  /**
   * After class.
   */
  @AfterClass
  public void afterClass() {}

}
```

SugarGhostTransitionTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class SugarGhostTransitionTesting.
 */
public class SugarGhostTransitionTesting {

  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Executed before each method be directly after reaching state Sugar.
   */
  @BeforeMethod
  public void setUp() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(2), 1);
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(2), 1);
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);
    // T6
    assertEquals(this.vendingMachine.coin(), 1);
    // T7
    assertEquals(this.vendingMachine.coin(), 1);
    // T22
    assertEquals(this.vendingMachine.sugar(), 1);
  }

  /**
   * Test 46.
   */
  @Test(priority = 46)
  public void test46() {
    assertEquals(this.vendingMachine.tea(), 0);
    assertEquals(this.vendingMachine.insert_large_cups(-1), 0);
    assertEquals(this.vendingMachine.insert_small_cups(-1), 0);
    assertEquals(this.vendingMachine.set_price(-1), 0);
    assertEquals(this.vendingMachine.insert_large_cups(1), 0);
    assertEquals(this.vendingMachine.insert_small_cups(1), 0);
    assertEquals(this.vendingMachine.set_price(1), 0);
    assertEquals(this.vendingMachine.dispose(), 0);
    assertEquals(this.vendingMachine.sugar(), 1);
    assertEquals(this.vendingMachine.cancel(), 1);
    assertEquals(this.vendingMachine.dispose(), 1);
  }

  /**
   * Tear down.
   *
   * @param result the result
   */
  @AfterMethod
  public void tearDown(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
      System.out.println("The " + result.getName() + " has failed!");
    }
    if (result.getStatus() == ITestResult.SUCCESS) {
      System.out.println("The " + result.getName() + " passed!");
    }
    if (result.getStatus() == ITestResult.SKIP) {
      System.out.println("The " + result.getName() + " has been skipped!");
    }
  }
```

```
 /**
  * After class.
  */
 @AfterClass
 public void afterClass() {}

}
```

NoSmallCupGhostTransitionTesting.java

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class NoSmallCupsGhostTransitionTesting.
 */
public class NoSmallCupsGhostTransitionTesting {

  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Executed before each method be directly after reaching state No Small Cups.
   */
  @BeforeMethod
  public void setUp() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);
    // T6
    assertEquals(this.vendingMachine.coin(), 1);
    // T7
    assertEquals(this.vendingMachine.coin(), 1);
    // T21
    assertEquals(this.vendingMachine.small_cup(), 1);
    // T25
    assertEquals(this.vendingMachine.tea(), 1);
  }

  /**
   * Test 47.
   */
  @Test(priority = 47)
  public void test47() {
    assertEquals(this.vendingMachine.small_cup(), 0);
    assertEquals(this.vendingMachine.large_cup(), 0);
    assertEquals(this.vendingMachine.sugar(), 0);
    assertEquals(this.vendingMachine.tea(), 0);
    assertEquals(this.vendingMachine.insert_large_cups(-1), 0);
    assertEquals(this.vendingMachine.insert_small_cups(-1), 0);
    assertEquals(this.vendingMachine.set_price(-1), 0);
    assertEquals(this.vendingMachine.insert_large_cups(1), 0);
    assertEquals(this.vendingMachine.set_price(1), 0);
    assertEquals(this.vendingMachine.cancel(), 0);
    assertEquals(this.vendingMachine.dispose(), 0);
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);
    assertEquals(this.vendingMachine.dispose(), 1);
  }
```

```
 /**
  * Tear down.
  *
  * @param result the result
  */
 @AfterMethod
 public void tearDown(ITestResult result) {
   if (result.getStatus() == ITestResult.FAILURE) {
     System.out.println("The " + result.getName() + " has failed!");
   }
   if (result.getStatus() == ITestResult.SUCCESS) {
     System.out.println("The " + result.getName() + " passed!");
   }
   if (result.getStatus() == ITestResult.SKIP) {
     System.out.println("The " + result.getName() + " has been skipped!");
   }
 }

 /**
  * After class.
  */
 @AfterClass
 public void afterClass() {}

}
```

NoLargeCupsGhostTransitionTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class NoLargeCupsGhostTransitionTesting.
 */
public class NoLargeCupsGhostTransitionTesting {

  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Executed before each method be directly after reaching state No Large Cups
   */
  @BeforeMethod
  public void setUp() {
    // T1
    assertNotNull(this.vendingMachine = new VendingMachine());
    // T2
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);
    // T3
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);
    // T4
    assertEquals(this.vendingMachine.set_price(50), 1);
    // T6
    assertEquals(this.vendingMachine.coin(), 1);
    // T7
    assertEquals(this.vendingMachine.coin(), 1);
    // T21
    assertEquals(this.vendingMachine.large_cup(), 1);
    // T24
    assertEquals(this.vendingMachine.tea(), 1);
  }

  /**
   * Test 48.
   */
  @Test(priority = 48)
  public void test48() {
    assertEquals(this.vendingMachine.small_cup(), 0);
    assertEquals(this.vendingMachine.large_cup(), 0);
    assertEquals(this.vendingMachine.sugar(), 0);
    assertEquals(this.vendingMachine.tea(), 0);
    assertEquals(this.vendingMachine.insert_large_cups(-1), 0);
    assertEquals(this.vendingMachine.insert_small_cups(-1), 0);
    assertEquals(this.vendingMachine.set_price(-1), 0);
    assertEquals(this.vendingMachine.insert_small_cups(1), 0);
    assertEquals(this.vendingMachine.set_price(1), 0);
    assertEquals(this.vendingMachine.cancel(), 0);
    assertEquals(this.vendingMachine.dispose(), 0);
    assertEquals(this.vendingMachine.insert_large_cups(1), 1);
    assertEquals(this.vendingMachine.dispose(), 1);
  }
```

```
 /**
  * Tear down.
  *
  * @param result the result
  */
 @AfterMethod
 public void tearDown(ITestResult result) {
  if (result.getStatus() == ITestResult.FAILURE) {
    System.out.println("The " + result.getName() + " has failed!");
  }
  if (result.getStatus() == ITestResult.SUCCESS) {
    System.out.println("The " + result.getName() + " passed!");
  }
  if (result.getStatus() == ITestResult.SKIP) {
    System.out.println("The " + result.getName() + " has been skipped!");
  }
 }

 /**
  * After class.
  */
 @AfterClass
 public void afterClass() {}

}
```

AdditionalBranchTesting.java:

```
package iit.test.valentinpichavant.tests;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertNotNull;

import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import iit.test.valentinpichavant.VendingMachine;

/**
 * The Class AdditionalBranchTesting.
 */
public class AdditionalBranchTesting {

  /** The vending machine. */
  VendingMachine vendingMachine;

  /**
   * Before class.
   */
  @BeforeClass
  public void beforeClass() {}

  /**
   * Test 49.
   */
  @Test(priority = 49)
  public void test49() {
    assertNotNull(this.vendingMachine = new VendingMachine());
    assertEquals(this.vendingMachine.insert_small_cups(1), 1);
    assertEquals(this.vendingMachine.set_price(25), 1);
    assertEquals(this.vendingMachine.coin(), 1);
    assertEquals(this.vendingMachine.large_cup(), 1);
    assertEquals(this.vendingMachine.tea(), 0);
    assertEquals(this.vendingMachine.cancel(), 1);
    assertEquals(this.vendingMachine.dispose(), 1);
  }
```

```
/**
 * Test 50.
 */
@Test(priority = 50)
public void test50() {
  assertNotNull(this.vendingMachine = new VendingMachine());
  assertEquals(this.vendingMachine.insert_large_cups(2), 1);
  assertEquals(this.vendingMachine.set_price(25), 1);
  assertEquals(this.vendingMachine.coin(), 1);
  assertEquals(this.vendingMachine.small_cup(), 1);
  assertEquals(this.vendingMachine.sugar(), 1);
  assertEquals(this.vendingMachine.tea(), 0);
  assertEquals(this.vendingMachine.cancel(), 1);
  assertEquals(this.vendingMachine.dispose(), 1);
}

/**
 * Test 51.
 */
@Test(priority = 51)
public void test51() {
  assertNotNull(this.vendingMachine = new VendingMachine());
  assertEquals(this.vendingMachine.set_price(25), 1);
  assertEquals(this.vendingMachine.coin(), 1);
  assertEquals(this.vendingMachine.sugar(), 1);
  assertEquals(this.vendingMachine.small_cup(), 1);
  assertEquals(this.vendingMachine.tea(), 0);
  assertEquals(this.vendingMachine.large_cup(), 1);
  assertEquals(this.vendingMachine.tea(), 0);
  assertEquals(this.vendingMachine.insert_large_cups(2), 0);
  assertEquals(this.vendingMachine.coin(), 1);
  assertEquals(this.vendingMachine.sugar(), 1);
  assertEquals(this.vendingMachine.small_cup(), 1);
  assertEquals(this.vendingMachine.tea(), 0);
  assertEquals(this.vendingMachine.cancel(), 1);
  assertEquals(this.vendingMachine.dispose(), 1);
}
```

```
  /**
   * Tear down.
   *
   * @param result the result
   */
  @AfterMethod
  public void tearDown(ITestResult result) {
    if (result.getStatus() == ITestResult.FAILURE) {
      System.out.println("The " + result.getName() + " has failed!");
    }
    if (result.getStatus() == ITestResult.SUCCESS) {
      System.out.println("The " + result.getName() + " passed!");
    }
    if (result.getStatus() == ITestResult.SKIP) {
      System.out.println("The " + result.getName() + " has been skipped!");
    }
  }

  /**
   * After class.
   */
  @AfterClass
  public void afterClass() {}

}
```

The class ExecuteTS.java:

```
package iit.test.valentinpichavant;

import java.io.File;
import java.io.FileNotFoundException;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * The Class ExecuteTS which executed the TS.txt if found in the same folder.
 */
public class ExecuteTS {


  /**
   * Instantiates a new execute TS.
   */
  public ExecuteTS(){
    Scanner s;
    try {
      // We read the TS.txt file
      s = new Scanner(new
File(Paths.get(".").toAbsolutePath().normalize().toString()+File.separator+"TS.txt"));
      ArrayList<String> list = new ArrayList<String>();
      //We parse it a list of string
      while (s.hasNextLine()){
        list.add(s.nextLine());
      }
      //We execute the test suite.
      this.executeTestSuite(list);
      s.close();
    } catch (FileNotFoundException e) {
      e.printStackTrace();
    }
  }

  /**
   * Execute test suite.
   *
   * @param file the file as a list of string
   */
```

```
private void executeTestSuite(List<String> file){
 //For each line of the file
 for(String line : file){
  //If the line is a test
  if(line.startsWith("Test#")){
   Scanner s;
   //We create a boolean to check if the test can be incorrect
   boolean fail = false;
   s = new Scanner(line);
   //We save the name of the test
   String test = s.next();
   //We initialize a new VendingMachine
   VendingMachine vendingMachine = new VendingMachine();
   //We start the test
   System.out.println("Begin of the execution of "+test);
   //While there is a next method and the test hasn't failed
   while (s.hasNext() && fail == false){
    //At the beginning nothing is return
    int returnCode = 0;
    //We test what we found
    switch(s.next()){
     //For each case, we do the corresponding call and display the result.
     case "coin":
      returnCode = vendingMachine.coin();
      System.out.println("coin was executed and return "+returnCode+".");
      break;
     case "small_cup":
      returnCode = vendingMachine.small_cup();
      System.out.println("small_cup was executed and return "+returnCode+".");
      break;
     case "large_cup":
      returnCode = vendingMachine.large_cup();
      System.out.println("large_cup was executed and return "+returnCode+".");
      break;
     case "sugar":
      returnCode = vendingMachine.sugar();
      System.out.println("sugar was executed and return "+returnCode+".");
      break;
     case "tea":
      returnCode = vendingMachine.tea();
      System.out.println("tea was executed and return "+returnCode+".");
      break;
     case "insert_large_cups":
      System.out.print("insert_large_cups with ");
```

```
        try{
          int value = Integer.parseInt(s.next());
          returnCode = vendingMachine.insert_large_cups(value);
          System.out.println("parameter "+value+" executed and return "+returnCode+".");
        }catch(NumberFormatException nfe){
          System.out.println("bad parameter \""+s+"\" was not executed.");
        }
        break;
      case "insert_small_cups":
        System.out.print("insert_small_cups with ");
        try{
          int value = Integer.parseInt(s.next());
          returnCode = vendingMachine.insert_small_cups(value);
          System.out.println("parameter "+value+" executed and return "+returnCode+".");
        }catch(NumberFormatException nfe){
          System.out.println("bad parameter \""+s+"\" was not executed.");
        }
        break;
      case "set_price":
        System.out.print("set_price with ");
        try{
          int value = Integer.parseInt(s.next());
          returnCode = vendingMachine.set_price(value);
          System.out.println("parameter "+value+" executed and return "+returnCode+".");
        }catch(NumberFormatException nfe){
          System.out.println("bad parameter \""+s+"\" was not executed.");
        }
        break;
      case "cancel":
        returnCode = vendingMachine.cancel();
        System.out.println("cancel was executed and return "+returnCode+".");
        break;
      case "dispose":
        returnCode = vendingMachine.dispose();
        System.out.println("dispose was executed and return "+returnCode+".");
        break;
      default:
        System.out.println("The "+test+" has failed.");
        fail = true;
        break;
    }
  }
  s.close();
```

```
        System.out.println("End of the successful execution of "+test.substring(0, test.length()-
1)+".\n");
      }
    }
  }
}
```