

Part 1 (Vending Machine) Unit Test

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 import edu.cpp.cs5800.VendingMachine.snacks.Snack;
4 import edu.cpp.cs5800.VendingMachine.states.Idle;
5 import org.junit.jupiter.api.Test;
6
7 import static org.junit.jupiter.api.Assertions.assertEquals;
8
9 public class IdleTest {
10     VendingMachine vendingMachine = new VendingMachine();
11     Idle idle = new Idle(vendingMachine);
12
13     @Test
14     public void testSelectSnack() {
15         int snackId = 1;
16         Snack snack = vendingMachine.getSnack(snackId);
17         String expected = "You have selected: " + snackId + " (" + snack.getName() + ")";
18         String actual = idle.selectSnack(snackId);
19         assertEquals(expected, actual);
20     }
21
22     @Test
23     public void testInsertAmount() {
24         String expected = "Invalid request: Please first select a snack!";
25         assertEquals(expected, idle.insertMoney(6));
26     }
27
28     @Test
29     public void testDispenseSnack() {
30         String expected = "Invalid request: Please first select a snack!";
31         assertEquals(expected, idle.dispenseSnack());
32     }
33 }
34
```

Part 1 (Vending Machine) Unit Test

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 import edu.cpp.cs5800.VendingMachine.snacks.Snack;
4 import org.junit.jupiter.api.Test;
5
6 import static org.junit.jupiter.api.Assertions.assertEquals;
7
8 class SnackDummy extends Snack {
9
10     public SnackDummy(int quantity, double price) {
11         super("SnackDummy", quantity, price);
12     }
13 }
14
15 public class SnackTest {
16     Snack snack = new SnackDummy(3, 1.5);
17
18     @Test
19     public void TestGetName() {
20         assertEquals("SnackDummy", snack.getName());
21     }
22
23     @Test
24     public void TestGetQuantity() {
25         assertEquals(3, snack.getQuantity());
26     }
27
28     @Test
29     public void TestGetPrice() {
30         assertEquals(1.5, snack.getPrice());
31     }
32
33     @Test
34     public void TestDispense() {
35         assertEquals("SnackDummy", snack.dispense());
36         assertEquals(2, snack.getQuantity());
37     }
38 }
39
```

Part 1 (Vending Machine) Unit Test

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 import edu.cpp.cs5800.VendingMachine.states.Idle;
4 import edu.cpp.cs5800.VendingMachine.states.WaitingForMoney;
5 import org.junit.jupiter.api.Test;
6
7 import static org.junit.jupiter.api.Assertions.assertEquals;
8
9 public class VendingMachineTest {
10     VendingMachine vendingMachine = new VendingMachine();
11
12     @Test
13     public void testGetSnack() {
14         assertEquals("Coke", vendingMachine.getSnack(1).getName());
15     }
16
17     @Test
18     public void testGetAndSetState() {
19         assertEquals((new Idle(vendingMachine)).toString(), vendingMachine.getState().
20 toString());
21         vendingMachine.setState(new WaitingForMoney(vendingMachine));
22         assertEquals((new WaitingForMoney(vendingMachine)).toString(),
23 vendingMachine.getState().toString());
24     }
25
26     @Test
27     public void testSetAndGetSelectedSnack() {
28         vendingMachine.setSelectedSnack(2);
29         assertEquals(2, vendingMachine.getSelectedSnack());
30     }
31 }
```

Part 1 (Vending Machine) Unit Test

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 import edu.cpp.cs5800.VendingMachine.states.DispensingSnack;
4 import org.junit.jupiter.api.Test;
5
6 import static org.junit.jupiter.api.Assertions.assertEquals;
7
8 public class DispensingSnackTest {
9     VendingMachine vendingMachine = new VendingMachine();
10    DispensingSnack dispensingSnack = new DispensingSnack(vendingMachine);
11
12    @Test
13    public void testSelectSnack() {
14        String expected = "Invalid request: Currently dispensing snack!";
15        assertEquals(expected, dispensingSnack.selectSnack(1));
16    }
17
18    @Test
19    public void testInsertAmount() {
20        String expected = "Invalid request: Currently dispensing snack!";
21        assertEquals(expected, dispensingSnack.insertMoney(6));
22    }
23
24    @Test
25    public void testDispenseSnack() {
26        String expected = "Completed transaction!";
27        assertEquals(expected, dispensingSnack.dispenseSnack());
28    }
29 }
30
```

Part 1 (Vending Machine) Unit Test

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 import edu.cpp.cs5800.VendingMachine.states.WaitingForMoney;
4 import org.junit.jupiter.api.Test;
5
6 import static org.junit.jupiter.api.Assertions.assertEquals;
7
8 public class WaitingForMoneyTest {
9     VendingMachine vendingMachine = new VendingMachine();
10    WaitingForMoney wfm = new WaitingForMoney(vendingMachine);
11
12    @Test
13    public void testSelectSnack() {
14        String expected = "Invalid request: Please inserted money!";
15        assertEquals(expected, wfm.selectSnack(1));
16    }
17
18    @Test
19    public void testInsertAmount() {
20        vendingMachine.setSelectedSnack(1);
21        vendingMachine.setState(wfm);
22        String expected = "You have inserted: $" + 6.0;
23        assertEquals(expected, wfm.insertMoney(6));
24    }
25
26    @Test
27    public void testDispenseSnack() {
28        String expected = "Invalid request: Please inserted money!";
29        assertEquals(expected, wfm.dispenseSnack());
30    }
31 }
32
```

Part 1 (Vending Machine) Unit Test

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 import org.junit.jupiter.api.Test;
4
5 import static org.junit.jupiter.api.Assertions.assertFalse;
6 import static org.junit.jupiter.api.Assertions.assertTrue;
7
8 public class SnackDispenseHandlerTest {
9     VendingMachine vendingMachine = new VendingMachine();
10    SnackDispenseHandler handler = vendingMachine.getSnackDispenseHandler();
11
12    @Test
13    public void testDispenseWithEnoughMoneyAndQty() {
14        assertTrue(handler.dispense(6, 10));
15    }
16
17    @Test
18    public void testDispenseWithNotEnoughQty() {
19        assertTrue(handler.dispense(6, 10));
20        assertFalse(handler.dispense(6, 10));
21    }
22
23    @Test
24    public void testDispenseWithNotEnoughMoney() {
25        assertFalse(handler.dispense(1, 1));
26    }
27
28    @Test
29    public void testDispenseSnackNotFound() {
30        assertFalse(handler.dispense(7, 1));
31    }
32 }
33
```