DTU Informatik
Informatik og Matematiks Modellering

Spring 2008
Hubert Baumeister

02161 Software Engineering 1

Exercise Design Patterns

Vending Machine

Implement a vending machine for vending bananas and apples. The machine takes 1, 2, and 5 DKK. A banana costs 7 DKK and an apple 3 DKK. It should be possible first to select the thing one wants to have and then input the money, or first input the money and then select either banana or apple. After each sale, the money paid to much should be returned, e.g. if one has put in 10 DKK for a banana, the vending machine should return 3 DKK.

The vending machine has a stock of bananas and apples. If there are no bananas anymore, the machine should say so when selecting a banana and return the money that was already input. The same should happen when the machine runs out of apples. There is no need to create an operation for refilling the machine. Instead, when creating the vending machine, it should be possible to tell the machine how much bananas and apples it has.

In addition it should be possible on demand to see how much apples and bananas the machine contains whenever the amount of bananas or apples change. Similarly, on demand one should be able to also see how much money the vending machine contains, i.e. the money that was input after the machine was set up and for which bananas and apples were dispensed.

More precisely, when the vending machine is started one should see the following screen:

```
   1) Input 1 DKK
   2) Input 2 DKK
   3) Input 5 DKK
   4) Select banana
   5) Select apple
   6) Toggle show number of apples and bananas
   7) Toggle show money
0 DKK
Select a number (1-7):  _
```

0 DKK is the amount of money currently input to buy something. If one e.g. inputs 1, 2, or 3, the same screen appears where the amount put into the machine is updated accordingly, e.g. after pressing 3 the screen will look like

```
   1) Input 1 DKK
...
5 DKK
Select a number (1-7):  _
```

If one selects a banana (e.g. types 4), the result should be:

```
Dispensing banana and 0 DKK rest money
   1) Input 1 DKK
...
0 DKK
Select a number (1-7): _
```

If one switches on reporting of numbers of bananas and apples by typing 6, the screen should report the number of bananas and apples:

```
10 bananas and 5 apples
...
   1) Input 1 DKK
...
Select a number (1-7): _
```

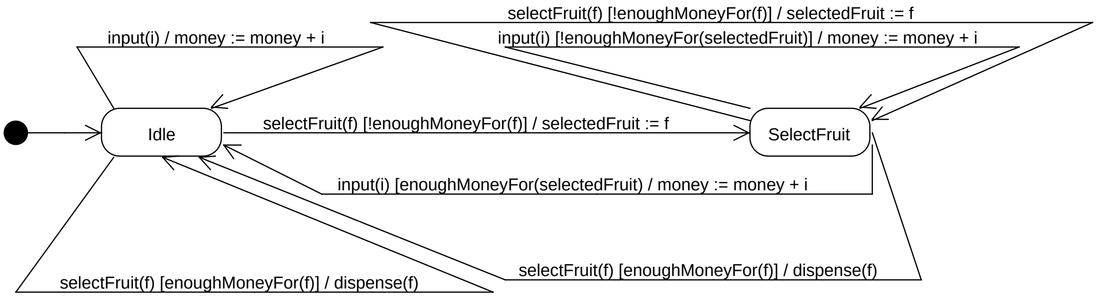
One can switch off showing the number of bananas and apples by again typing 6. Similarly, one should be able to toggle seeing the amount of money by typing 7. In case reporting is switched on and there are 100 DKK in the machine, one should see:

```
Machine contains 100 DKK
...
   1) Input 1 DKK
...
Select a number (1-7): _
```

Note that it should be possible to have showing the stock and number of money on at the same time as well as only one or the other.

Tasks

The following diagram shows the behaviour of the vending machine.


selectFruit(f) [!enoughMoneyFor(f)] / selectedFruit := f
input(i) / money := money + i
input(i) [!enoughMoneyFor(selectedFruit)] / money := money + i
Idle
selectFruit(f) [!enoughMoneyFor(f)] / selectedFruit := f
SelectFruit
input(i) [enoughMoneyFor(selectedFruit) / money := money + i
selectFruit(f) [enoughMoneyFor(f)] / dispense(f)
selectFruit(f) [enoughMoneyFor(f)] / dispense(f)

Note that the state machine reduces the actions 1–3 and 4–5 to two separate methods, `input(int money)` and `selectFruit(String fruit)`.

Implement the vending machine using the state pattern for the vending machine and the observer pattern for printing the contents of the vending machine and how much money the machine has collected so far.

Design Patterns

1. Find examples of the composite pattern in the `java.awt.*` classes.
2. Check for the programs you wrote so far in the course where design patterns could be applied, or where you already have applied them.