Barista-matic Programming Assignment

PROBLEM DESCRIPTION:

Your task is to create a simulator of an automatic coffee dispensing machine, called the *Barista-matic*. The machine maintains an inventory of drink ingredients, and is able to dispense a fixed set of possible drinks by combining these ingredients in different amounts. The cost of a drink is determined by its component ingredients.

Upon startup, the *Barista-matic* should display a list of its current inventory, followed by a menu to allow the user to select a drink. As drinks are dispensed, the inventory should be updated. Only drinks for which there is sufficient inventory can be dispensed.

The specified input and output formats for the *Barista-matic* must be followed exactly. At the end of these instructions, you will find examples of some input/output scenarios.

Your Barista-matic machine should be capable of dispensing the following drinks:

Drink Name	Ingredients
Coffee	3 units of coffee, 1 unit of sugar, 1 unit of cream
Decaf Coffee	3 units of Decaf Coffee, 1 unit of sugar, 1 unit of cream
Caffe Latte	2 units of espresso, 1 unit of steamed milk
Caffe Americano	3 units of espresso
Caffe Mocha	1 units of Espresso, 1 unit of cocoa, 1 unit of steamed milk, 1 unit of whipped cream
Cappuccino	2 units of Espresso, 1 unit of steamed milk, 1 unit of foamed milk

Per-unit ingredient costs are as follows:

Ingredient	Unit Cost
Coffee	$0.75
Decaf Coffee	$0.75
Sugar	$0.25
Cream	$0.25
Steamed Milk	$0.35
Foamed Milk	$0.35
Espresso	$1.10
Cocoa	$0.90
Whipped Cream	$1.00

Initially the Barista-matic should contain 10 units of all ingredients, and restocking the machine should restore each ingredient to a maximum of 10 units.

INPUT FORMAT:

Your solution should read from the standard input stream, one command per line. No prompts or other extraneous user messages should be displayed. Blank input lines should be ignored.

Each valid command consists of a single character, as follows:

- 'R' or 'r' - restock the inventory and redisplay the menu
- 'Q' or 'q' - quit the application
- [1-6] - order the drink with the corresponding number in the menu

If the user enters an invalid command, then the program should display a single-line message with the following format:

```
Invalid selection: <characters that were entered>
```

If the user selects a valid drink number, and the machine has the required ingredients to make the drink, then the program should display a single-line message with the following format:

```
Dispensing: <drink name>
```

On the other hand, if the drink order cannot be completed, then the program should display a single-line message with the following format:

```
Out of stock: <drink name>
```

The inventory and menu (see next section) should be displayed immediately following any applicable message.

OUTPUT FORMAT:

All output should be written to the standard output stream. At program startup, and following the processing of every command, the machine inventory and the drink menu should be displayed. Both the inventory list and the drink menu should be displayed in alphabetic order (by ingredient name or drink name, respectively), in the following format:

```
Inventory:
<ingredient name>,<quantity in inventory>
...
<ingredient name>,<quantity in inventory>
Menu:
<drink number>,<drink name>,<cost>,<in-stock>
...
<drink number>,<drink name>,<cost>,<in-stock>
```

Drinks should be numbered sequentially, starting at 1, in the order they are displayed in the menu. The in-stock indicator should be either "true" or "false".

Note: the sample output is indented in these instructions to make it easier to read. The output generated by your program should *not* have any whitespace at the beginning of a line.

TECHNICAL NOTES:

Your solution should be a command-line program written in Java (or other language, as appropriate for position). If you use any external libraries in developing your solution (i.e. libraries that are not part of the standard Java platform) then you should bundle these libraries with your code so that we can run your solution.

It is *not* required that the initial machine configuration (inventory counts, available drinks and prices, etc.) be dynamic. In particular, it is acceptable to perform this initialization in code, rather than reading the configuration from an external file or database. However, your program should be flexible enough to allow new drinks to be added to the menu without requiring extensive code changes.

Make sure your program works correctly for all combinations of inputs. You may include automated tests if you like (using a framework such as JUnit or NUnit), but this is not required.

Extensive inline or method-level comments are not required, unless you want to include them to highlight particular aspects of your design or implementation.

EXAMPLE:

Upon application startup, the initial inventory list and drink menu would look like this:

```
Inventory:
Cocoa,10
Coffee,10
Cream,10
Decaf Coffee,10
Espresso,10
Foamed Milk,10
Steamed Milk,10
Sugar,10
Whipped Cream,10
Menu:
1,Caffe Americano,$3.30,true
2,Caffe Latte,$2.55,true
3,Caffe Mocha,$3.35,true
4,Cappuccino,$2.90,true
5,Coffee,$2.75,true
6,Decaf Coffee,$2.75,true
```

For input consisting of the following commands:

```
2
q
```

the program would produce the following output (including the startup output):

```
Inventory:
Cocoa,10
Coffee,10
Cream,10
Decaf Coffee,10
Espresso,10
Foamed Milk,10
Steamed Milk,10
Sugar,10
Whipped Cream,10
Menu:
1,Caffe Americano,$3.30,true
2,Caffe Latte,$2.55,true
3,Caffe Mocha,$3.35,true
4,Cappuccino,$2.90,true
5,Coffee,$2.75,true
6,Decaf Coffee,$2.75,true
Dispensing: Caffe Latte
Inventory:
Cocoa,10
Coffee,10
```

Cream,10
Decaf Coffee,10
Espresso,8
Foamed Milk,10
Steamed Milk,9
Sugar,10
Whipped Cream,10
Menu:
1,Caffe Americano,$3.30,true
2,Caffe Latte,$2.55,true
3,Caffe Mocha,$3.35,true
4,Cappuccino,$2.90,true
5,Coffee,$2.75,true
6,Decaf Coffee,$2.75,true