**CSS 142 Assignment 2**

**University of Washington Bothell  
Autumn 2017**

**Due: Wednesday, 11 Oct 2017, 1:00pm (15 minutes before class)**

**Goal**: Understanding input/output, branching, and starting loops

You have been asked to write software for the new **CoffeeOrder 3000** machine. The interaction with the machine is as follows:

**Test case 1: User wants no drinks and enters 0**

Welcome to CoffeeOrder 3000

How many espresso drinks would you like today (0-3): 0

OK, so you do not want any espressos today!

Thank you for using CoffeeOrder 3000

**Test case 2: User wants 1 drink: Mocha (size Grande)**

Welcome to CoffeeOrder 3000

How many espresso drinks would you like today (0-3): 1

Getting order for espresso drink #1

What type (1: Americano, 2: Latte, 3: Mocha): 3

What size (1: Tall, 2: Grande): 2

Your total is 4.45

Thank you for using CoffeeOrder 3000

**Test case 3: User wants 2 drinks: Americano (size Grande) and Mocha (size Tall)**

Welcome to CoffeeOrder 3000

How many espresso drinks would you like today (0-3): 2

Getting order for espresso drink #1

What type (1: Americano, 2: Latte, 3: Mocha): 1

What size (1: Tall, 2: Grande): 2

Getting order for espresso drink #2

What type (1: Americano, 2: Latte, 3: Mocha): 3

What size (1: Tall, 2: Grande): 1

Your total is 6.80

Thank you for using CoffeeOrder 3000

**Test case 4: User wants 5 drinks (invalid input)**

Welcome to CoffeeOrder 3000

How many espresso drinks would you like today (0-3): 5

**java.lang.AssertionError**

**Outputs in Assignment 2:**

1. Welcome to CoffeeOrder 3000
2. How many espresso drinks would you like today (0-3):
   1. Note: 1 space after ‘:’
3. Print out the next 3 lines n times
4. Getting order for espresso drink #n
   1. Note: n is the number of drink entered by user, and starts from 1 to n
5. What type (1: Americano, 2: Latte, 3: Mocha):
   1. Note: 1 space after ‘:’
6. What size (1: Tall, 2: Grande):
   1. Note: 1 space after ‘:’
7. Your total is <total>
   1. Note: <total> is the total cost of all drinks
8. Thank you for using CoffeeOrder 3000
   1. Note: use println or \n to print out new line at the end of each output

**Coffee prices are as follows:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Tall** | **Grande** |
| Americano | 2.65 | 2.95 |
| Latte | 3.35 | 3.95 |
| Mocha | 3.85 | 4.45 |

It might be useful to tackle this problem in stages

Level 0: Handle the case of 0 coffee

Level 1: Handle the case of 4 or more coffees by giving an error (we will learn about handling bad input more gracefully later)

Level 2: Write the function getPrice and test it. One possible way to write the function would be as follows:

public static double getPrice(int coffeeType, int size)

Level 3: Write a loop that will take 1-3 orders, but at this level don’t accept any input from the user, but make sure the output has the correct lines:

Getting order for espresso drink #1

What type (1: Americano, 2: Latte, 3: Mocha):

What size (1: Tall, 2: Grande):

Level 4: Take the actual coffee orders, add up the prices to get the total and print it.

Level 5: Test and simplify your program. Are there parts that can be done in a separate function rather than main.

**NOTE**: You need to use assert() to capture invalid inputs. For example, I want my int number to be positive, I use: assert(number > 0);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Ratings** | | | **Points** |
| Indentation, Variable and Function Names | Meets Expectations  2 | Partially Meets Expectations  1 | Does Not Meet Expectations  0 | 2 |
| Level 0 and 1 - No coffee condition, too many coffees | Meets Expectations  1 | Does Not Meet Expectations  0 | | 1 |
| Level 2 + Appropriate functions as needed | Meets Expectations  2 | Partially Meets Expectations  1 | Does Not Meet Expectations  0 | 2 |
| Level 3 and 4 - Loop implemented, correct output, correct total | Meets Expectations  4 | Partially Meets Expectations  2 | Does Not Meet Expectations  0 | 4 |
| Level 5 - Overall readability and structure | Meets Expectations  1 | Does Not Meet Expectations  0 | | 1 |

BlueJ creates a README.txt file for every new project. You can edit this file inside BlueJ and it is located in the same directory as your java file.

**CSS 142 Assignment 2 - ANSWERS**

/\*\*

\* Write a description of class CoffeeOrder here.

\*

\* sizes Tall, Grande

\* Americano 2.65 2.95

\* Latte 3.35 3.95

\* Mocha 3.85 4.45

\* @author (your name)

\* @version (a version number or a date)

\*/

import java.util.Scanner;

public class CoffeeOrder

{

public static double getPrice(int coffeeType, int size) {

assert ((coffeeType > 0) && (coffeeType < 4));

assert ((size > 0) && (size < 3));

if (coffeeType == 1) {

if (size == 1) {

return 2.65; // Americano, Tall

}else {

return 2.95; // Americano, Grande

}

} else if (coffeeType == 2){

if (size == 1){

return 3.35; // Latte, Tall

}else{

return 3.95; // Latte, Grande

}

}else if (coffeeType == 3) {

if (size == 1) {

return 3.85; // Mocha , Tall

} else {

return 4.45; // Mocha , Grande

}

}

return 0;

}

public static void getOrder()

{

Scanner keyboard = new Scanner(System.in);

System.out.println("Welcome to CoffeeOrder 3000");

System.out.println("How many espresso drinks would you like today (0-3): ");

int coffees = keyboard.nextInt();

assert ((coffees > -1) && (coffees < 4));

double total = 0.0;

for (int i = 1; i <= coffees; i++)

{

System.out.format("Getting order for espresso drink #%d\n", i);

System.out.println("What type (1: Americano, 2: Latte, 3: Mocha): ");

int coffeeType = keyboard.nextInt();

System.out.println("What size (1: Tall, 2: Grande): ");

int size = keyboard.nextInt();

total = total + getPrice(coffeeType, size);

} if (total > 0) {

System.out.format("Your total is %.2f\n", total);

} else {

System.out.println("OK, so you do not want any espressos today!");

}

System.out.println("Thank you for using CoffeeOrder 3000");

}

public static void main(String[] args)

{

getOrder();

}

}