/\*\*

\* @ (#)BrownCadenaHustedOballeEscarenoWooten02TPA2.java

\* @author(s) Amari Brown, Victoria Cadena, Jack Husteed, Aldo Oballe-Escareno, Alexandre Wooten

\* @version 1.00 2022/07/30 11:01 AM

\*

\* PROGRAM PURPOSE: Create a program that accepts

\* a customers order for hammocks. The customer

\* gets a discount based on the order total (size).

\* A sales receipt will be generated.

\*/

import java.util.\*; //IMPORTING JAVA UTIL PACKAGE

import java.util.Scanner; //IMPORTING INPUT CAPTURE CLASS

import java.util.Calendar; // IMPORTING DATE AND TIME CAPUTURE CLASS

public class BrownCadenaHustedOballeEscarenoWooten02TPA2

{

// Global Variables

static Scanner input = new Scanner(System.in); // to take user input

static Calendar dateTime = Calendar.getInstance(); // to get calendar date and time

static String hammockDesc = "", colorSelected = ""; // hammock desctiption and color selected strings

static double price = 0.0; // price of individual units

static boolean repeat = false; // to test repeat

// The string format for our hammock company

static String salesReceipt = String.format("%n%nSALES RECIPT" +

"%n%nLAZY HAZY DAYS, INC." +

"%nHuebner Oaks Mall" +

"%nSan Antonio, TX" +

"%n%nDate: %tD" +

"%nTime: %tr%n", dateTime, dateTime);

/\*\*

\* This main method went through several iterations and was contributed by our full group.

\* We worked by sending updated .java files that contained fixed variables, logic, and grammar errors.

\* The purpose of this main method is to call the promptHammock() method for the hammock choice, call the

\* promptColor() method for the color choice, call both the setHammockSizePrice() and setColor() methods to

\* store the size and the colors, and ask for the quantity. This method will repeat as long as the morePurchases

\* variable is set to 'y'. It also contains the calculations for discount, discSubtotal, tax, and total. These

\* variables will then be sent to the salesReciept() method to calculate and display your reciept.

\*/

public static void main(String[] args)

{

//Local variables

int hammock = 0, // hammock choice

quantity = 0, // hammock quantity

color = 0, // hammock color

iterations = 1; // total iterations

char morePurchases = 'y'; // test for morePurchases for the many do-while loops

double itemTotal = 0.0, // item totals

subtotal = 0.0, // sub total for items

discount = 0.0, // discount based on subtotal

discSubtotal = 0.0, // subtotal accounting for discount

tax = 0.0, // tax at .0825

total = 0.0, // total accounting for all variables

discRate = 0.0; // discount rate based on quantity purchased

boolean printFinal = false; // once true, the program will print the final reciept

do // do-while that tests for morePurchases.

{

do // do-while that tests for morePurchases and if the hammock variable is outside of 1 - 3

{

hammock = promptHammock(); // the hammock stores the value returned from the promptHammock() method

if(hammock < 1 || hammock > 3) // only executes if the user inputs a value outside of the 1 - 3 range.

{

System.out.printf("You entered an invalid hammock choice. Enter 'Y' to re-enter your " +

"choice or 'N' to exit the program: ");

morePurchases = input.next().toLowerCase().charAt(0); // converts the users input to lowercase. Will exit

// the program if the user inputs 'n'.

}// END OF IF STATEMENT

}while(morePurchases == 'y' && hammock < 1 || hammock > 3); // END OF DO-WHILE

if(morePurchases == 'y') // tests if more purchases is 'y'

{

setHammockSizePrice(hammock); // Sends the hammock variable to the setHammockSizePrice method

quantity = promptQuantity(); // quantity variable stores the value returned from the promptQuantity() method

do // do-while that tests for morePurchases and if color is outside the 1 - 5 range

{

color = promptColor();// color variable stores the value returned from the promptColor() method

if(color < 1 || color > 5) // only executes if the user-input is outside the 1 - 5 range

{

System.out.printf("You entered an invalid color choice. Enter 'Y' to re-enter your " +

"choice of 'N' to exit the program: ");

morePurchases = input.next().toLowerCase().charAt(0); // converts the users input to lowercase. Will exit

// the program if the user inputs 'n'.

}//END close brace line comment

}while(morePurchases == 'y' && color < 1 || color > 5); //END OF DO-WHILE

if(morePurchases == 'y') // tests if more purchases is 'y'

{

setHammockColor(color); // sends the color variable to the setHammockColor() method

itemTotal = quantity \* price; // itemTotal calculation

subtotal += itemTotal; // subtotal calculation

}//END OF IF STATEMENT

}//END OF IF STATEMENT

if(morePurchases == 'y') // tests if more purchases is 'y'

{

// adds the string format to the salesReceipt variable

salesReceipt += String.format("%n%-24s %c %-13s %5s %,7d %4s %s%,14.2f", hammockDesc, '-', colorSelected,

" ", quantity, " ", iterations > 1 ? " " : "$", itemTotal);

System.out.printf("%nDo you want to add another hammock? Enter 'Y' or 'N': ");

morePurchases = input.next().toLowerCase().charAt(0); // asks the user if they want to add another hammock

if(morePurchases == 'n') // tests if more purchases is 'n'

{

printFinal = true; // Once true this will display the salesReciept

}//END Insert if line comment

}//END Insert if line comment

iterations += 1; // Increment iterations

}while(morePurchases == 'y'); //END OF DO-WHILE LOOP

if(subtotal >= 100000.00) // if statement that determines the discount based on the subtotal

{

discRate = .035;

}

else

{

if(subtotal >= 50000.00)

{

discRate = .03;

}

else

{

if(subtotal >= 10000.00)

{

discRate = .025;

}

else

{

if(subtotal >= 5000)

{

discRate = .02;

}

else

{

discRate = 0.0;

}//END if subtotal >= 5000 2% discount else discount is 0

}//END if subtotal >= 10000 2.5% discount else subtotal >= 5000

}//END if subtotal >= 50000 3% discount else subtotal >= 10000

}//END if subtotal >= 100000 3.5% discount else subtotal >= 50000

discount = discRate \* subtotal; // discount calculation

discSubtotal += subtotal - discount; // discSubtotal calculation

tax = discSubtotal \* .0825; // tax calculation

total = discSubtotal + tax; // total calculation

if(printFinal == true) // if printFinal == true then the displaySalesReceipt() method executes the final reciept

{

displaySalesReceipt(subtotal, discount, tax, total);

}//END Insert if line comment

System.exit(0); // exit main

}//END main()

/\*\*

\* This method went through several iterations and was contributed by our full group.

\* We worked by sending updated .java files that contained fixed variables, logic, and grammer errors.

\* The promptHammock() method displays a prompt for the user to input a number to be

\* returned and stored within the hammock variable in main.

\*/

public static int promptHammock(){

int hammock; // local hammock variable to store user input

do{

System.out.printf("LAZY HAZY DAYS, INC.%n%n");

System.out.printf("Our beautiful cotton hammocks sport a traditional look and are very comfortable.%n%n" +

"1. Small - 48 in. x 11ft. - Good for 1 person\t\t\t\t$100.00%n" +

"2. Large - 55 in. x 13ft. - Good for 2 people\t\t\t\t$140.00%n" +

"3. Deluxe - 60 in. x 13ft. - Good for 2 or more people\t\t\t$175.00%n%n");

System.out.printf("Enter your choice: "); //PROMPT 1

repeat = !input.hasNextInt(); // returns true if the input buffer contains anything not a number

hammock = input.nextInt(); // captures the user input as the hammock variable

validateNumber(); // call the validateNumber() method

} while(repeat == true); // END OF DO-WHILE

return hammock; // return the hammock variable to main

} // End of promptHammock

/\*\*

\* This method went through several iterations and was contributed by our full group.

\* We worked by sending updated .java files that contained fixed variables, logic, and grammer errors.

\* The setHammockSizePrice method accepts the sent variable as int hammock. This will then be used

\* in a switch statement to determine the hammockDesc and price variables.

\*/

public static void setHammockSizePrice(int hammock){

switch(hammock){

case 1: // if 1 is chosen then Small and 100 are given to the hammockDesc and price variables

hammockDesc = "Small - 48 in. x 11ft.";

price = 100.00;

break;

case 2: // if 2 is chosen then Large and 140 are given to the hammockDesc and price variables

hammockDesc = "Large - 55 in. x 13ft.";

price = 140.00;

break;

case 3: // if 3 is chosen then Deluxe and 175 are given to the hammockDesc and price variables

hammockDesc = "Deluxe - 60 in. x 13ft.";

price = 175.00;

break;

}

} // End of setHammocksSizePrice

/\*\*

\* This method went through several iterations and was contributed by our full group.

\* We worked by sending updated .java files that contained fixed variables, logic, and grammer errors.

\* The promptQuantity() method prompts the user to input the quantity desired. This number is returned to

\* quantity variable in main.

\*/

public static int promptQuantity(){

do{

System.out.printf("%nEnter the quantity: "); //PROMPT 2

repeat = !input.hasNextInt(); // returns true if the input buffer contains anything not a number

validateNumber(); // call the validateNumber() method

} while(repeat == true); // END OF DO-WHILE

return input.nextInt(); // Return the quantity value of the prompt to the quantity variable in main

} // End of promptQuantity

/\*\*

\* This method went through several iterations and was contributed by our full group.

\* We worked by sending updated .java files that contained fixed variables, logic, and grammer errors.

\* This method captures user input based on the prompt of color. It returns this value back to the color

\* variable in main.

\*/

public static int promptColor(){

do{

System.out.printf("%n1. Crimson Red%n" +

"2. Emerald Green%n" +

"3. Indigo Blue%n" +

"4. Natural%n" +

"5. Purple Haze%n%n");

System.out.printf("Enter your choice of colors: "); //PROMPT 3

repeat = !input.hasNextInt(); // returns true if the input buffer contains anything not a number

validateNumber();// call the validateNumber() method

} while(repeat == true);// END OF DO-WHILE LOOP

return input.nextInt(); // returns the integer to the color variable in main

} // End of PromptColor

/\*\*

\* This method went through several iterations and was contributed by our full group.

\* We worked by sending updated .java files that contained fixed variables, logic, and grammer errors.

\* This method is a switch statement that determines which color case is true.

\*/

public static void setHammockColor(int color){

switch(color){

case 1: //If 1 is entered Crimson Red will take the value of colorSelected

colorSelected = "Crimson Red";

break;

case 2: //If 2 is entered Emerald Green will take the value of colorSelected

colorSelected = "Emerald Green";

break;

case 3: //If 3 is entered Indigo Blue will take the value of colorSelected

colorSelected = "Indigo Blue";

break;

case 4: //If 4 is entered Natural will take the value of colorSelected

colorSelected = "Natural";

break;

case 5: //If 5 is entered Purple Haze will take the value of colorSelected

colorSelected = "Purple Haze";

break;

default: //If the number entered is not within the range 'Invalid Color" will be displayed

colorSelected = "Invalid color";

System.out.printf("Invalid color choice!%n%n");

break;

}

} // End of setHammockColor

/\*\*

\* This method went through several iterations and was contributed by our full group.

\* We worked by sending updated .java files that contained fixed variables, logic, and grammer errors.

\* This method validates the numbers from user input it does not return a value

\* If the repeat boolean is true an error message will be printed the above is for the validatenumber method.

\*/

public static void validateNumber(){

if(repeat == true){

input.next();

System.out.printf("%nYou must enter a valid integer or floating-point value!%n");

} // END OF IF STATEMENT

} // End of validateNumber

/\*\*

\* This method went through several iterations and was contributed by our full group.

\* We worked by sending updated .java files that contained fixed variables, logic, and grammer errors.

\* This method formats the sales reciept to be printed from the values entered and calculations

\* This adds the sales reciept to be outputted to the salesReceipt variable and prints it.

\*/

public static void displaySalesReceipt(double subtotal, double discount, double tax, double total){

salesReceipt += String.format(

"%n%n%52s %-6s $%,14.2f" +

"%n%52s %-7s %,14.2f" +

"%n%52s %-7s %,14.2f" +

"%n%n%52s %-6s $%,14.2f%n",

"SUBTOTAL:", " ",

subtotal, "DISCOUNT:", " ",

discount, "TAX @ 8.250%:",

" ", tax, "TOTAL:", " ", total);

System.out.printf("%s",salesReceipt);

} // End of displaySalesReceipt

}//END APPLICATION CLASS BrownCadenaHustedOballeEscarenoWooten02TPA2