/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Author: Frankie Deleon & John Olsen

\* Course Section: IS-2063-ON2

\* Date: 04/23/2023

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\*

\* PROGRAM PURPOSE:main class for the program. It creates a SalesSystem object named sales, calls the start() method of the

\* sales object to begin the sales system, and then exits the program with a status code of 0.

\*/

public class DeleonOlsenON1PA3 // main class DeleonOlsenON1PA3. By: Frankie Deleon

{

/\*Main method for DeleonOlsenON1PA3 class. By: Frankie Deleon

\*/

public static void main(String[] args)

{

SalesSystem sales = new SalesSystem(); // Create SalesSystem object named sales. By: Frankie Deleon

sales.start(); // Call start() from SalesSystem object sales. By: Frankie Deleon

System.exit(0); //Exit statement. By: Frankie Deleon

}

}

/\* Output 1:

\*\*SALES ON TRACK\*\*

TANDEM ENTERPRISES

Begi1n the processing of a sales report? Enter 'Y' or 'N': n

Exiting Sales System.

TANDEM ENTERPRISES

Begi1n the processing of a sales report? Enter 'Y' or 'N': y

What is the projected annual sales for Tandem? 5OO000

Warning: You entered an invalid integer or floating-point value.

What is the projected annual sales for Tandem? 500000

How many sales reps work for Tandem? @

Warning: You entered an invalid integer or floating-point value.

How many sales reps work for Tandem? 2

Enter the name of a sales rep: ju!ian caesar

Invalid name! Please re-enter: julian caesar

Sales Rep Name: julian caesar

Is this name correct? 'Y' or 'N': y

Enter the number of quarters worked (no less than 1 or greater than 4): l

Warning: You entered an invalid integer or floating-point value.

OUT OF RANGE! Re-enter the number of quarters worked (no less than 1 or greater than 4): 0

OUT OF RANGE! Re-enter the number of quarters worked (no less than 1 or greater than 4): 5

OUT OF RANGE! Re-enter the number of quarters worked (no less than 1 or greater than 4): 1

1. First Quarter

2. Second Quarter

3. Third Quarter

4. Fourth Quarter

Choose the quarter in which sales were earned: !

Warning: You entered an invalid integer or floating-point value.

OUT OF RANGE!

1. First Quarter

2. Second Quarter

3. Third Quarter

4. Fourth Quarter

Choose the quarter in which sales were earned: 0

OUT OF RANGE!

1. First Quarter

2. Second Quarter

3. Third Quarter

4. Fourth Quarter

Choose the quarter in which sales were earned: 5

OUT OF RANGE!

1. First Quarter

2. Second Quarter

3. Third Quarter

4. Fourth Quarter

Choose the quarter in which sales were earned: 1

Enter the sales revenue for the 1st month of the First Quarter: !00000

Warning: You entered an invalid integer or floating-point value.

Enter the sales revenue for the 1st month of the First Quarter: 100000

Enter the sales revenue for the 2nd month of the First Quarter: 50000

Enter the sales revenue for the 3rd month of the First Quarter: 100000

Enter the name of the next sales rep: monique La femme

Sales Rep Name: monique La femme

Is this name correct? 'Y' or 'N': y

Enter the number of quarters worked (no less than 1 or greater than 4): 2

1. First Quarter

2. Second Quarter

3. Third Quarter

4. Fourth Quarter

Choose the quarter in which sales were earned: 1

Enter the sales revenue for the 1st month of the First Quarter: 100000

Enter the sales revenue for the 2nd month of the First Quarter: 75000

Enter the sales revenue for the 3rd month of the First Quarter: 25000

1. First Quarter

2. Second Quarter

3. Third Quarter

4. Fourth Quarter

Choose the next quarter in which sales were earned: 2

Enter the sales revenue for the 1st month of the Second Quarter: 10000

Enter the sales revenue for the 2nd month of the Second Quarter: 25000

Enter the sales revenue for the 3rd month of the Second Quarter: 30000

%n%nTANDEM ENTERPRISES

SALES REVENUE FOR 1 QUARTER(S) OF 2023

SALES REP: Julian Caesar

Total Year-To-Date: $ 250,000.00

Keep up the GOOD work, Julian. There is a possible year-end bonus!

%n%n TANDEM ENTERPRISES

SALES REVENUE FOR 2 QUARTER(S) OF 2023

SALES REP: Monique La femme

Total Year-To-Date: $ 265,000.00

Keep up the GOOD work, Monique. There is a possible year-end bonus!

CORPORATE SALES PERFORMANCE

It's been a GOOD year so far. There could be a year-end bonus of

about 2-5% if we can keep on top of our sales goals. Thank you all

and please continue your excellent effort!

Enter the file name for sales report (WARNING: This will erase a pre-existing file!): SalesReports.txt

Data written to the SalesReports.txt file.

Enter the name for the sales report file: salesreports.txt

04/23/23, 1, Julian Caesar, 250000.00

04/23/23, 2, Monique La femme, 265000.00

Exiting Sales System.

\* Output 2:

\*\*\*SALES NOT ON TRACK\*\*\*

TANDEM ENTERPRISES

Begi1n the processing of a sales report? Enter 'Y' or 'N': Y

What is the projected annual sales for Tandem? 500000

How many sales reps work for Tandem? 1

Enter the name of a sales rep: stella brown

Sales Rep Name: stella brown

Is this name correct? 'Y' or 'N': y

Enter the number of quarters worked (no less than 1 or greater than 4): 1

1. First Quarter

2. Second Quarter

3. Third Quarter

4. Fourth Quarter

Choose the quarter in which sales were earned: 3

Enter the sales revenue for the 1st month of the Third Quarter: 150000

Enter the sales revenue for the 2nd month of the Third Quarter: 55000

Enter the sales revenue for the 3rd month of the Third Quarter: 80000

%n%nTANDEM ENTERPRISES

SALES REVENUE FOR 1 QUARTER(S) OF 2023

SALES REP: Stella Brown

Total Year-To-Date: $ 285,000.00

Keep up the GOOD work, Stella. There is a possible year-end bonus!

CORPORATE SALES PERFORMANCE

Sales are lagging projections. A year-end bonus may not be possible.

Enter the file name for sales report (WARNING: This will erase a pre-existing file!): SalesReport.txt

Data written to the SalesReport.txt file.

Enter the name for the sales report file: SalesReport.txt

04/23/23, 1, Stella Brown, 285000.00

Exiting Sales System.

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Author: Frankie Deleon & John Olsen

\* Course Section: IS-2063-ON2

\* Date: 04/23/2023

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* WARNING: THE CODE IN THIS PROGRAM "CANNOT" BE ALTERED.

\* NO POSTING OF THIS CODE IS ALLOWED ANY WHERE AS IT

\* IS THE INTELLECTUAL PROPERTY OF THE AUTHOR.

\*

\* STUDENTS ARE TO INSERT CODE AND COMMENTS WHERE INDICATED IN CAPS

\* STARTING WITH "STUDENT INSERTS ... " OR CODE ... ". USE DRJAVA'S

\* Find TO LOCATE THE INSERT AND CODE INSTRUCTIONS. \*

\*

\* @(#)SalesReport.java

\* @author Linda Shepherd

\* @version 1.00 2022/10/28 11:55 PM

\*

\* PROGRAM PURPOSE: STUDENT INSERTS.

\* By: Frankie Deleon

\* The SalesReport program creates the get and set methods used for the SalesSystem program.The copy method in

\* the SalesReport class that creates a new SalesReport object reportObj which accepts noOfQtrs,

\* salesRep, quarterlySalesin that order. The equals method which accepts the SalesReport report object

\* and returns a boolean isEqual.The method test whether noOfQtrs, salesRep, AND quarterlySales ARE EQUAL

\* TO THE SAME VARIABLES IN THE INCOMING report OBJECT. If true isEqual equals true. The setProjectedSales()

\* method asks the user to enter the projected annual sales for Tandem and validates the input to ensure

\* that it is a double. If the input is not a double, the method calls the validateNumber() method to

\* validate the input as a number. Once the input is validated as a double, the method assigns the value

\* to the projectedSales variable. Finally, the method clears the input buffer using the nextLine()

\* method to ensure that any remaining input is removed before proceeding. The setNoSalesReps() method

\* prompts the user to input the number of sales representatives that work for Tandem.

\* It validates the user input and returns the value entered as an integer.

\* If the user enters a non-integer value, the method will continue to prompt untila valid value is entered.

\* The value entered is also assigned to the field 'noSalesReps'. returns noSalesReps entered by the user

\* as an integer. The setSalesRep method sets the value of the salesRep field to the value passed as

\* a parameter. Returns nothing.setSalesRep() receives the sales rep count, prompts for the name of the rep and ensures

\* the proper capitalization of the name. \* setNoQtrs() prompts for the number of quarters a sales rep

\* has worked. The entry is validated for proper conversion to an integer. determineMonthNo() receives

\* the monthCounter to figure out which month it is in the quarter: 1st, 2nd, 3rd. setSalesRevenue() asks for the sales revenue earned

\* for a given month within a given quarter. The entry is validated for proper conversion to a double

\* before it is returned to the calling statement. setQuarterlySales method sets the value of the

\* quarterlySales field to the value passed as a parameter. Returns nothing.

\* calculateQtrlySales() adds the current sales revenue to the existing quarterly sales revenue of the

\* sales rep and returns nothing. calculateAnnualSales() adds the quarterly sales revenue.

\* getSalesRepRevReport() prints the total year-to-date sales revenue earned by a sales rep for the number of

\* quarters worked in the current year. to the existing annual sales revenue. getRepTargetMsg() method

\* calculates the sales earned by the sales rep to determine if the sales rep is on target. It then creates a message

\* about year-end bonuses based on the sales rep's sales performance. getCompanyTargetMsg method

\* calculates the percentage of the projected annual sales that the company has achieved so far. It then prints a

\* message to the console indicating whether or not the company is on track to meet its sales goals and if a year-end bonus is possible.

\* validateNumber method for handling user input when an expected integer or double value is not entered.

\* getDate() method accepts no parameters and returns a formatted string of the date as 99/99/9999.

\*

\*/

import java.util.Scanner; //By: John Importing the Scanner class from the java.util package for taking input from the user. //STUDENT INSERTS LINE COMMENTS FOR EACH IMPORT STMT.

import java.util.Calendar; //By: John Importing the Calendar class from the java.util package for working with dates and times.

public class SalesReport

{

//STUDENT INSERTS LINE COMMENTS DESCRIPTIVE OF THE PURPOSE OF EACH VARIABLE.

private Scanner input = new Scanner(System.in); // SCANNER OBJECT TO TAKE INPUT FROM USER.

private Calendar dateTime = Calendar.getInstance(); // CALENDAR OBJECT TO WORK WITH DATES AND TIMES.

private String monthNo = ""; //By: John STRING VARIABLE TO STORE THE NUMBER OF THE MONTH.

private String salesRep = ""; //By: John STRING VARIABLE TO STORE THE NAME OF THE SALES REPRESENTATIVE.

private String quarter = ""; //By: John STRING VARIABLE TO STORE THE QUARTER OF THE YEAR.

private double quarterlySales = 0.0; //By: John DOUBLE VARIABLE TO STORE THE QUARTERLY SALES OF THE REPRESENTATIVE.

private double salesRevenue = 0.0; //By: John DOUBLE VARIABLE TO STORE THE SALES REVENUE OF THE REPRESENTATIVE.

/\* These static fields are data for

\* the company and not just the salesRep which

\* is why they're being treated as static, so all

\* SalesReport objects can use them.\*/

private static double annualSales = 0.0; //By: John STATIC VARIABLE TO STORE THE TOTAL ANNUAL SALES.

private static double projectedSales = 0.0; //By: John STATIC VARIABLE TO STORE THE PROJECTED SALES FOR THE YEAR.

private static double percOfTargetCo = 0.0; //By: John STATIC VARIABLE TO STORE THE PERCENTAGE OF TARGET SALES ACHIEVED BY THE COMPANY.

private double percOfTargetRep = 0.0; // INSTANCE VARIABLE TO STORE THE PERCENTAGE OF TARGET SALES ACHIEVED BY THE REPRESENTATIVE.

private int noOfQtrs = 0; //By: John INSTANCE VARIABLE TO STORE THE NUMBER OF QUARTERS IN THE YEAR.

private int qtrChoice = 0; //By: John INSTANCE VARIABLE TO STORE THE USER'S CHOICE OF QUARTER.

private int noOfMonths = 3; //By: John INSTANCE VARIABLE TO STORE THE NUMBER OF MONTHS IN EACH QUARTER.

private static int noSalesReps = 0; //By: John STATIC VARIABLE TO STORE THE NUMBER OF SALES REPRESENTATIVES IN THE COMPANY.

private boolean repeat;

/\* NOTE: EXCEPT CONSTRUCTORS THAT DON'T HAVE A return TYPE.

\* ALL OTHER METHODS IN THIS CLASS ARE INSTANCE METHODS WHICH

\* REQUIRES AN OBJECT TO CALL THEM IN THE CLIENT CLASS.\*/

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By: John

\* THIS IS A CONSTRUCTOR METHOD FOR THE SALESREPORT CLASS.

\* IT DOES NOT HAVE A RETURN TYPE AND IS USED TO INITIALIZE THE OBJECT WHEN IT IS CREATED.

\*/

public SalesReport()

{

}//END Default Constructor

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By: Frankie Deleon

\* a constructor for the SalesReport class that takes three parameters: an integer representing the number

\* of quarters, a String representing the name of the sales representative, and a double representing the

\* quarterly sales amount. The constructor then calls three set methods to set the values of these parameters

\* for the SalesReport object being created.

\*

\*/

//CODE THE CONSTRUCTOR'S HEADER BASED ON THE CLOSE BRACE LINE COMMENT.

public SalesReport(int noOfQtrs, String salesRep, double quarterlySales) // By: Frankie Deleon

{

setNoQtrs(noOfQtrs);

setSalesRep(salesRep);

setQuarterlySales(quarterlySales);

}//END Constructor(noOfQtrs: int, salesRep: String, quarterlySales: double)

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By: Frankie Deleon

\* copy method in the SalesReport class that creates a new SalesReport object

\* reportObj which accepts noOfQtrs, salesRep, quarterlySales

\* in that order.

\*

\*/

public SalesReport copy()

{

/\* CODE BELOW THE reportObj ACCEPTING noOfQtrs, salesRep, quarterlySales

\* IN THAT ORDER.\*/

SalesReport reportObj = new SalesReport(getNoQtrs(), getSalesRep(), getQuarterlySales());

return reportObj;

}//END copy(): SalesReport

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By: Frankie Deleon

\* equals method which accepts the SalesReport report object and returns a boolean isEqual.

\* The method test whether noOfQtrs, salesRep, AND quarterlySales

\* ARE EQUAL TO THE SAME VARIABLES IN THE INCOMING report

\* OBJECT. If true isEqual equals true.

\*

\*/

//CODE THE equals HEADER BASED ON THE CLOSE BRACE LINE COMMENT.

public boolean equals(SalesReport report) //By: Frankie Deleon

{

/\* Variable isEqual determines content equality of 2 SaleReport objects.

\* Starting premise is NO (false).\*/

boolean isEqual = false;

/\* CODE BELOW THE if HEADER THAT TESTS, IN THE ORDER LISTED,

\* WHETHER noOfQtrs, salesRep, AND quarterlySales

\* ARE EQUAL TO THE SAME VARIABLES IN THE INCOMING report

\* OBJECT.\*/

if(this.noOfQtrs == report.noOfQtrs

&& this.salesRep.equals(report.salesRep)

&& this.quarterlySales == report.quarterlySales) // By:Frankie Deleon

{

isEqual = true;

}//END if objects have same salesRep,

return isEqual;

}//END equals(report: SalesReport): boolean

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By: Frankie Deleon

\* The setProjectedSales() method asks the user to enter the projected annual sales for Tandem

\* and validates the input to ensure that it is a double. If the input is not a double,

\* the method calls the validateNumber() method to validate the input as a number.

\* Once the input is validated as a double, the method assigns the value

\* to the projectedSales variable. Finally, the method clears the input buffer using the nextLine()

\* method to ensure that any remaining input is removed before proceeding

\*

\*/

public void setProjectedSales()

{

do

{

System.out.printf("%nWhat is the projected annual sales for Tandem? ");

repeat = !input.hasNextDouble(); //Assigns a boolean on whether the input value is a double.

validateNumber(); //A single validation method for numbers regardless of type.

}while(repeat);

projectedSales = input.nextDouble(); //Can safely convert input data to a double.

/\* Clears the input buffer of what's left by next methods for numbers, so a subsequent

\* nextLine() call works properly.\*/

input.nextLine();

}//END setProjectedSales(): void

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* The setNoSalesReps() method prompts the user to input the number of sales representatives that work for Tandem.

\* It validates the user input and returns the value entered as an integer.

\* If the user enters a non-integer value, the method will continue to prompt until

\* a valid value is entered.

\* The value entered is also assigned to the field 'noSalesReps'.

\* returns noSalesReps entered by the user as an integer.

\*/

public int setNoSalesReps()

{

int noSalesReps = 0; //Local variable for returning captured value.

do

{

System.out.printf("%nHow many sales reps work for Tandem? ");

repeat = !input.hasNextInt();

validateNumber();

}while(repeat);

noSalesReps = input.nextInt();

input.nextLine();

this.noSalesReps = noSalesReps; //Assigns to field.

return noSalesReps;

}//END setNoSalesReps(): int

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* The setSalesRep method sets the value of the salesRep field to the value passed as a parameter.

\* Returns nothing.

\*

\*/

//CODE THE HEADER BASED ON THE CLOSE BRACE LINE COMMENT.

public void setSalesRep(String salesRep) //By: Frankie Deleon

{

this.salesRep = salesRep;

}//END setSalesRep(salesRep: String): final void

/\*\*

\* setSalesRep() receives the sales rep count, prompts for the

\* name of the rep and ensures the proper capitalization of the

\* name.

\*/

public void setSalesRep(int salesRepCtr)

{

int index = 0; //Stores the position of a letter in a String object.

String salesRepCopy = null, //Copy of the salesRep's name.

first = null, //First name in salesRep.

last = null; //Last name in salesRep.

char correct = ' '; //Determines the value in the field repeat, a boolean.

do

{

/\* Determines how the prompt is presented. Uses integer

\* value in a ternary operation to determine what goes in

\* the %s for how the prompt will read.\*/

System.out.printf("%nEnter the name of %s sales rep: ",

salesRepCtr == 1 ? "a" : "the next");

salesRep = input.nextLine();

/\*Takes out all spaces in salesRep and stores in the copy variable.\*/

salesRepCopy = new String(salesRep).replace(" ", "");

/\*Resolves a sales rep name that is not an alpha using isAlpha().\*/

while(!isAlpha(salesRepCopy))

{

System.out.printf("%nInvalid name! Please re-enter: ");

salesRep = input.nextLine();

salesRepCopy = new String(salesRep).replace(" ", "");

}//while salesRep's name is NOT alphabetic

/\*Asks user to validate name.\*/

System.out.printf("%nSales Rep Name: %s"

+ "%n%nIs this name correct? \'Y\' or \'N\': ",

salesRep);

correct = input.nextLine().toLowerCase().charAt(0);

/\* A ternary is used to determine the value of repeat based on the

\* value in correct which assesses the correctness of a sales rep's name.\*/

repeat = correct == 'y' ? false : true;

}while(repeat); //do-while sales rep name is not correct

index = salesRep.indexOf(" "); //Locate where the first space is in salesRep.

if(index > 0) //Is there a space? If so, there is another word in the name.

{

/\* Extract the first name from salesRep, uppercase the first letter

\* and add the rest of the letters lowercased.\*/

first = Character.toUpperCase(salesRep.charAt(0))

+ salesRep.substring(1, index).toLowerCase();

/\* CODE BELOW THE EXTRACTION FOR THE last NAME FROM salesRep

\* AND UPPERCASE/LOWERCASE WHERE NEEDED.

\* - THE ARGUMENT FOR charAt() IS index + 1.

\* - THE ARGUMENTS FOR THE substring() IS

\* o index + 2

\* o salesRep.Length()\*/

last = Character.toUpperCase(salesRep.charAt(index + 1))

+ salesRep.substring(index + 2, salesRep.length()).toLowerCase();

/\*CODE BELOW THE STATEMENT TO LOCATE THE FIRST SPACE IN THE LAST NAME.\*/

index = last.indexOf("");

if(index > 0) //If there is a space in the last name, there's another word.

{

/\* Add to the first name a space, then the word from the last name

\* starting at the first letter in the last name until the index for

\* the space.\*/

first += " " + last.substring(0, index);

/\* Now capitalize the first letter in the next word of the last name

\* and add the rest of the letters lowercased.\*/

last = Character.toUpperCase(last.charAt(index + 1))

+ last.substring(index + 2, last.length()).toLowerCase();

}//END if index > 0

/\*Reconstitute the salesRep's name which is now properly capitalized.\*/

salesRep = first + " " + last;

}//if there is more than one name, capitalize the first letter in each

else

{

salesRep = Character.toUpperCase(salesRep.charAt(0))

+ salesRep.substring(1, salesRep.length()).toLowerCase();

}//else capitalize first letter of a single name

}//END setSalesRep(salesRepCtr: int): void

/\*\*

\* setNoQtrs() assigns an incoming value for noOfQtrs

\* to the field. It is final because it is called from

\* a constructor.

\*/

public final void setNoQtrs(int noOfQtrs)

{

this.noOfQtrs = noOfQtrs;

}//END setNoQtrs(noOfQtrs: int): final void

/\*\*

\* setNoQtrs() prompts for the number of quarters a sales

\* rep has worked. The entry is validated for proper

\* conversion to an integer.

\*/

public void setNoQtrs()

{

do

{

System.out.printf("%nEnter the number of quarters worked (no less "

+ "than 1 or greater than 4): ");

repeat = !input.hasNextInt();

if(repeat)

{

validateNumber();

}//END if takes care of invalid integer

else

{

noOfQtrs = input.nextInt();

}//END else converts input to an integer and assigns to field

/\* If the number entered for noOfQtrs is not in the

\* proper range of 1-4, then while-loop corrects it.\*/

while(noOfQtrs < 1 || noOfQtrs > 4)

{

System.out.printf("%nOUT OF RANGE! Re-enter the number of quarters worked (no less "

+ "than 1 or greater than 4): ");

/\*Every time an integer is entered it has to be validated.\*/

repeat = !input.hasNextInt();

if(repeat)

{

validateNumber();

}//END if takes care of invalid integer

else

{

noOfQtrs = input.nextInt();

}//END else converts input to an integer and assigns to field

}//END while noOfQtrs is out of range

}while(repeat); //END do-while repeats when number of quarters is invalid

input.nextLine();

}//END setNoQtrs(): void

/\*\*

\* chooseQtr() prompts for the quarter in which the sales

\* were earned. The entry is validated for proper

\* conversion to an integer before it is returned to the

\* calling statement.

\*/

public void chooseQtr(int qtrCounter)

{

do

{

System.out.printf("%n1. First Quarter"

+ "%n2. Second Quarter"

+ "%n3. Third Quarter"

+ "%n4. Fourth Quarter"

+ "%n%nChoose the %squarter in which sales were earned: ",

qtrCounter > 1 ? "next " : "");

repeat = !input.hasNextInt();

if(repeat)

{

validateNumber();

}//END if takes care of invalid integer

else

{

qtrChoice = input.nextInt();

}//END else converts input to an integer and assigns to field

/\*CODE BELOW FOR while HEADER OUT OF RANGE TEST FOR qtrChoice.\*/

while(qtrChoice < 1 || qtrChoice > 4)

{

System.out.printf("%nOUT OF RANGE!%n"

+ "%n1. First Quarter"

+ "%n2. Second Quarter"

+ "%n3. Third Quarter"

+ "%n4. Fourth Quarter"

+ "%n%nChoose the %squarter in which sales were earned: ",

qtrCounter > 1 ? "next " : ""); //By: Frankie Deleon

repeat = !input.hasNextInt(); //CODE THE ASSIGNMENT TO repeat. By: Frankie Deleon

if(repeat)//CODE THE if TEST OF repeat By: Frankie Deleon

{

validateNumber();//CODE THE CALL TO THE VALIDATE METHOD. By: Frankie Deleon

}//END if takes care of invalid integer

else

{

qtrChoice = input.nextInt();

}//END else converts input to an integer and assigns to field

}//END while qtrChoice is out of range

}while(repeat); //END do-while repeats when qtrChoice is invalid

input.nextLine();//CODE THE CLEAR BUFFER STATEMENT. By: Frankie Deleon

quarter = (qtrChoice == 1) ? "First Quarter"

: (qtrChoice == 2) ? "Second Quarter"

: (qtrChoice == 3) ? "Third Quarter"

: "Fourth Quarter";

}//END chooseQtr(int): void

/\*\*

\* determineMonthNo() receives the monthCounter to

\* figure out which month it is in the quarter:

\* 1st, 2nd, 3rd.

\*/

public void determineMonthNo(int monthCounter)

{

switch(monthCounter)

{

case 1: monthNo = "1st";

break;

case 2: monthNo = "2nd";

break;

case 3: monthNo = "3rd";

}//END switch on monthCounter for 1st thru 3rd

}//END determineMonthNo(monthCounter: int): void

/\*\*

\* setSalesRevenue() asks for the sales revenue earned

\* for a given month within a given quarter. The entry is

\* validated for proper conversion to a double before it

\* is returned to the calling statement.

\*/

public void setSalesRevenue()

{

do

{

System.out.printf("%nEnter the sales revenue for the %s "

+ "month of the %s: ", monthNo, quarter);

repeat = !input.hasNextDouble();

validateNumber();

}while(repeat); //END do-while repeats when richter is invalid

salesRevenue = input.nextDouble();

input.nextLine();

}//END setSalesRevenue(): void

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* setQuarterlySales method sets the value of the quarterlySales field to the value passed as a parameter.

\* Returns nothing.

\*/

public void setQuarterlySales(double quarterlySales)//CODE THE METHOD HEADER BASED ON THE CLOSE BRACE LINE COMMENT. By: Frankie Deleon

{

this.quarterlySales = quarterlySales;

}//END setQuarterlySales(quarterlySales: double): final void

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* The method calculateQtrlySales() adds the current sales revenue

\* to the existing quarterly sales revenue of the sales rep and returns nothing.

\*/

public void calculateQtrlySales()

{

quarterlySales += salesRevenue;

}//END calculateSales(): void

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* The method calculateAnnualSales() adds the quarterly sales revenue

\* to the existing annual sales revenue.

\*/

public void calculateAnnualSales()

{

annualSales += quarterlySales;

}//END calculateSales(): void

/\*\*

\* getSalesRepRevReport() prints the total year-to-date

\* sales revenue earned by a sales rep for the number of

\* quarters worked in the current year.

\*/

public String getSalesRepRevReport()

{

return String.format("%nSALES REVENUE FOR %d QUARTER(S) OF %tY"

+ "%nSALES REP: %s"

+ "%n%nTotal Year-To-Date: $%,19.2f%n",

noOfQtrs, dateTime, salesRep, quarterlySales);

}//END getSalesRepRevReport(): String

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* The getRepTargetMsg() method calculates the sales earned by the sales rep to determine

\* if the sales rep is on target. It then creates a message about year-end bonuses based on

\* the sales rep's sales performance. If the sales rep has less than 4 quarters of sales data,

\* the method generates an encouraging message if the sales are on target,

\* or a warning message if the sales are lagging behind projections.

\* If the sales rep has 4 quarters of sales data, the method will calculate

\* whether or not the sales rep has achieved their annual sales goal,

\* and create an appropriate message. The message is returned as a String.

\* The method uses the String.format() method to create the message with the sales rep's name

\* and a possible year-end bonus message. If the sales rep's name has a space in it, the first name

\* is extracted and capitalized. The method returns the message

\*/

public String getRepTargetMsg()

{

//Calculating the sales earned by the sales rep to see if rep is on target.

percOfTargetRep = quarterlySales/(projectedSales/noSalesReps) \* 100;

String message = ""; //STORES A MESSAGE ABOUT YEAR-END BONUSES.

/\* Based on whether there is a space in the sales rep's name, extract

\* the first name, else, keep name as is.\*/

String rep = salesRep.indexOf(" ") > 0 ?

salesRep.substring(0, salesRep.indexOf(" ")) : salesRep;

if(noOfQtrs < 4) //If the number of quarters is less than 4

{

/\* Is the sales rep's sales on target? If so, an encouraging message

\* is printed; otherwise, a warning of lagging sales. The substring()

\* is used to extract the first name and ensure the first letter is

\* capitalized.

\*/

message = String.format(percOfTargetRep >= 50 ? String.format("%nKeep up the GOOD work, %s. "

+ "There is a possible year-end bonus!%n", rep)

: "%nSo far sales are lagging behind projections.%n");

}//END if noOfQtrs < 4 OR >= 4

return message;

}//END getRepTargetMsg(): String

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\*

\* getCompanyTargetMsg method calculates the percentage of the projected annual sales that the company has achieved so far.

\* It then prints a message to the console indicating whether or not the company is on track to meet

\* its sales goals and if a year-end bonus is possible. If the percentage of target sales achieved is

\* greater than or equal to 100%, a positive message is printed, indicating that a year-end bonus is possible

\* if the company can maintain its sales efforts. If the percentage of target sales achieved is

\* less than 100%, a negative message is printed, warning that a year-end bonus may not be possible

\* if sales do not improve.

\*/

public void getCompanyTargetMsg()

{

//Calculating the annual sales to date to see if the company is on target.

percOfTargetCo = (annualSales/projectedSales) \* 100;

System.out.printf("%nCORPORATE SALES PERFORMANCE%n");

/\* Is the company on target to meet or exceed its projected sales?

\* If so, a year-end bonus message is printed; otherwise, a warning

\* precluding a year-end bonus.

\*/

System.out.printf(percOfTargetCo >= 100 ? "%nIt\'s been a GOOD year so far. "

+ "There could be a year-end bonus of "

+ "%nabout 2-5%% if we can keep on top of our sales goals. Thank you all "

+ "%nand please continue your excellent effort!%n"

: "%nSales are lagging projections. A year-end bonus may "

+ "not be possible.%n");

}//END getCompanyTargetMsg(): void

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* validateNumber method for handling user input when an expected integer or double value is not entered.

\* It uses a boolean variable repeat to determine if the user input is not an integer or double.

\* If repeat is true, it uses the next() method of the Scanner class to consume the non-numeric value

\* that caused the error, and then outputs a warning message that the input was invalid.

\*/

public void validateNumber()

{

if(repeat) //As long as the input is "not" an integer or double.

{

input.next(); //Consumes the non-numeric value that caused the error.

System.out.printf("%nWarning: You entered an invalid integer or "

+ "floating-point value.%n"); //RE-PROMPT

}//END if repeat when a number is an invalid type

}//END validateNumber(): void

/\*\*

\* isAlpha()

\* - accepts a String to analyze.

\* - is final because it is not to be overridden/replaced.

\* - returns a boolean as to whether the word is alphabetic or not.

\*

\* chars() returns the integer values of the characters in word.

\*

\* allMatch determines whether the integer values for each character

\* matches the predicate (criterion) that each character is a letter.

\*

\* The :: is a method reference operator for calling isLetter from

\* the Character class.

\*

\* @param word is the incoming String value to test.

\* @return is true when the word is not empty and is alphabetic

\* or false when it isn't.

\*/

/\*CODE THE METHOD HEADER BASED ON THE CLOSE BRACE LINE COMMENT.\*/

public final boolean isAlpha(String word) //By: Frankie Deleon

{

/\* Test to see if the word is not empty AND if each letter

\* in a word is an alphabetic character.

\*/

return word != null && word.chars().allMatch(Character :: isLetter);

}//END isAlpha(word: String): final boolean

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* The getDate() method accepts no parameters and returns a formatted string of the date as 99/99/9999.

\*/

public String getDate()

{

return String.format("%tD", dateTime); //Returns a formatted date as 99/99/9999.

}//END getDate(): String

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getProjectedSales() method accepts no parameters and returns projectedSales double.

\*/

/\*CODE THE METHOD HEADER BASED ON THE CLOSE BRACE LINE COMMENT.\*/

public double getProjectedSales() //By: Frankie Deleon

{

return projectedSales;

}//END getProjectedSales: double

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getNoSalesReps() method accepts no parameters and returns noSalesReps int.

\*

\*/

/\*CODE THE METHOD HEADER BASED ON THE CLOSE BRACE LINE COMMENT.\*/

public int getNoSalesReps() //By: Frankie Deleon

{

return noSalesReps;//CODE THE return STATEMENT. By: Frankie Deleon

}//END getNoSalesReps(): int

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\* By:Frankie Deleon

\* getSalesRep() method accepts no parameters and reutns salesRep String.

\*

\*/

/\*CODE THE METHOD HEADER BASED ON THE CLOSE BRACE LINE COMMENT.\*/

public String getSalesRep() //By: Frankie Deleon

{

return salesRep;//CODE THE return STATEMENT. By: Frankie Deleon

}//END getSalesRep(): String

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getNoQtrs() method accepts no parameters and reutns noOfQtrs int.

\*/

/\*CODE THE METHOD HEADER BASED ON THE CLOSE BRACE LINE COMMENT.\*/

public int getNoQtrs()//By: Frankie Deleon

{

return noOfQtrs;//CODE THE return STATEMENT. By: Frankie Deleon

}//END getNoQtrs(): int

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getQtrChoice() method accepts no parameters and reutns qtrChoice int.

\*/

/\*CODE THE METHOD HEADER BASED ON THE CLOSE BRACE LINE COMMENT.\*/

public int getQtrChoice() //By: Frankie Deleon

{

return qtrChoice;//CODE THE return STATEMENT. By: Frankie Deleon

}//END getQtrChoice(): int

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getQuarter() method accepts no parameters and reutns quarter String.

\*/

public String getQuarter()

{

return quarter;

}//END getQuarter(): String

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getNoMonths() method accepts no parameters and reutns noOfMonths int.

\*/

public int getNoMonths()

{

return noOfMonths;

}//END getNoMonths: int

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getSalesRevenue() method accepts no parameters and reutns salesRevenue double.

\*/

public double getSalesRevenue()

{

return salesRevenue;

}//END getSalesRevenue(): double

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getQuarterlySales() method accepts no parameters and reutns quarterlySales double.

\*/

public double getQuarterlySales()

{

return quarterlySales;

}//END getQuarterlySales(): double

/\*\*

\* STUDENT INSERTS DESCRIPTION OF WHAT'S GOING ON WITH THE CODE

\* INSIDE THE METHOD.

\*

\* By:Frankie Deleon

\* getAnnualSales() method accepts no parameters and reutns annualSales double.

\*/

public double getAnnualSales()

{

return annualSales;

}//END getAnnualSales(): double

}//END CLASS SalesReport

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Author: Frankie Deleon & John Olsen

\* Course Section: IS-2063-ON2

\* Date: 04/23/2023

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* PROGRAM PURPOSE:

\* By: Frankie Deleon

\* The SalesSystem program generate sales reports for Tandem Enterprises.

\* It prompts the user to enter the sales system and, if the input is 'Y', calls methods to process

\* a sales report, write the sales report data to a file, and check the input file. The program uses

\* methods processSalesReport where A new SalesReport object is created, and its setProjectedSales() method

\* is called to set the projected sales. The setNoSalesReps() method is then called to set the number of

\* sales representatives, and the reports array is initialized with the size variable.

\* A for loop is used to iterate through each sales representative, set their sales data for each

\* quarter and month, and calculate their annual sales. The getSalesRepRevReport() and getRepTargetMsg()

\* methods are called to retrieve the sales representative's revenue report and their target message,

\* respectively. The revenue report and target message are added to the aSalesReport variable,

\* and the variable is added to the salesReports ArrayList. Finally, the for loop prints each report

\* using System.out.printf(), and the getCompanyTargetMsg() method is called to retrieve the company's

\* target message.The writeSalesRepData method writes the sales report data to a file. It prompts the user to enter a filename,

\* opens the file in a PrintWriter, then uses a for-loop to retrieve the sales data from the SalesReport array

\* and format it into a string with specific format specifiers. It then writes this string to the file using the

\* PrintWriter object. If an IOException is thrown, it sets fileError to true and prints an error message.

\* If no file error is present, the method closes the PrintWriter object and prints a success message stating

\* the data has been written to the file. The checkInputFile method checks for the existence of a file, reads its

\* contents, and prints them out. It uses a try-catch block to handle potential exceptions

\* such as file not found or null pointer exceptions when trying to access or read a record.

\*/

import java.util.Scanner; // import scanner By: Frankie Deleon

import java.util.Calendar; // import calendar By: Frankie Deleon

import java.util.ArrayList; // import arraylist By: Frankie Deleon

import java.io.File; // import file By: Frankie Deleon

import java.io.PrintWriter; // import printWriter By: Frankie Deleon

import java.io.IOException; //import IOException By: Frankie Deleon

/\*By: Frankie Deleon

\* SalesSystem class declares the the global variables for the program.

\*/

public class SalesSystem

{ //begin SalesSystem class By: Frankie Deleon

private SalesReport[] reports; // declare private SalesReport array reports By: Frankie Deleon

private Scanner input = new Scanner(System.in); // declare input scanner variable By: Frankie Deleon

private Calendar dateTime; //declare dateTime calendar variable By: Frankie Deleon

private ArrayList<String> salesReports = new ArrayList<String>(); // declare salesReports Arraylist String By: Frankie Deleon

private String fileName = ""; // declare fileName String By: Frankie Deleon

private char count; // declare char count. By: Frankie Deleon

/\*

\* SalesSystem constructor, returns nothing. By: Frankie Deleon

\*/

public void SalesStystem()

{ // begin SalesSystm By: Frankie Deleon

} // end SalesSystem By: Frankie Deleon

/\*By: Frankie Deleon

\* start method prompts users to enter the sales system, Tandem Enterprises to begin the process

\* of starting a sales report, if the input is 'Y' call the following methods, processSalesReport();,

\* writeSalesRepData();, and checkInputFile();

\*/

public void start()

{ // begin start By: Frankie Deleon

/\*By: Frankie Deleon

\* prompt user Priming read to enter the sales system.

\* TANDEM ENTERPRISES

\* Begin the processing of a sales report? Enter 'Y' or 'N':

\*

\* If the answer is N, call exitMessage() to print an exit message

\*/

System.out.printf("%nPriming read to enter the sales system.%n");

System.out.printf("%nTANDEM ENTERPRISES%n");

System.out.printf("%nBegi1n the processing of a sales report? Enter 'Y' or 'N': ");

count = input.nextLine().toUpperCase().charAt(0);

if(count == 'Y') // if statement, if count is equal to Y call the following methods By: Frankie Deleon

{

processSalesReport();

writeSalesRepData();

checkInputFile();

} // end if By: Frankie Deleon

exitMessage(); // exit message By: Frankie Deleon

}// end start method By: Frankie Deleon

/\*By: Frankie Deleon

\* processSalesReport method initializes three local variables, namely size,

\* qtrCounter, and monthCounter, to zero and declares a String variable aSalesReport

\* and initializes it with a header. A new SalesReport object is created, and its setProjectedSales() method

\* is called to set the projected sales. The setNoSalesReps() method is then called to set the number of

\* sales representatives, and the reports array is initialized with the size variable.

\* A for loop is used to iterate through each sales representative, set their sales data for each

\* quarter and month, and calculate their annual sales. The getSalesRepRevReport() and getRepTargetMsg()

\* methods are called to retrieve the sales representative's revenue report and their target message,

\* respectively. The revenue report and target message are added to the aSalesReport variable,

\* and the variable is added to the salesReports ArrayList. Finally, the for loop prints each report

\* using System.out.printf(), and the getCompanyTargetMsg() method is called to retrieve the company's

\* target message.

\*/

public void processSalesReport()

{// begin processSalesReport method By: Frankie Deleon

int size = 0; // declare local size variable and initalize to 0. By: Frankie Deleon

int qtrCounter = 0; // declare local qtrCounter variable and initalize to 0. By: Frankie Deleon

int monthCounter = 0; // declare local monthCounter variable and initalize to 0. By: Frankie Deleon

String aSalesReport = "%n%nTANDEM ENTERPRISES"; // declare aSalesReport string and initialize to "\n\nTANDEM ENTERPRISES" By: Frankie Deleon

SalesReport anotherObj = new SalesReport(); // declare new SalesReport object anotherObj. By: Frankie Deleon

anotherObj.setProjectedSales(); //call setProjectedSales() with new SalesReport object. By: Frankie Deleon

size = anotherObj.setNoSalesReps(); // gather the size and call the setNoSalesReps() method. By: Frankie Deleon

reports = new SalesReport[size]; // array initialized with the size variable By: Frankie Deleon

for(int i = 0; i < size; i++) //

{

qtrCounter = 1; // re-initalize qtrCounter to 1. By: Frankie Deleon

reports[i] = new SalesReport(); //// Create a new SalesReport object and assign it to the current index of the arrayBy: Frankie Deleon

reports[i].setSalesRep(i + 1); // call setSalesRep() by sending it the sum of i + 1 By: Frankie Deleon

reports[i].setNoQtrs(); // call setNoQtrs() By: Frankie Deleon

while(qtrCounter <= reports[i].getNoQtrs()) // while qtrCounter is less than or equal to the call to getNoQtrs()By: Frankie Deleon

{

monthCounter = 1; // reinitialize monthCounter to 1 By: Frankie Deleon

reports[i].chooseQtr(qtrCounter); // call chooseQtr by sending it qtrCounterBy: Frankie Deleon

while(monthCounter <= reports[i].getNoMonths()) //while monthCounter is less than or equal to the call of getNoMonths()By: Frankie Deleon

{

reports[i].determineMonthNo(monthCounter); // Call determineMonthNo() by sending it monthCounterBy: Frankie Deleon

reports[i].setSalesRevenue(); // Call setSalesRevenue().By: Frankie Deleon

reports[i].calculateQtrlySales();// Call calculateQtrlySales()By: Frankie Deleon

++monthCounter;// Pre-increment monthCounter.By: Frankie Deleon

}

++qtrCounter; // Pre-increment qtrCounter.By: Frankie Deleon

}

reports[i].calculateAnnualSales(); //Call calculateAnnualSales(). By: Frankie Deleon

aSalesReport += reports[i].getSalesRepRevReport(); //Add to aSalesReport the call to getSalesRepRevReport().By: Frankie Deleon

aSalesReport += reports[i].getRepTargetMsg(); //Add to aSalesReport the call to getRepTargetMsg().By: Frankie Deleon

salesReports.add(aSalesReport); //add() to the ArrayList salesReports, aSalesReport.By: Frankie Deleon

/\*Re-initialize aSalesReport to the a String literal formatted with 2 line advances at the beginning:

\* “TANDEM ENTERPRISES”.By: Frankie Deleon

\*/

aSalesReport = "%n%n TANDEM ENTERPRISES";

}

/\*enhanced for-loop assign each String object from salesReports to eachReport.By: Frankie Deleon

\*/

for(String eachReport : salesReports)

{

System.out.printf("%n%s", eachReport); // print eachReport By: Frankie Deleon

}

/\*Call getCompanyTargetMsg() using the non-array object for SalesReport By: Frankie Deleon

\*/

anotherObj.getCompanyTargetMsg();

} //// end processSalesReport method By: Frankie Deleon

/\*By: Frankie Deleon

\* writeSalesRepData method writes the sales report data to a file. It prompts the user to enter a filename,

\* opens the file in a PrintWriter, then uses a for-loop to retrieve the sales data from the SalesReport array

\* and format it into a string with specific format specifiers. It then writes this string to the file using the

\* PrintWriter object. If an IOException is thrown, it sets fileError to true and prints an error message.

\* If no file error is present, the method closes the PrintWriter object and prints a success message stating

\* the data has been written to the file.

\*/

public void writeSalesRepData()

{ // begin writeSalesRepData method By: Frankie Deleon

String record; // declare and initialize record String. By: Frankie Deleon

PrintWriter outputFile = null; //declare and initialize PrintWriter outputFile to null. By: Frankie Deleon

boolean fileError = false; // declare and initalize boolean test variable fileError to false. By: Frankie Deleon

try { // start try block By: Frankie Deleon

/\* By: Frankie Deleon

\* prompt user to Enter the file name for sales report (WARNING: This will erase a pre-existing file!):

\*/

System.out.printf("%nEnter the file name for sales report (WARNING: This will erase a pre-existing file!): ");

fileName = input.next(); //capture user input to fileName variable. By: Frankie Deleon

outputFile = new PrintWriter(fileName); //Open the file in PrintWriter. By: Frankie Deleon

/\*By: Frankie Deleon

\* for-loop to get the data from the array into the record variable using

\* these format specifiers: "%s, %d, %s, %.2f%n" in the order of the get

\* methods for the date, number of quarters, sales rep, and quarterly sales.

\*/

for(int i = 0; i < reports.length; i++)

{

record = String.format("%s, %d, %s, %.2f%n",

reports[i].getDate(),

reports[i].getNoQtrs(),

reports[i].getSalesRep(),

reports[i].getQuarterlySales());

outputFile.printf(record);

}

} catch(IOException e) //catch block that catches an IOException into e. By: Frankie Deleon

{

/\*Use the err object from System to call printf() by sending it this message: File cannot be created.

\*/

System.err.printf("%nFile cannot be created.%n");

fileError = true; //Re-initialize fileError to true. By: Frankie Deleon

}

if(!fileError) //a single-selection if, test for there is no fileError. By: Frankie Deleon

{

outputFile.close(); //Close the outputFile By: Frankie Deleon

/\*Print the message: Data written to the Xxxxxxxxxxx file. where the x’s is the name of the file.

\* By: Frankie Deleon

\*/

System.out.printf("Data written to the %s file.%n", fileName);

}

} // End writeSalesRepData method. By: Frankie Deleon

/\*By: Frankie Deleon

\* checkInputFile method that checks for the existence of a file, reads its contents, and prints them

\* out. It uses a try-catch block to handle potential exceptions such as file not found

\* or null pointer exceptions when trying to access or read a record.

\*/

public void checkInputFile()

{ // begin checkInputFile method By: Frankie

String fileName = ""; // declare fileName String varibale. By: Frankie Deleon

File file = null; // declare File variable file. By: Frankie Deleon

String fileRecord = ""; //declare String variable fileRecord. By: Frankie Deleon

boolean fileNotFound = false; // declare boolean variable fileNotFound. By: Frankie Deleon

Scanner inputFile = null; // declare inputFile Scanner. By: Frankie Deleon

try { // being try block By: Frankie Deleon

/\*By: Frankie Deleon

\* Enter SalesReports.txt as the name of the file for this prompt.

\* Enter the name for the sales report file:

\*/

System.out.printf("%nEnter the name for the sales report file: ");

input.nextLine(); //Gather user input. By: Frankie Deleon

fileName = input.nextLine(); // Create the filename with user input. By: Frankie Deleon

file = new File(fileName); // Finish creating the File object with fileName. By: Frankie Deleon

inputFile = new Scanner(file); // Finish creating the Scanner object with file. By: Frankie Deleon

while(inputFile.hasNext()) //Use a while-loop by testing inputFile.hasNext() By: Frankie Deleon

{

fileRecord = inputFile.nextLine(); //read the next line from the inputFile into fileRecord. By: Frankie Deleon

/\*print what was just read. Use a format specifier with a line advance at the beginning. By: Frankie Deleon

\*/

System.out.printf("%n%s", fileRecord);

}

System.out.println(); //Print a blank line using println(). By: Frankie Deleon

}catch(IOException e) { //catch block that catches IOException into e By: Frankie Deleon

/\*err object using printf() sending it this message: File not found! By: Frankie Deleon

\*/

System.err.printf("File not found!%n");

fileNotFound = true; //fileNotFound equals true if catch. By: Frankie Deleon

}catch(NullPointerException e) { // a catch block that catches NullPointerException into e By: Frankie Deleon

/\*err object using printf()sending it this message: Record couldn’t be accessed or read. By: Frankie Deleon

\*/

System.err.printf("Record couldn't be accessed or read.%n");

fileNotFound = true; //fileNotFound equals true if catch. By: Frankie Deleon

}

if(!fileNotFound) //g a single-selection if, test for there is no fileNotFound. By: Frankie Deleon

{

inputFile.close(); // close inputFile. By: Frankie Deleon

}

} // // End checkInputFile method By: Frankie

/\* By: Frankie Deleon

\* exitMessage method prints the exit message "Exiting Sales System."

\*/

public void exitMessage()

{ // begin exitMessage method By: Frankie

System.out.printf("%nExiting Sales System.%n"); //print the exit message Exiting Sales System. By: Frankie Deleon

} // End exitMessage method By: Frankie

} //End SalesSystem class By: Frankie Deleon