# PaintShopMenu.java

**package** differentCoat;

**import** java.util.ArrayList;

/\*\*

\*

\* **@author** Bogdan Pasterak

\* POP Assignment

\* tutor Angela Mercy

\* Letterkenny Training Centre

\* 15/02/2019

\*

\*/

**public** **class** PaintShopMenu {

// vesion for object stream

// private final static String PATH = "customerList.ser";

**private** **final** **static** String ***PATH*** = "customerList.txt";

**public** **static** **void** main(String[] args) {

// load if avaliable customer list

ArrayList<Customer> customerList = CalculateMenu.*loadList*(***PATH***);

// variable to hold the user's response

**int** choise;

// main loop with Menu

**do** {

System.***out***.println("\n------------ Different Coat ---------------------");

System.***out***.println(" \*\*\* Menu Options \*\*\*");

System.***out***.println(" 1. Process and Display Paint Required Calculations");

System.***out***.println(" 2. Search for Customer Measurements");

System.***out***.println(" 3. Remove Customer Measurements");

System.***out***.println(" 4. Display all Customers details");

System.***out***.println(" 5. Edit Customer details");

System.***out***.println(" 6. Write Customer list to file");

System.***out***.println(" 0. Quit");

System.***out***.println("------------------------------------------------------");

// take the choise int between 0 and 6

choise = MyScanner.*getInt06*();

// call to the appropriate method

**switch** (choise) {

**case** 1:

CalculateMenu.*newCustomer*(customerList);

**break**;

**case** 2:

CalculateMenu.*searchCustomer*(customerList);

**break**;

**case** 3:

CalculateMenu.*removeCustomer*(customerList);

**break**;

**case** 4:

CalculateMenu.*displayCustomers*(customerList);

**break**;

**case** 5:

CalculateMenu.*editCustomer*(customerList);

**break**;

**case** 6:

CalculateMenu.*saveList*(customerList, ***PATH***);

**break**;

**case** 0:

System.***out***.println("See you later");

**break**;

}

// repeat until you have chosen 0

} **while** ( choise > 0 );

}

}

# CalculateMenu.java

**package** differentCoat;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.ArrayList;

**public** **class** CalculateMenu {

**private** **static** **final** String ***HEADER*** = "List of clients Different Coat";

**public** **static** **void** newCustomer(ArrayList<Customer> customerList) {

String name, phone;

**char** roomType; // avaliable S,R,C

**double**[] dimensions;

**double** surface = 0;

**int** cans;

// get customer details

System.***out***.println("\n --- Enter details of client ---");

name = MyScanner.*getString*(" Enter the customer's name: ");

phone = MyScanner.*getPhone*(**false**);

roomType = MyScanner.*getRoomType*();

dimensions = MyScanner.*getDimensions*(roomType);

// calculate dependent surface

**switch** (roomType) {

**case** 'S':

surface = SurfaceCalculator.*squareRoom*(dimensions[0], dimensions[1]);

**break**;

**case** 'R':

surface = SurfaceCalculator.*rectangularRoom*(dimensions[0], dimensions[1], dimensions[2]);

**break**;

**case** 'C':

surface = SurfaceCalculator.*cylindricalRoom*(dimensions[0], dimensions[1]);

**break**;

}

// calculate required cans of paint

cans = PaintRequiredCalculator.*numberOfCans*(surface);

// adding new customer

customerList.add(**new** Customer(name, phone, cans));

System.***out***.println("\nAdded " + customerList.get(customerList.size()-1));

}

**public** **static** **void** searchCustomer(ArrayList<Customer> customerList) {

**int** index;

// get avaliable index

index = *getIndexCustomer*(customerList);

// print using toString()

**if** (index != -1)

System.***out***.println("\n" + customerList.get(index));

**else**

System.***out***.println("The list of clients is empty");

}

**public** **static** **void** removeCustomer(ArrayList<Customer> customerList) {

**int** index;

// get avaliable index

index = *getIndexCustomer*(customerList);

// Print and remove client

**if** (index != -1) {

System.***out***.println("\nRemmove " + customerList.get(index));

customerList.remove(index);

}

**else**

System.***out***.println("The list of clients is empty");

}

**public** **static** **void** displayCustomers(ArrayList<Customer> customerList) {

System.***out***.println("\n---- All Customers details ----");

// Print all on list

**for** (Customer customer : customerList) {

System.***out***.println("\n" + customer);

}

}

**public** **static** **void** editCustomer(ArrayList<Customer> customerList) {

// variable

**int** index;

Customer customer;

String name, phone;

Character roomType; // avaliable S,R,C and null

**double**[] dimensions;

**double** surface = 0;

**int** cans;

// customer to edit

index = *getIndexCustomer*(customerList);

**if** (index != -1) {

customer = customerList.get(index);

System.***out***.println(" --- Edit Customer with ID: " + customer.getId() + " ---");

// if you want to change name

System.***out***.println("\n Current name : " + customer.getName());

name = MyScanner.*getStringOrEmpty*(" Type new name (Enter - skip): ");

**if** (name.length() > 0)

customer.setName(name);

// if you want to change phone

System.***out***.println("\n Current phone No : " + customer.getPhone());

phone = MyScanner.*getPhone*(**true**);

**if** (phone != **null**)

customer.setPhone(phone);

// if you want to change calculation

System.***out***.print("\n Current ordered paint cans : " + customer.getPaintCans());

roomType = MyScanner.*getRoomTypeOrEmpty*();

**if** (roomType != **null**) {

dimensions = MyScanner.*getDimensions*(roomType);

// calculate dependent surface

**switch** (roomType) {

**case** 'S':

surface = SurfaceCalculator.*squareRoom*(dimensions[0], dimensions[1]);

**break**;

**case** 'R':

surface = SurfaceCalculator.*rectangularRoom*(dimensions[0], dimensions[1], dimensions[2]);

**break**;

**case** 'C':

surface = SurfaceCalculator.*cylindricalRoom*(dimensions[0], dimensions[1]);

**break**;

}

// calculate required cans of paint

cans = PaintRequiredCalculator.*numberOfCans*(surface);

// and set new value

customer.setPaintCans(cans);

}

System.***out***.println("\nEdited " + customer);

}

**else**

System.***out***.println("The list of clients is empty");

}

**public** **static** **void** saveList(ArrayList<Customer> customerList, **final** String PATH) {

// try / catch with source

/\*

// OO save , data in binary format

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(PATH))){

oos.writeObject(customerList);

System.out.println("The customer list save is successful");

} catch (Exception e) {

System.out.println(e.getMessage());

}

\*/

// change to text record

**try** (PrintWriter printWriter = **new** PrintWriter(PATH)) {

printWriter.println(***HEADER***);

**for** (Customer customer : customerList) {

// seve customer as text line

printWriter.println(customer.toSave());

}

System.***out***.println("Data saved to a file : " + PATH);

} **catch** (Exception e) {

System.***err***.println(e.getMessage());

}

}

**public** **static** ArrayList<Customer> loadList(**final** String PATH) {

// cleate list of customer

ArrayList<Customer> customerList = **new** ArrayList<>();

// try / catch with source

/\*

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(PATH))) {

// read list from File

customerList = (ArrayList<Customer>) ois.readObject();

// get id of last one customer on the list

int last = customerList.get(customerList.size()-1).getId();

// the new customer will have an increases of one ID

Customer.setStartId(last + 1);

System.out.println("List loaded");

} catch (IOException | ClassNotFoundException e) {

System.err.println(e.getMessage());

System.out.println("I create a temporary list with three clients");

customerList.add(new Customer("Jon Doe", "0877-333333", 5));

customerList.add(new Customer("Anna D'ark", "0877-999999", 3));

customerList.add(new Customer("Jeny Mc'Donalds", "0877-111111", 7));

}

\*/

// Input for text file

String customerLine;

**try** (BufferedReader bufferedReader = **new** BufferedReader(**new** FileReader(PATH))) {

**if** (bufferedReader.readLine().equals(***HEADER***))

**while** ((customerLine = bufferedReader.readLine()) != **null**) {

// adding customer, getNewCustomer can throw eception

customerList.add(Customer.*getNewCustomer*(customerLine));

}

**else**

**throw** **new** IOException("This is not a list of Different Coat clients");

System.***out***.println("List loaded");

} **catch** (IncorrectObjectTypeExeption e1 ) {

System.***err***.println(e1.getMessage());

System.***out***.println("I add temporary to list one client");

customerList.add(**new** Customer("Bogdan Pasterak", "087 7000000", 9));

} **catch** (IOException e2) {

System.***err***.println(e2.getMessage());

System.***out***.println("I add temporary to list three clients");

customerList.add(**new** Customer("Jon Doe", "087 7333333", 5));

customerList.add(**new** Customer("Anna D'ark", "087 7999999", 3));

customerList.add(**new** Customer("Jeny Mc'Donalds", "087 7111111", 7));

}

**return** customerList;

}

// get

**private** **static** **int** getIndexCustomer(ArrayList<Customer> customerList) {

**int** idToSearch;

**if** ( customerList.size() > 0)

// until you enter the correct id number

**do** {

idToSearch = MyScanner.*getInt*("Enter ID customer: ");

**for** (**int** i = 0; i < customerList.size(); i++)

**if** (customerList.get(i).getId() == idToSearch) {

**return** i;

}

System.***out***.println("There is no client with id " + idToSearch);

} **while** ( **true** );

**else**

System.out.println("There is no clients on the list ");

**return** -1;

}

}

# Customer.java

**package** differentCoat;

**import** java.io.Serializable;

**public** **class** Customer **implements** Serializable {

// number for Serializable

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

// class variable

**private** **static** **int** *idLastCustomer*;

// constant id for each customer

**private** **final** **int** id;

// members variable

**private** String name;

**private** String phone;

**private** **int** paintCans;

// constructor to initialize ID customera

**private** Customer() {

*idLastCustomer*++;

id = *idLastCustomer*;

}

// constructor to create new customer

**public** Customer(String name, String phone, **int** paintCans) {

**this**();

setName(name);

setPhone(phone);

setPaintCans(paintCans);

}

// getters and setters

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

// replace the tab with four spaces, the tab is a separator when writing data

**this**.name = name.trim().replaceAll("\t", " ");

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

// replace the tab with four spaces, the tab is a separator when writing data

**this**.phone = MyScanner.*formatPhoneNo*(phone);

}

**public** **int** getPaintCans() {

**return** paintCans;

}

**public** **void** setPaintCans(**int** paintCans) {

**if** (paintCans < 0 || paintCans > 100 )

paintCans = 0;

**this**.paintCans = paintCans;

}

// id only getter

**public** **int** getId() {

**return** id;

}

// method needed to read the list as an object

**public** **static** **boolean** setStartId(**int** start) {

// greater than 0

**if** (start <= 0)

**return** **false**;

// it was already initialized, you can only once

**if** (*idLastCustomer* != 0)

**return** **false**;

// constructor incrase

start--;

*idLastCustomer* = start;

// success

**return** **true**;

}

// method to convert customer to 1 line of string (tab is a separator

**public** String toSave() {

**return** id + "\t" + name + "\t" + phone + "\t" + paintCans;

}

// method return new customer from String (loaded) if correct

**public** **static** Customer getNewCustomer(String customer) **throws** IncorrectObjectTypeExeption {

// seperate data

String[] customerDetails = customer.trim().split("\t");

// if number data match

**if** (customerDetails.length == 4) {

**try** {

// id and paintCans are int type

**int** nrId = Integer.*parseInt*(customerDetails[0]);

**int** cans = Integer.*parseInt*(customerDetails[3]);

// validate phone

String phoneNo = MyScanner.*formatPhoneNo*(customerDetails[2]);

// if a gap in id numbers leave a gap

**if** (*idLastCustomer* + 1 < nrId )

*idLastCustomer* = nrId - 1;

**else** **if** (*idLastCustomer* >= nrId)

**throw** **new** IncorrectObjectTypeExeption("Wrong order of customers on the list");

**if** (phoneNo == **null**)

**throw** **new** IncorrectObjectTypeExeption("Wrong phone No customer " + customerDetails[1]);

// return new customer

**return** **new** Customer(customerDetails[1], phoneNo, cans);

} **catch** (NumberFormatException e) {

**throw** **new** IncorrectObjectTypeExeption("Data type not compatible with Customer data");

}

} **else**

**throw** **new** IncorrectObjectTypeExeption("Data number doesn't match the Customer type");

//return null;

}

@Override

**public** String toString() {

String customer;

customer = "Customer ID - " + id + "\n";

customer += " Name - " + name + "\n";

customer += " Phone Nr - " + phone + "\n";

customer += " Ordered paint cans - " + paintCans;

**return** customer;

}

}

# SurfaceCalculator.java

**package** differentCoat;

**public** **class** SurfaceCalculator {

// calculating the wall area for painting in a square room

**public** **static** **double** squareRoom(**double** height, **double** length) {

// 4 walls with dimension height and side

**return** 4 \* height \* length;

}

// calculating the wall area for painting in a rectangular room

**public** **static** **double** rectangularRoom(**double** height, **double** length, **double** width) {

// 2 walls width by height and 2 walls lenght by height

**return** 2 \* height \* ( width + length ) ;

}

// calculating the wall area for painting in a cylindrical room

**public** **static** **double** cylindricalRoom(**double** height, **double** diameter) {

// one wall with height and length 2 \* r \* pi

**return** height \* diameter \* Math.***PI***;

}

}

# PaintRequiredCalculator.java

**package** differentCoat;

**public** **class** PaintRequiredCalculator {

// On the basis of the surface to be painted, he calculates the amount of 5-liter cans

**public** **static** **int** numberOfCans ( **double** surface ) {

// there are no negative surfaces

**if** (surface < 0)

**return** 0;

// divide by 14 rounded into full cans

**return** (**int**) Math.*ceil*(surface / 14);

}

}

# IncorrectObjectTypeExeption.java

**package** differentCoat;

**public** **class** IncorrectObjectTypeExeption **extends** Exception {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** IncorrectObjectTypeExeption() {

**super**();

}

**public** IncorrectObjectTypeExeption(String message) {

**super**(message);

}

}

# MyScanner.java

**package** differentCoat;

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** MyScanner {

**private** **static** Scanner *scanner* = **new** Scanner(System.***in***);

**private** **static** String *prefix*;

**private** **static** String *subscriber*;

// get int form 0 to 6

**public** **static** **int** getInt06() {

**int** answer = -1;

// between 0 and 6

**do** {

answer = *getInt*("Enter number ( 0 - 6 ): ");

**if** (answer < 0 || answer > 6)

System.***out***.println("Number outside the allowed range.");

} **while** (answer < 0 || answer > 6);

**return** answer;

}

// get int

**public** **static** **int** getInt(String message) {

String enter;

**int** answer = -1;

// loop until get correct value

**do** {

System.***out***.print(message);

enter = *scanner*.nextLine();

**if** (enter.trim().isEmpty())

System.***out***.println("You have to choose some options.");

**else** {

**try** {

answer = Integer.*parseInt*(enter);

} **catch** (NumberFormatException e) {

System.***out***.println("You must give the number.");

}

}

} **while** (answer == -1);

**return** answer;

}

// get Srting ( not empty )

**public** **static** String getString(String message) {

String enter;

**do** {

System.***out***.print(message);

enter = *scanner*.nextLine();

**if** (enter.trim().isEmpty())

System.***out***.println("You have to enter some text.");

} **while** (enter.trim().isEmpty());

**return** enter;

}

// only s, r or c character avaliable, lower or upper case

**public** **static** **char** getRoomType() {

String enter;

**char** room = 'X';

System.***out***.println("\n --- Choice of room type ---");

System.***out***.println(" S: Square shape room");

System.***out***.println(" R: Rectangular shape room");

System.***out***.println(" C: Cylindrical shape room");

**do** {

System.***out***.print(" Enter type of room: ");

enter = *scanner*.nextLine().trim().toUpperCase();

**if** (enter.isEmpty())

System.***out***.println(" You have to enter some type.");

**else** {

room = enter.charAt(0);

**if** (room != 'S' && room != 'R' && room != 'C')

System.***out***.println(" We do not have that type.");

}

} **while** (room != 'S' && room != 'R' && room != 'C');

**return** room;

}

// get 2 or 3 dimensions of room

**public** **static** **double**[] getDimensions(**char** roomType) {

**double**[] dimensions;

String[] dimensionsStrings;

String message, enter;

**boolean** ok = **false**;

System.***out***.println("\n --- Enter of dimensions ---");

// Appropriate message

**switch** (roomType) {

**case** 'S':

message = " Enter height and length ( h.h l.l ): ";

**break**;

**case** 'R':

message = " Enter height, length and width ( h.h l.l w.w ): ";

**break**;

**case** 'C':

message = " Enter height and diameter ( h.h d.d ): ";

**break**;

**default**:

message = " No Data";

}

// until type proper value

**do** {

System.***out***.print(message);

enter = *scanner*.nextLine().trim();

**if** (enter.isEmpty())

System.***out***.println(" You have to enter some dimensions.");

**else** {

// Stream - new Class Java 8

// build String array

dimensionsStrings = Arrays

.*stream*(enter.split("[- /:;,\*\\t\\\\]"))

.filter(s -> s.length()>0)

.toArray(String[]::**new**);

// number of variable

**if** ((dimensionsStrings.length == 2 && (roomType == 'S' || roomType == 'C'))

|| (dimensionsStrings.length == 3 && roomType == 'R' )) {

ok = **true**;

// if all are numbers

**for** (String dimension : dimensionsStrings) {

ok = ok && *isDouble*(dimension);

}

**if** ( ok ) {

// message

System.***out***.print(" You typed : ");

dimensions = **new** **double**[dimensionsStrings.length];

**for** (**int** i = 0; i < dimensionsStrings.length; i++) {

dimensions[i] = Double.*parseDouble*(dimensionsStrings[i]);

System.***out***.print(dimensions[i]);

**if** ( i + 2 == dimensionsStrings.length)

System.***out***.print(" and ");

**else** **if** ( i + 1 == dimensionsStrings.length)

System.***out***.print(" .");

**else**

System.***out***.print(" , ");

}

System.***out***.println();

**return** dimensions;

} **else**

System.***out***.println(" Data is not numbers.");

} **else**

System.***out***.println(" Incompatible data quantity.");

}

} **while** ( ! ok );

**return** **null**;

}

// test of String is double

**private** **static** **boolean** isDouble(String dimension) {

**boolean** is = **true**;

**try** {

Double.*parseDouble*(dimension);

} **catch** (NumberFormatException e) {

is = **false**;

}

**return** is;

}

**public** **static** String getStringOrEmpty(String message) {

System.***out***.print(message);

**return** *scanner*.nextLine().trim();

}

// only s, r, c or null Character avaliable

**public** **static** Character getRoomTypeOrEmpty() {

String enter;

**char** room = 'X';

// mini menu

System.***out***.println("\n --- Choice of room type (Enter - skip)---");

System.***out***.println(" S: Square shape room");

System.***out***.println(" R: Rectangular shape room");

System.***out***.println(" C: Cylindrical shape room");

**do** {

System.***out***.print(" Enter type of room: ");

enter = *scanner*.nextLine().trim().toUpperCase();

**if** (enter.isEmpty())

// skip

**return** **null**;

**else** {

room = enter.charAt(0);

// wrong option

**if** (room != 'S' && room != 'R' && room != 'C')

System.***out***.println(" We do not have that type.");

}

} **while** (room != 'S' && room != 'R' && room != 'C');

**return** room;

}

// get valid phone No

**public** **static** String getPhone(**boolean** canSkip) {

String enter;

**do** {

System.***out***.print(" Enter the customer's phone No (0XX XXX XXXX)" +

((canSkip) ? " (Enter - skip): ": ": " ));

enter = *scanner*.nextLine().trim();

**if** (enter.isEmpty()) {

**if** (canSkip)

**return** **null**;

**else**

System.***out***.println(" You have to enter some phone No.");

} **else** {

**if** (*validatePhoneNo*(enter))

**return** *formatPhoneNo*(enter);

**else**

System.***out***.println("Invalid number.");

}

} **while** (**true**);

}

// https://en.wikipedia.org/wiki/Telephone\_numbers\_in\_the\_Republic\_of\_Ireland

**public** **static** **boolean** validatePhoneNo(String phone) {

phone = phone.replaceAll("[- /.:;,\*\\t()]", "");

**if** (phone.length() < 6)

**return** **false**;

// change prefix to '0'

**if** ( phone.startsWith("00353") )

phone = "0" + phone.substring(5);

**else** **if** ( phone.startsWith("+353") )

phone = "0" + phone.substring(4);

**if** ( ! phone.matches("^[\\d]{7,11}$") )

// if (phone.charAt(0) != '0')

**return** **false**;

// Dublin

**if** (phone.matches("^(01).{5,7}$")) {

*prefix* = phone.substring(0, 2);

*subscriber* = phone.substring(2);

**return** **true**;

// more option

} **else** **if** (phone.matches("^(02[1-9]).{5,7}$")) {

*prefix* = phone.substring(0, 3);

*subscriber* = phone.substring(3);

**return** **true**;

} **else** **if** (phone.matches("^(040[24]).{5,7}$")) {

*prefix* = phone.substring(0, 4);

*subscriber* = phone.substring(4);

**return** **true**;

} **else** **if** (phone.matches("^(04[1-79]).{5,7}$")) {

*prefix* = phone.substring(0, 3);

*subscriber* = phone.substring(3);

**return** **true**;

} **else** **if** (phone.matches("^(050[45]).{5,7}$")) {

*prefix* = phone.substring(0, 4);

*subscriber* = phone.substring(4);

**return** **true**;

} **else** **if** (phone.matches("^(05[1-9]).{5,7}$")) {

*prefix* = phone.substring(0, 3);

*subscriber* = phone.substring(3);

**return** **true**;

} **else** **if** (phone.matches("^(06[1-9]).{5,7}$")) {

*prefix* = phone.substring(0, 3);

*subscriber* = phone.substring(3);

**return** **true**;

} **else** **if** (phone.matches("^(07[14]).{5,7}$")) {

*prefix* = phone.substring(0, 3);

*subscriber* = phone.substring(3);

**return** **true**;

} **else** **if** (phone.matches("^(09[0-9]).{5,7}$")) {

*prefix* = phone.substring(0, 3);

*subscriber* = phone.substring(3);

**return** **true**;

// mobile

} **else** **if** (phone.matches("^(08[35679]).{7}$")) {

*prefix* = phone.substring(0, 3);

*subscriber* = phone.substring(3);

**return** **true**;

}

**return** **false**;

}

// change format phone No to proper if not return null

**public** **static** String formatPhoneNo(String phone) {

**if** ( *validatePhoneNo*(phone))

**return** *prefix* + " " + *subscriber*.substring(0, 3) + " " + *subscriber*.substring(3);

**else**

**return** **null**;

}

}