Software Documentation

# Requirements

* PC
* Eclipse ( Java environment )
* Word

# Design

The entire software consists of two packages and related documentation and a set of screenshots. Package “differentCoat” contains classes required for the operation of the program.

* Class “PaintShopMenu” with main method.
* Class “CalculateMenu” contains the methods you call in the menu.
* Class “SurfaceCalculator” a set of methods for calculating the surface to be painted.
* Class “PaintRequiredCalculator” – “numberOfCans” method calculates the amount of paint cans needed
* Class “Customer” - class of Customer objects.
* Class “MyScanner” - methods for communication with the user.
* Class “IncorrectObjectTypeExeption” - exceptions when reading a file from a file.

Package “differentCoatTest”. It consists of a JUnit Test Case and a JUnit Test Suite grouping them.

* JUnit Test Suite “AllTests”.
* Junit Test Case “PaintRequiredCalculatorTest” – test “numberOfCans” method.
* Junit Test Case “SurfaceCalculatorTest” – test methods of “SurfaceCalculator”.
* Two Junit Test Case validate and format phone No in “MyScanner”.

# Technical

### Package “differentCoat”

1. Class “PaintShopMenu”. The program starts work here. The "main" method creates an ArrayList object containing client data stored on disk. Then it enters the loop in which Menu is displayed, the choice is read and the appropriate method is called. The choice of the method takes place using the switch structure. Loop repeats until we choose the "0" Quit option. After leaving the program, the work ends.
2. Class “CalculateMenu”. Contains methods called in “PaintShopMenu”.
   1. “newCustomer” - Joins a new customer list sent by reference letters. Customer data is collected from the user via the keyboard, using the methods contained in the “MyScanner” class.
   2. “searchCustomer” - Prints the data from the list about the selected customer on the screen. The customer's index in the list is retrieved from the auxiliary search method “getIndexCustomer” for this index based on the customer ID.
   3. “removeCustomer” - It allows you to remove a client from the list. Like “searchCustomer”, he uses “getIndexCustomer”.
   4. “displayCustomers” - Prints all clients currently on the list you have sent. It was possible to internalize the Enhanced type of loop. A single client is printed using the "toString" method.
   5. “editCustomer” - This method allows you to edit customer data. The exception is ID assigned to "rigid" to the client. The individual data can be edited or left unchanged by pressing Enter.
   6. “saveList” - Another method from the menu. It allows you to save a list with current customer data to a file.
   7. “loadList” - This method is called automatically when the program starts. In the event of a problem with the data being read to the level, it creates an entire list with three sample clients. If the problem occurred at the level of one client, it adds one example to already read.
   8. “getIndexCustomer” - This is a private auxiliary method. Searches the customer's index in the list with the given ID. If it does not exist, it asks you to enter the ID again.
3. Class “SurfaceCalculator”. It contains three methods for three types of rooms. Each of them calculates the surfaces of the walls to be painted. Returns a value of double type expressing surface in square meters.
   1. “squareRoom” - Calculates the surfaces of four walls with dimensions height and length.
   2. “rectangularRoom” - The rectangular room consists of two opposite wall pairs. Input data is height, length and width type of double.
   3. “cylindricalRoom” - This is actually one wall (rectangle) whose length is equal to the circumference of the wheel. To calculate the surface, we use the height, diameter, and value of Pi from the standard Math library.
4. Class “PaintRequiredCalculator” – method “numberOfCans” - From a given area (double), it calculates the number of required paint cans needed to cover it. Returns the value of type int.
5. Class “Customer” - This is a class of Customer objects. Each customer has four variables.
   1. “id” – final int, unique value given once to a new customer.
   2. “name” – String, name of customer.
   3. “phone” – String, validated and formatted phone number.
   4. “paintCans” – int, the number of cans of paint ordered.

The class also has a static variable "idLastCustomer" used to give customers subsequent ID numbers.

The private default constructor is used to give the new client ID.

Constructor with parameters for security uses setters.

In addition to standard getters and setters ("id" only getter) there are several methods.

1. “setStartId” – static sets the start value “idLastCustomer” once.
2. “toSave” - converts object data into text lines separated by tabs (to write in text files).
3. “getNewCustomer” - Returns references to a new object created on the basis of a text line. In the event of incompatibility, the data throws an exception. Used when reading data.
4. Override “toString” – Returns String to display customer data
5. Class “IncorrectObjectTypeExeption” - inherits after exceptions, has two constructors referencing the base classes.
6. Class “MyScanner” - service class for communication with the user.
   1. “getInt06” - gets the int value from the user in the range from 0 to 6. Options in the menu.
   2. “getInt” - gets the int value from the user. It displays the message given in the parameter.
   3. “getString” - like the previous one but returns String.
   4. “getRoomType” - displays the menu. Gets a char describing the type of room. Accepts small and large letters.
   5. “getDimensions” - displays prompt dependent on the parameter (room type). Takes from the user String and converts it to a double array describing the dimensions of the rooms.
   6. “isDouble” - checks if the input String can be converted to double. Returns boolean.
   7. “getStringOrEmpty” – displays prompt. Returns String, allows empty.
   8. “getRoomTypeOrEmpty” – like “getRoomType”, allows null value (Character class).
   9. “getPhone” - Retrieves a phone number from the user, validates, formats. It allows no response if the (boolean canSkip) parameter allows it.
   10. “validatePhoneNo” - checks the correctness of the telephone number.
   11. “formatPhoneNo” - formats the phone number.

### Package “differentCoatTest”

1. JUnit Test Suite “AllTests” - Runs all JUnit Test Case and produces a group report.
2. JUnit Test Case “CustomerCreateTest” - checks the correctness of the constructor action. The order of the ID numbers given to the objects and the possibility of setting the starting number.
3. JUnit Test Case “CustomerGetterSetterTest” – testing getters, setters, “toSave” method and “getNewCustomer”. The latter is checked in many ways. The reason is its complexity, validation of data and throwing of exceptions.
4. JUnit Test Case “CustomerToStringTest” - checks the format of the String returned by the "toString" method.
5. JUnit Test Case “FormatPhoneNoTest” - Correct formatting of the telephone number. Includes sending an incorrect number (return null).
6. JUnit Test Case “PaintRequiredCalculatorTest” - calculating the number of paint cans (int) on the given surface (double). Checks limit values ​​as well as values ​​not allowed (negative)
7. JUnit Test Case “SurfaceCalculatorTest” - checks the correctness of calculations with different data of three methods for calculating the surface of painting in rooms.
8. JUnit Test Case “ValidatePhoneNoTest” – method from “MyScanner” - checks different configurations of telephone numbers in Ireland. Specification consistent with the data provided on the page: <https://en.wikipedia.org/wiki/Telephone_numbers_in_the_Republic_of_Ireland> .