

25/05/2021



JAVA CAR PROGRAMME

COURSEWORK 2 GRAPHICAL USER INTERFACE

**ARKADIUSZ GRUDZIEN** STUDENT ID: 20001306

MODULE LEADER: **SANDRA FERNANDO**

MODULE NAME: **PROGRAMMING CS4001**

PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed below. Please sign below to say that you have read and understand these extracts:

(Signature:) Arkadiusz Grudzien Date: 25/05/2021

This header sheet should be attached to the work you submit. No work will be accepted without it.

Extracts from University *Regulations on*

Cheating, Plagiarism and Collusion

Section 2.3: "The following broad types of offence can be identified and are provided as indicative examples …..

1. **Cheating: including taking unauthorised material into an examination; consulting unauthorised material outside the examination hall during the examination; obtaining an unseen examination paper in advance of the examination; copying from another examinee; using an unauthorised calculator during the examination or storing unauthorised material in the memory of a programmable calculator which is taken into the examination; copying coursework.**
2. **Falsifying data in experimental results.**
3. Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.
4. **Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.**
5. Collusion to present joint work as the work solely of one individual.
6. Plagiarism, where the work or ideas of another are presented as the candidate's own.
7. Other conduct calculated to secure an advantage on assessment.
8. Assisting in any of the above.

Some notes on what this means for students:

1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.

Taking extracts from published sources *without attribution* is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Stating an author’s argument and attributing it, perhaps by quoting, achieve quoting ideas

Table of Contents

[Introduction 2](#_Toc72867803)

[Basic description of the whole program 2](#_Toc72867804)

[Class diagram performing whole structure of the program. 3](#_Toc72867805)

[Description 3](#_Toc72867806)

# Introduction

Graphical User Interface is form of user interface that allows user to interact with electronic devices through graphical icons and even in advanced coding has sound and other inputs from user such as drawing with pen. It makes the life easer for people who does not know how to user the command line interface.

The goal for this project is to create a Graphical User Interface for already created classes in the first part of the coursework. GUI must allow user to add the car to buy and to rent. It needs to have also an option to rent the car and buy the car and return the car. This program according to the description must be able to open in BlueJ or NetBeans. The first part of coursework I have chosen a BlueJ for creating it. This time I went step ahead and chosen different program for creating it. This editor is called Visual Studio Code.

Java is an object-oriented programming language where significant role has classes. Java is a language for creating programs compilating to binary code what computer understands.

# Description of the program

When the program is starting firstly displaying the interactive menu. In the menu user can choose options. The all entities (file, add, edit, client, info) from the menu bar has advantage of expanding it to specific buttons. File option is for that the user can close the program. Add is for adding a car for buy and rent. When the car to buy button is clicked then new window is appearing and displaying fields for entering as follows: Car’s name. description, price, registration year, mileage. When button called “Add” is clicked then the car is added to array and to the file(carToBuy.dat) which is creating automatically. Afterwards the windows disappear. The same happens with car rent button but there is a different data added to the array and to the file (carsToRent.dat). The feature that the program creating the files is useful. It allows for the user to see entered data even if the program runs again.

In the add menu is Edit button, additional thing if user wants to change some details of a car to buy. Client option is for the client user where is possible to buy the car, rent the car and return the car. Info is just an information about the program.

# Code description

The structure of the code has 12 classes. Three classes Car, CarToBuy, CarToRent, has been recreated with few changes. The console data, which was previously displayed in the console, now is displayed in the GUI at certain points. The main method is called CarCompany.java where is created new object of the class called Menu.java. Then all the buttons in the menu creating new windows by creating new specific objects such as: EditCarToBuy.java, AddCarToBuy.java, AddCarToRent.java, BuyCar.java, RentCar.java and ReturnCar.java.

# Class diagram performing whole structure of the program.

To create the diagram, I had simply use UML extension for BlueJ. The arrows I have added in Adobe Photoshop.

Figure 1 Class diagram showing the inheritance relationship between the classes

|  |  |
| --- | --- |
| **Method/Constructor** | **Description** |
| **class Car** | |
| Car(String newDescription) | This is a constructor for car class. Its taking one parameter description. Its making the object of the car class. |
| getDescription() | This is accessor method for getting the description. |
| getCustomerName() | This is accessor method for getting the customer’s name. |
| printCarDetails() | This method displaying the description and customer’s name of the car also is used in other classes in the program. |
| setCustomerName() | Method for setting the customer’s name. |
| **class CarToBuy** | |
| CarToBuy(String description, double newPrice, int newRegYear, int newMileage) | This is a constructor for car to buy to create an object of the car class. It is taking 3 parameters (description, price and mileage) |
| getRegYear() | This is accessor method for getting the registration year of the car. |
| getMileage | This is accessor method for getting the mileage of the car. |
| getSold() | This is accessor method for getting nor the car was sold. (true or false) |
| getPrice() | This is accessor method for getting the car’s price |
| displayDetails() | This method, if the car was not sold displaying car’s information about the car for sell, the description, the customer’s name (if applicable) cost of the car, the registration year and mileage otherwise informing that the car was sold. To print the customer’s name and description the printDetails method is called form Car class. |
| setPrice() | Method for setting the price for the car. |
| buyTheCar() | This method allowing to buy the car. If the car is already sold its displaying message about that otherwise displaying message that the car has been successfully bought. |
| **class CarToRent** | |
| CarToRent(String description, int newAdminFee, int newDailyRate) | This constructor is for creating an object car to rent, it is taking three parameters description (from superclass), administration fee and daily rate. |
| getRentalDate() | This method returning starting rental date of the car as a string. |
| getReturnDate() | This method returning finishing date of the rental. |
| getDescription() | This method is returning brief description of car for rent. |
| getAdminFee() | This method returning administration fee which customer must pay to rent the car. |
| getDailyRate | This method returning daily charge for the car. |
| getNumberOfDays() | This method retuning for how many days the car has been rented. |
| getTotalAccumulated() | The total accumulated represents the total amount paid by all customers to date. |
| getOnLoan | This method retuning nor the car is on loan (true or false). |
| setDailyRate(int newDailyRate) | This method allowing to set the daily rate. |
| setAdminFee(int newAdminFee) | This method allowing to set the administration fee. |
| rentTheCar(String customerName, String rentalDate, String returnDate) | If the car is already rented its displaying information that the car has been rented and when will be returned otherwise it is assigning the customer’s name, the rental date and the return date also is calculating how much the customer will be paying. It’s also counting the number of days between the rental date and return date by using the daysBetween method. |
| daysBetween(String rentalDate, String returnDate) | This is additional method for counting number of days between rental and return date.  For this method I had to import additional java libraries. |
| returnTheCar() | If the car was rented, it is assigning the default parameters of the car to rent and displaying that the car has been returned otherwise its obviously saying that the car has been not rented. |
| rentCarDescription() | The method showing description and total amount (total accumulated) paid by all customers to date |
| printCarToRentDetails() | This method displaying information about the car to rent by calling method from superclass printCarDetails. Its administration fee and daily rate too. It’s also displaying, if the car was rented, its rental date and return date and for how many days is rented. |