Distributed Systems

Assignment 2

Marius Pop

Group 30444

Table of Contents

[1. Requirements 2](#_Toc466727243)

[2. Conceptual architecture of the distributed system 2](#_Toc466727244)

[3. UML Deployment diagram 3](#_Toc466727245)

[4. Build and execution 3](#_Toc466727246)

[4.1. Build 3](#_Toc466727247)

[4.2. Execution 3](#_Toc466727248)

# Requirements

Design, implement and test a client – server distributed system that uses RPC to compute taxes and selling prices for cars.

# Conceptual architecture of the distributed system

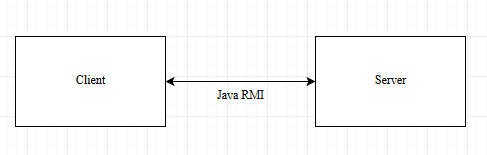
The architecture on which the system is based is the client – server architecture. The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

In this system, the communication between the client and the server is done by using Java RMI. Java RMI is a Java API that performs remote method invocation, the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage collection.

On the server, we have the implementation of some services and one or more clients remotely invoke the methods by using an interface. Both the client and the server must have the same interface for the services, but the service is implemented on the server side. The server gets some inputs, makes the computation, and sends to the client the result.

On the client side, there is the graphical user interface and an interface for the service implemented on the server side.

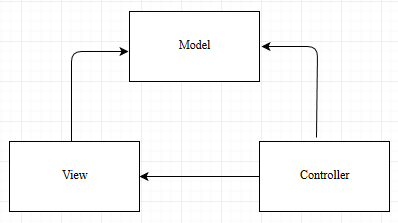
In the picture, presented below, you can see the conceptual architecture of the system.



On the client side, the MVC design pattern was used. This pattern separates the application into three parts:

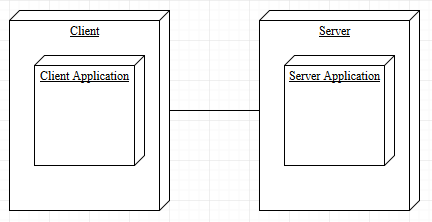
* The model - Model represents and object carrying data. It can also contain logic.
* The view – View represents the visualization of the data.
* The controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

In the picture, presented below, you can see the conceptual architecture of the client.



# UML Deployment diagram

Due to the fact that the application is a distributed one, the server application can be run on one system and the client application can be run on one or more systems.



# Build and execution

## Build

In order to build the application, you need to install Eclipse IDE.

Import the server application and the client application into the workspace.

First, run the server application. A message should be displayed in the console, stating that the server has started.

Next, run the client application. The GUI should be displayed.

## Execution

In order to use the application, the server should be started. If the server is not started, follow the instructions from the previous step to start it.

Now start the client application. The GUI should be displayed.

If you want to calculate the tax for your car, it is enough to input the engine size of your car. Now click on the „Compute Tax” button. The result, should be displayed below the buttons.

If you want to compute the selling price of your car, you have to input the fabrication year of the car an the purchasing price. Now click on the „Compute Selling Price” button. The result, should be displayed below the buttons.