Assignment Two

1. Requirements

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

* 1. Functional requirements
* Users introduce the information of their cars using a simple form (Web or Desktop):
* int year – fabrication year
* int engineSize – engine size
* double price- purchasing price
* The application uses RPC to send the car information to the distributed object from the server that computes the following information depending on the client request:
  + - Tax for a car:

Where sum depends on the engine size:

|  |  |
| --- | --- |
| Engine size | Sum |
| < 1600 | 8 |
| 1601-2000 | 18 |
| 2001-2600 | 72 |
| 2601-3000 | 174 |
| >3001 | 290 |

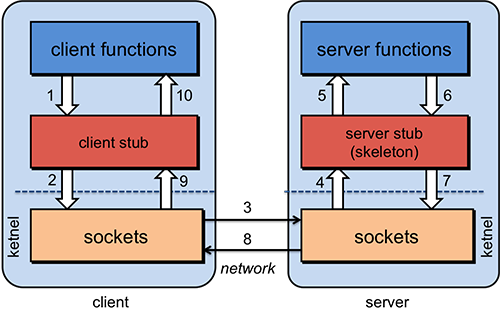
* + - Selling price for a car
* The result of the invoked operation, tax, respectively selling price, is displayed on the client GUI.
  1. Implementation technologies

Use the following technologies: JAVA RMI or .NET Remoting.

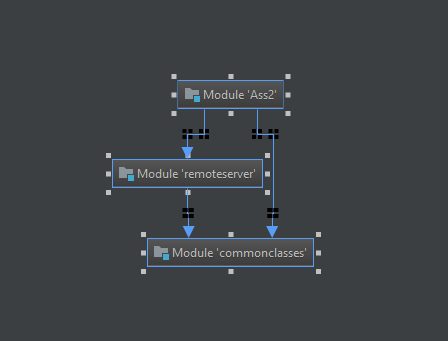
1. Conceptual architecture of the distributed system

* The system uses the layers architectural design pattern. It divides the application in three layers: the presentation layer, which is used to handle the front end of the application, the business layer which handles all the logic of the system (input handling) and the data layer which handles the connection and access to the database (the DAO’s).
* In distributed computing, a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in a different address space (commonly on another computer on a shared network), which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction. That is, the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote.[1] This is a form of client–server interaction (caller is client, executor is server), typically implemented via a request–response message-passing system. In the object-oriented programming paradigm, RPC calls are represented by remote method invocation (RMI). The RPC model implies a level of location transparency, namely that calling procedures is largely the same whether it is local or remote, but usually they are not identical, so local calls can be distinguished from remote calls. Remote calls are usually orders of magnitude slower and less reliable than local calls, so distinguishing them is important.

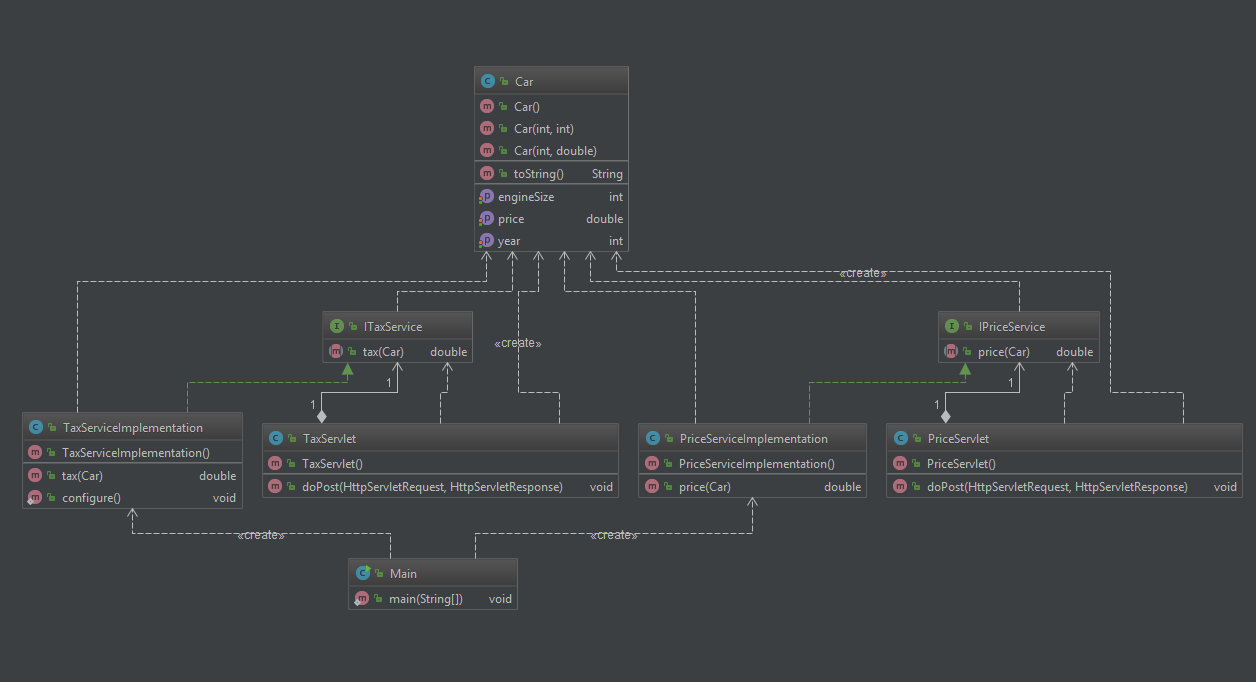
RPCs are a form of inter-process communication (IPC), in that different processes have different address spaces: if on the same host machine, they have distinct virtual address spaces, even though the physical address space is the same; while if they are on different hosts, the physical address space is different. Many different (often incompatible) technologies have been used to implement the concept.



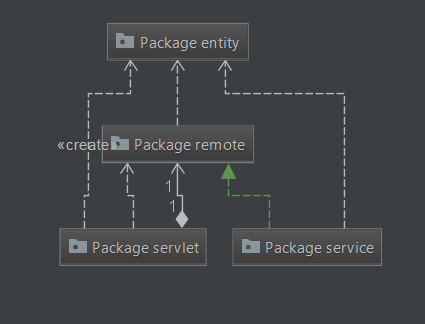
* Module diagram:



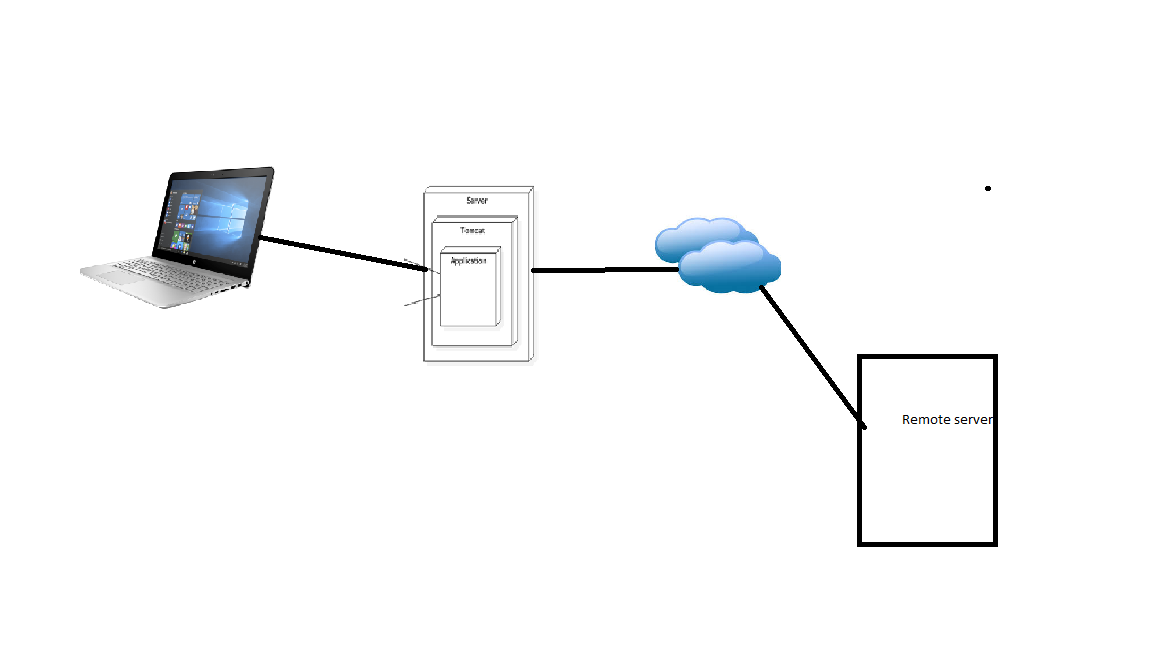
* Class diagram:



* Package diagram:



1. UML Deployment diagram



1. Readme file containing build and execution considerations

* Install instructions
* You must have java installed (<https://java.com/en/download/help/windows_manual_download.xml>)
* You must have tomcat installed

(<https://tomcat.apache.org/download-90.cgi>)

* Running and accessing the application
  + - Go to the application folder
    - Open a command window
    - Run the command: java -jar filename.jar
    - Open your browser and go to localhost:/8080 and you'll be redirected to the main page where you can calculate taxes