Web Services Term Project

(Option 2 – Pick this – DoctorPatient Example)

Kenneth Silvere

UIN: 662516116

Design, Resources & Approach

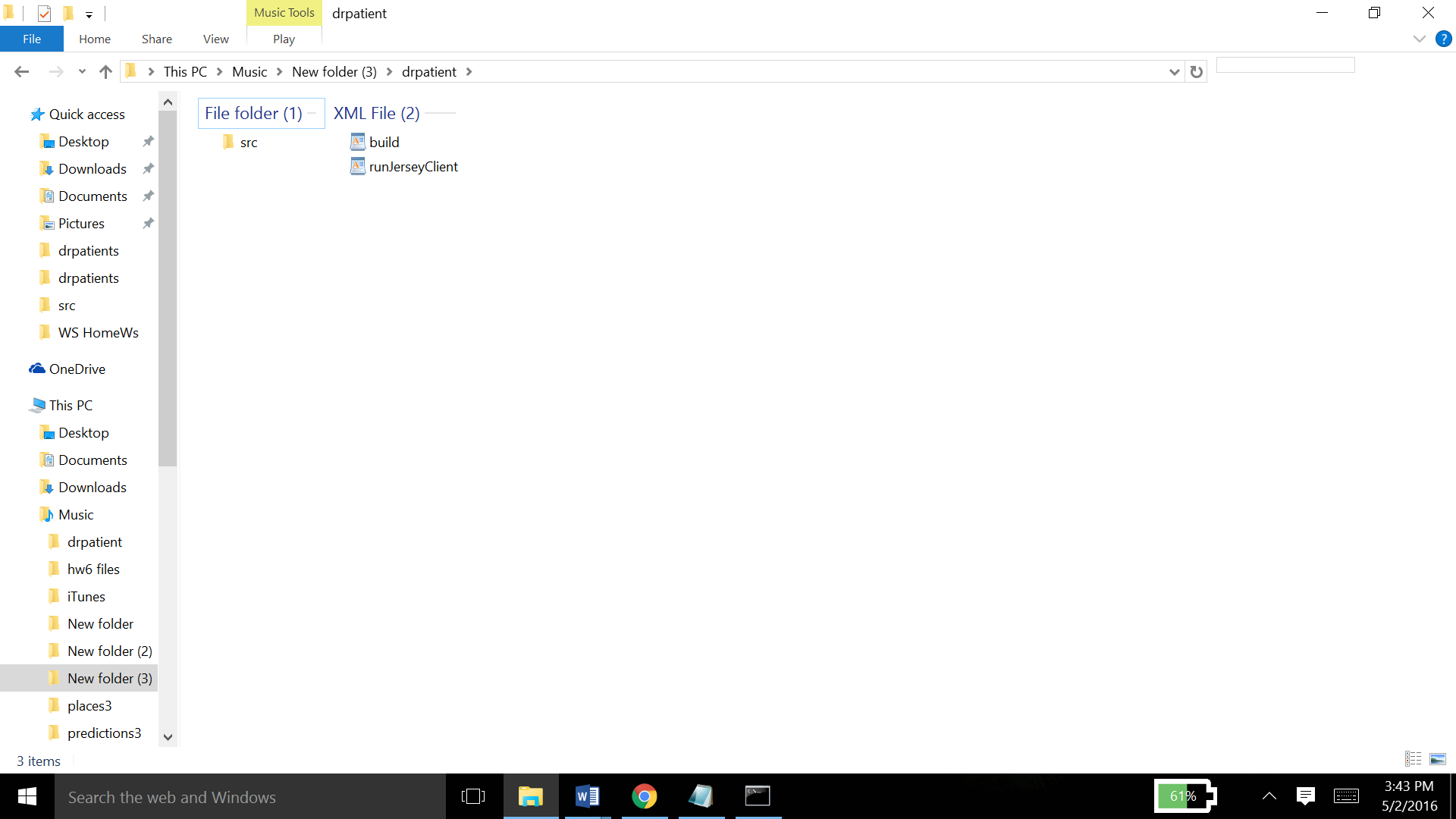
The main structure of this project was derived from the programs in the textbook.

My main resource was the textbook we use for this course, and to be more specific, the places3 assignment plays a major role.

Briefly specifying and describing the project files and resources:

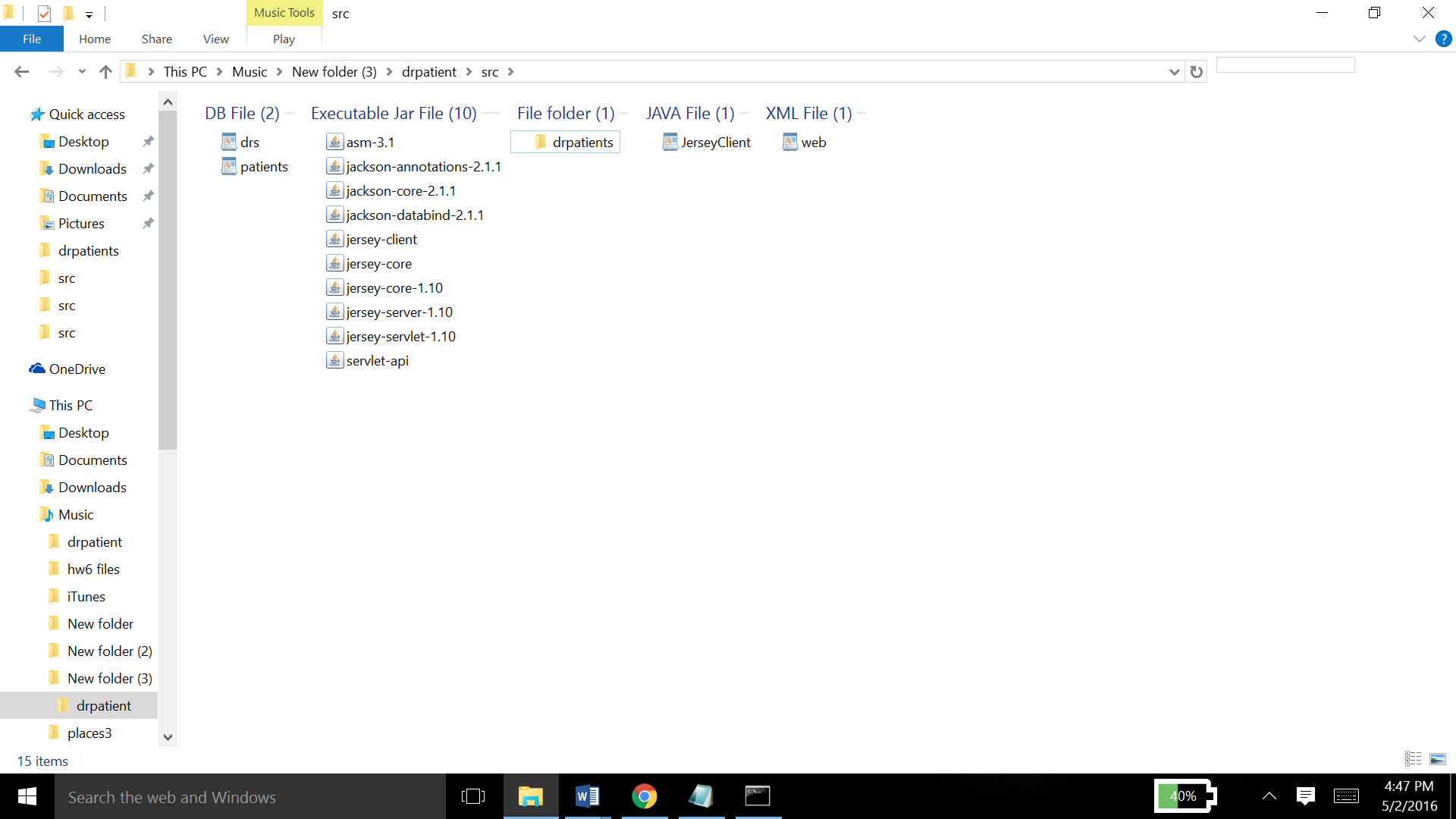
The main files in the project directory are

* src folder ( Conatins all the source programs and source code for the project )
* build.xml file ( builds up the project with the compiled source code)
* runJerseyClient.xml file (Jersey Client using JAX-RS API )

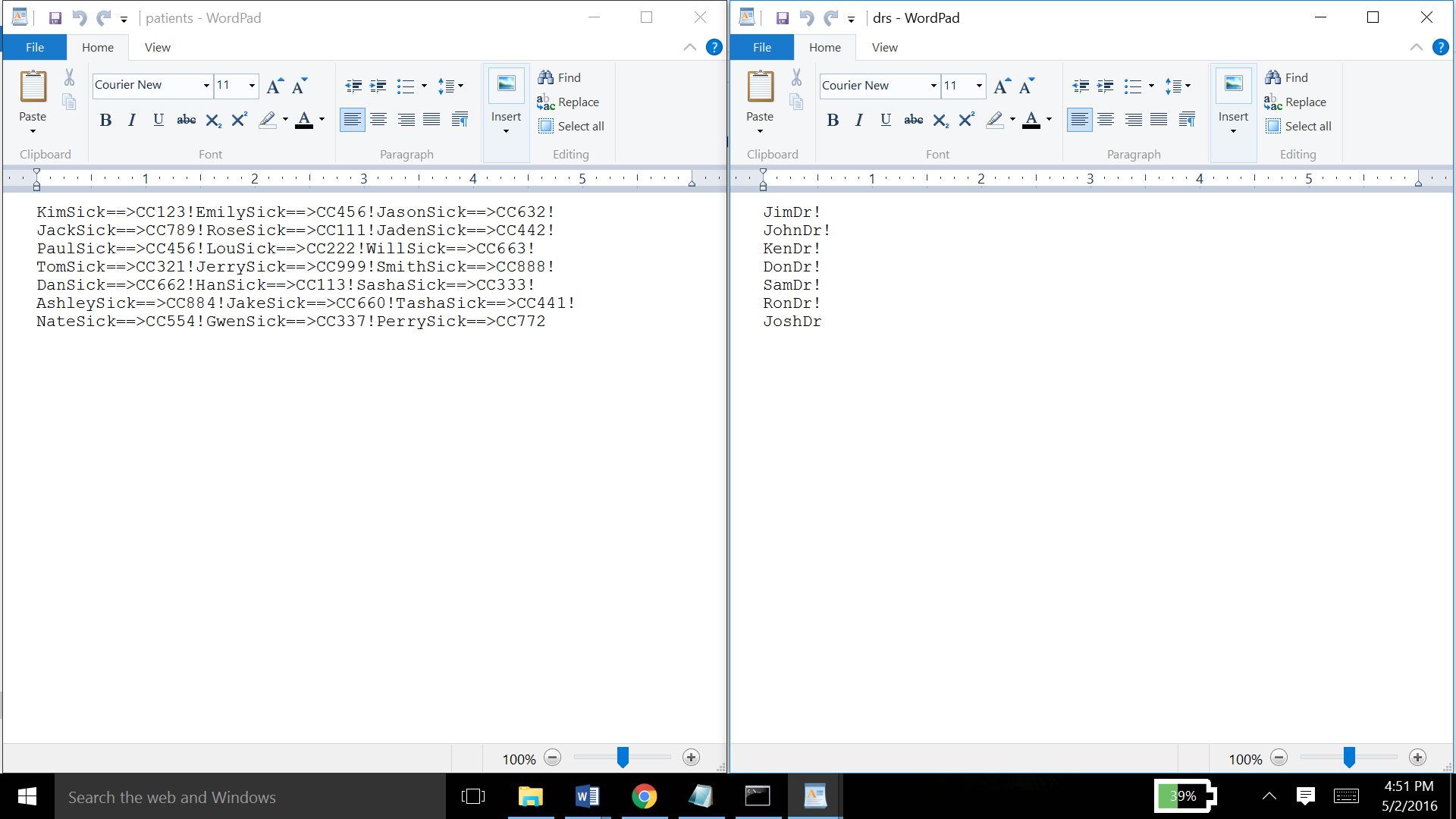


Next going into the src folder to see the core of project

Contents of the src folder:



Firstly there are two DB files, namely drs.db and patients.db, which hold the data of doctor names and patient names with insurance numbers, respectively.

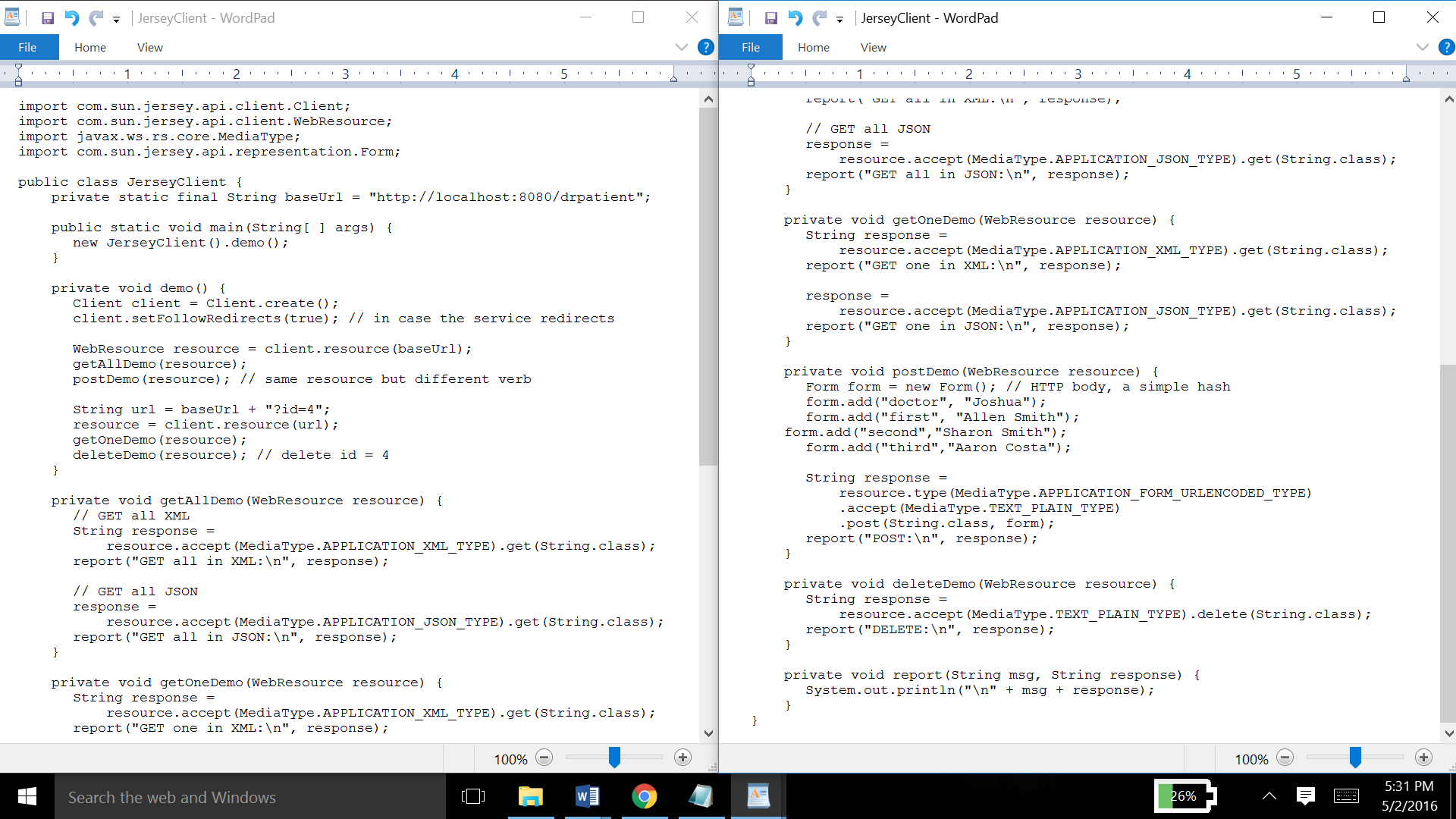


Each name is separated by a ‘!’ in both the DB files.

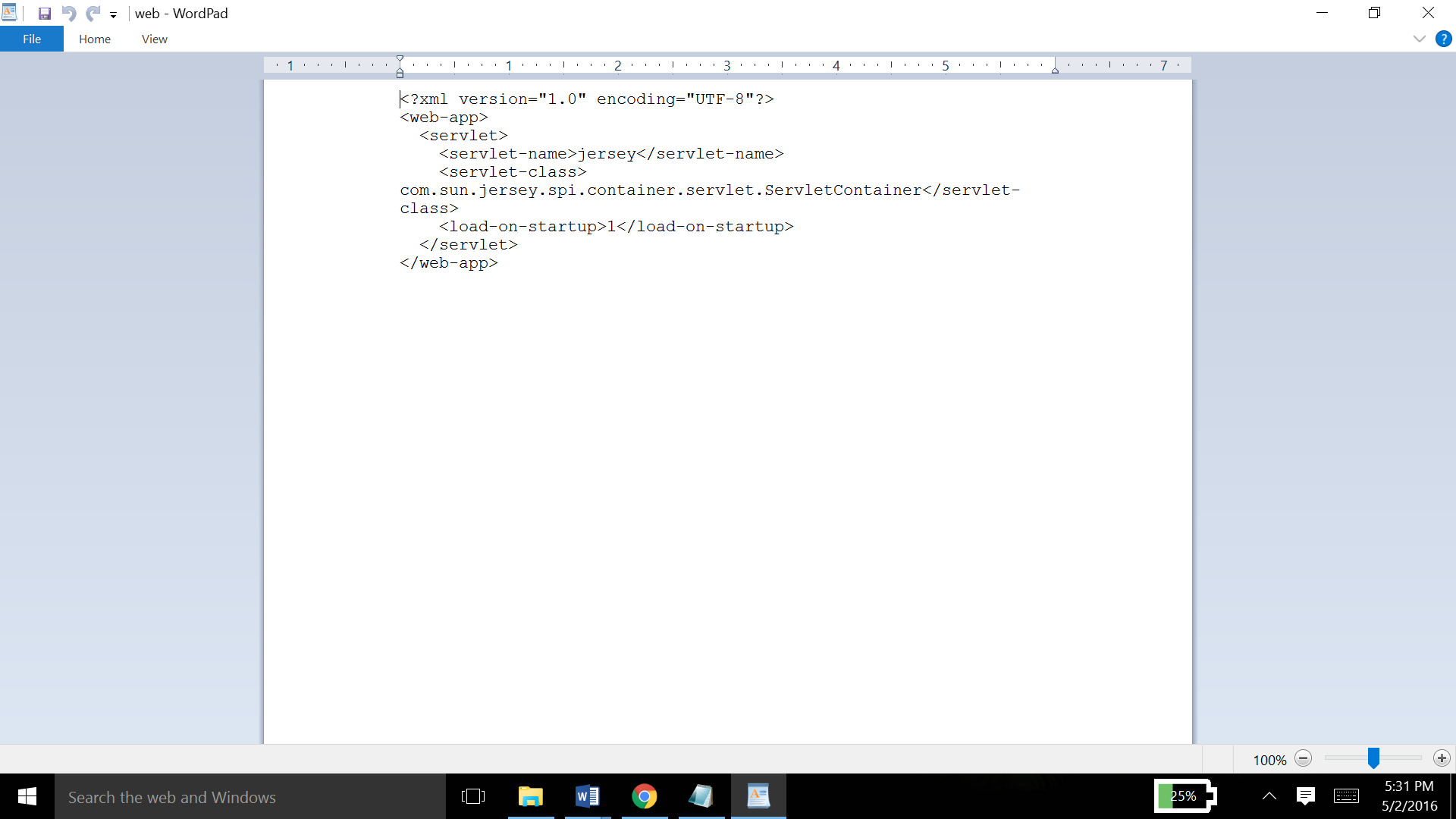
Next is a list of all the jar files used in and needed for the compilation and execution of this project for various tasks and purposes. Following is the list of all those jar files

* asm-3.1
* jackson-annotations-2.1.1
* jackson-core-2.1.1
* jackson-datablind-2.1.1
* jersey-client
* jersey-core
* jersey-core-1.10
* jersey-server-1.10
* jesey-servlet-1.10
* servlet-api

Next is the JerseyClient.java file which acts as the client to this service.



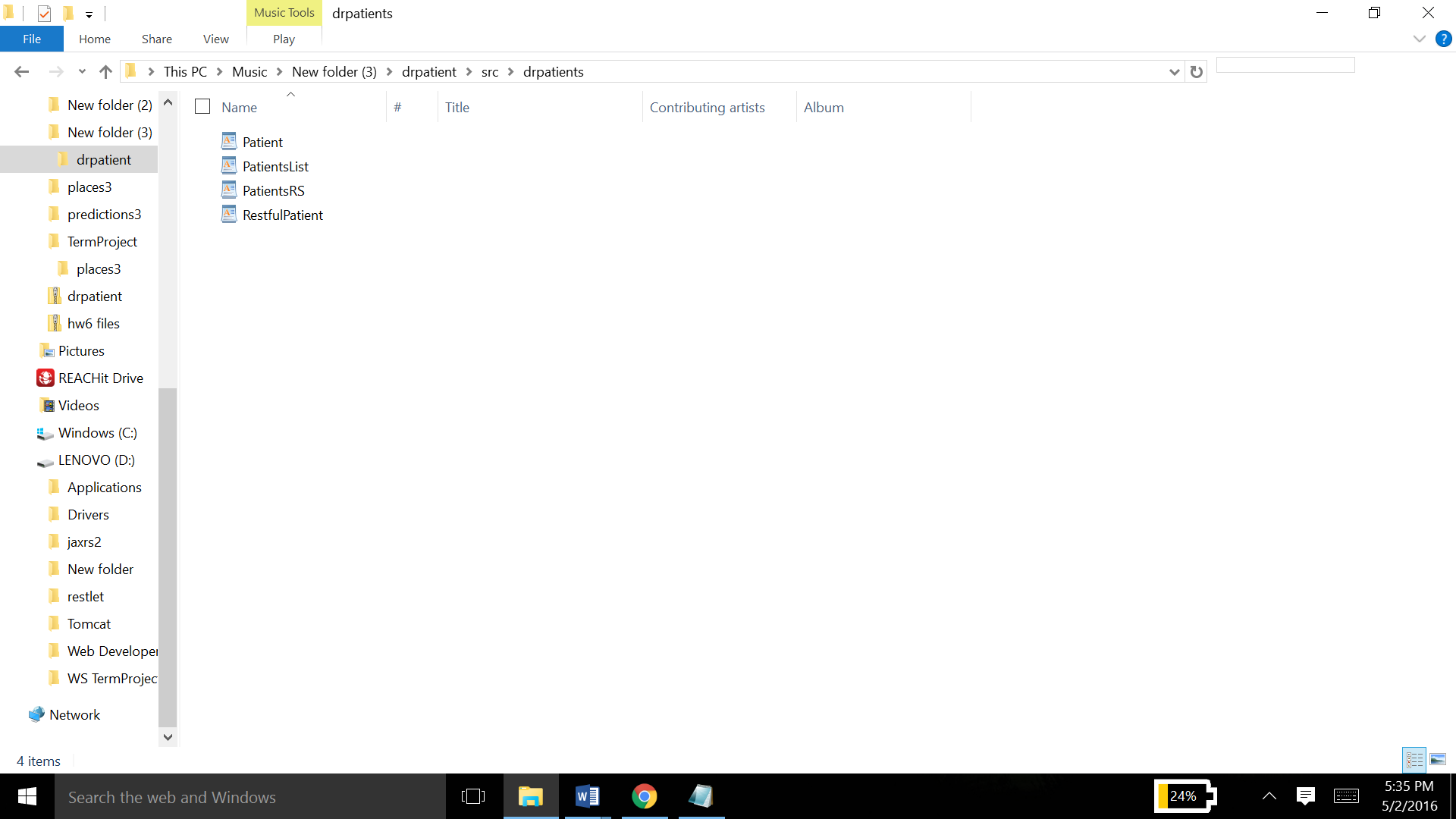
After that is the web.xml file which calls the servlet



Finally, there is the drpatients folder which contains all the core programs of the projects.

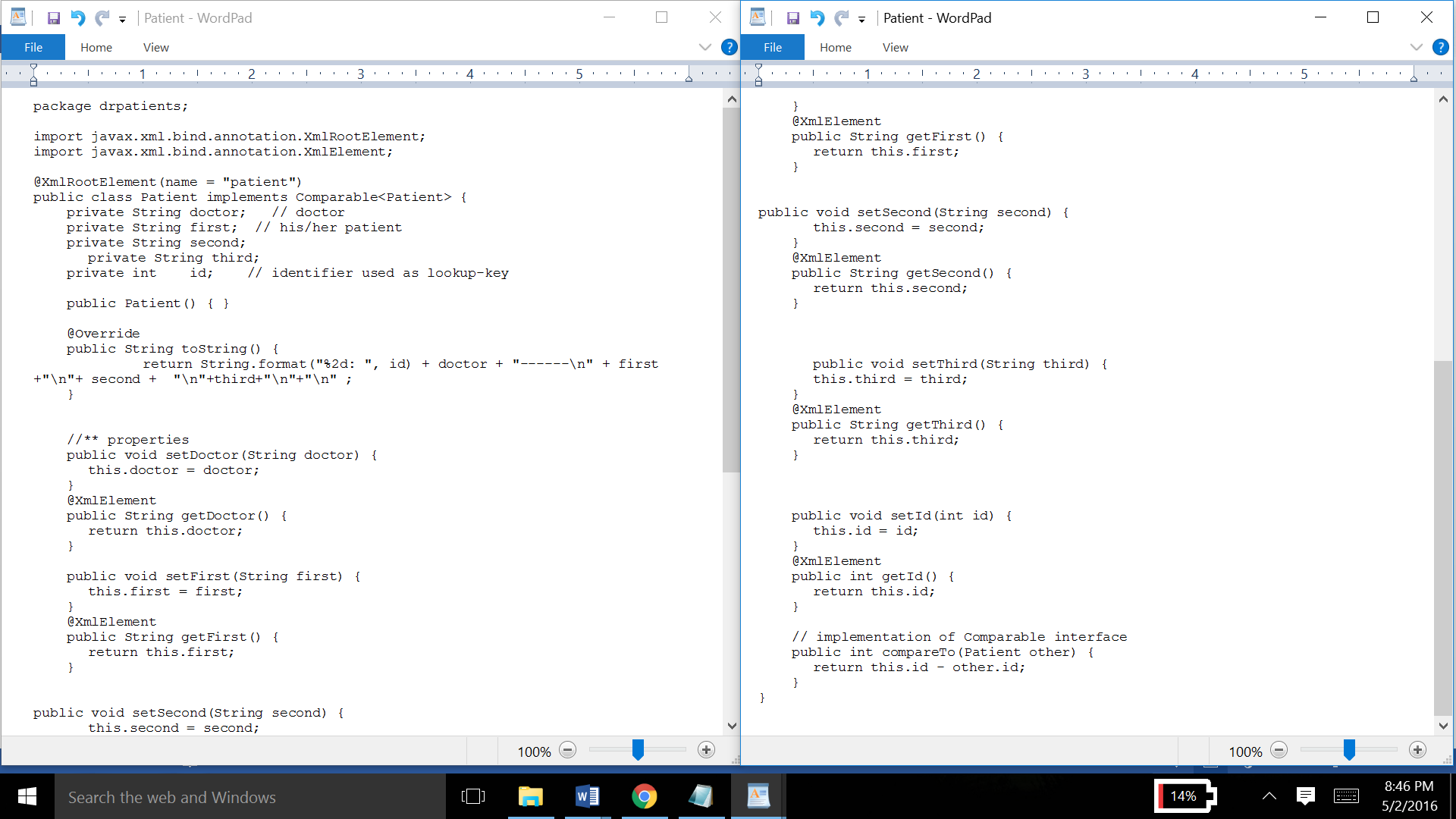
It consists of four programs namely

Patient.java, PatientList.java, PatientsRS.java, RestfulPatient.java

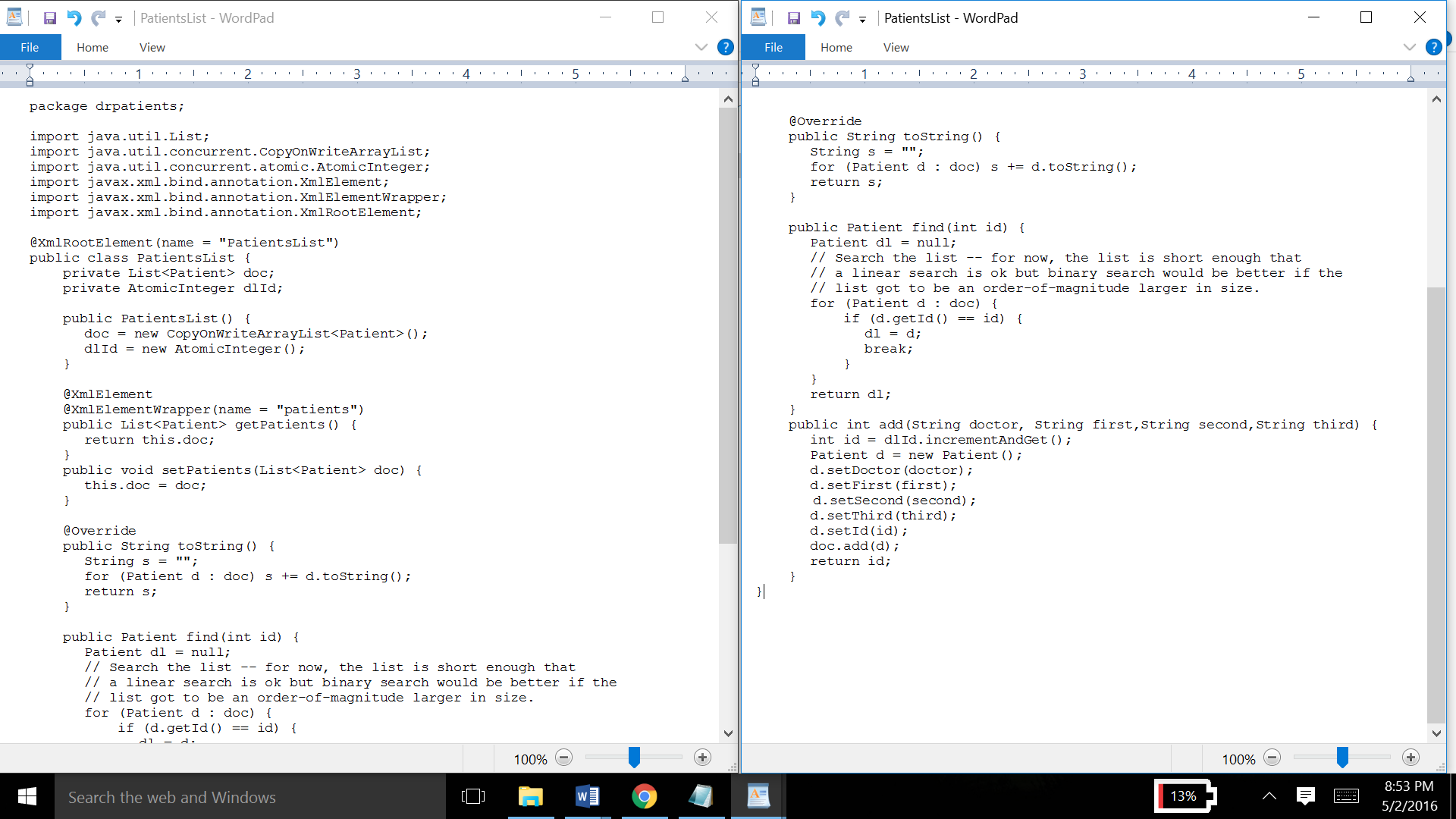


First, the Patient.java file. It holds the properties of the three patients and the doctor and his id.

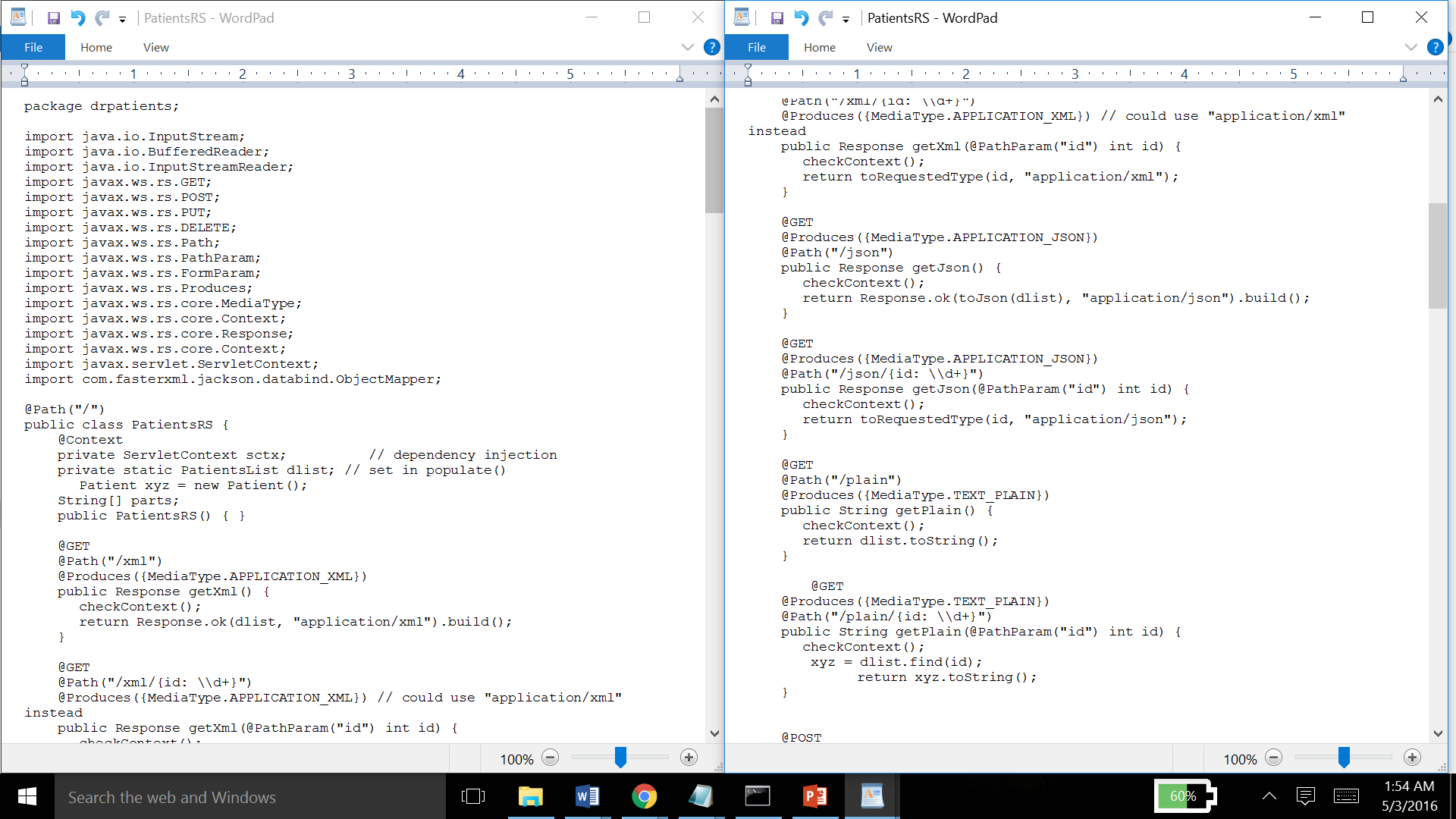
It also has methods to set and get these properties.



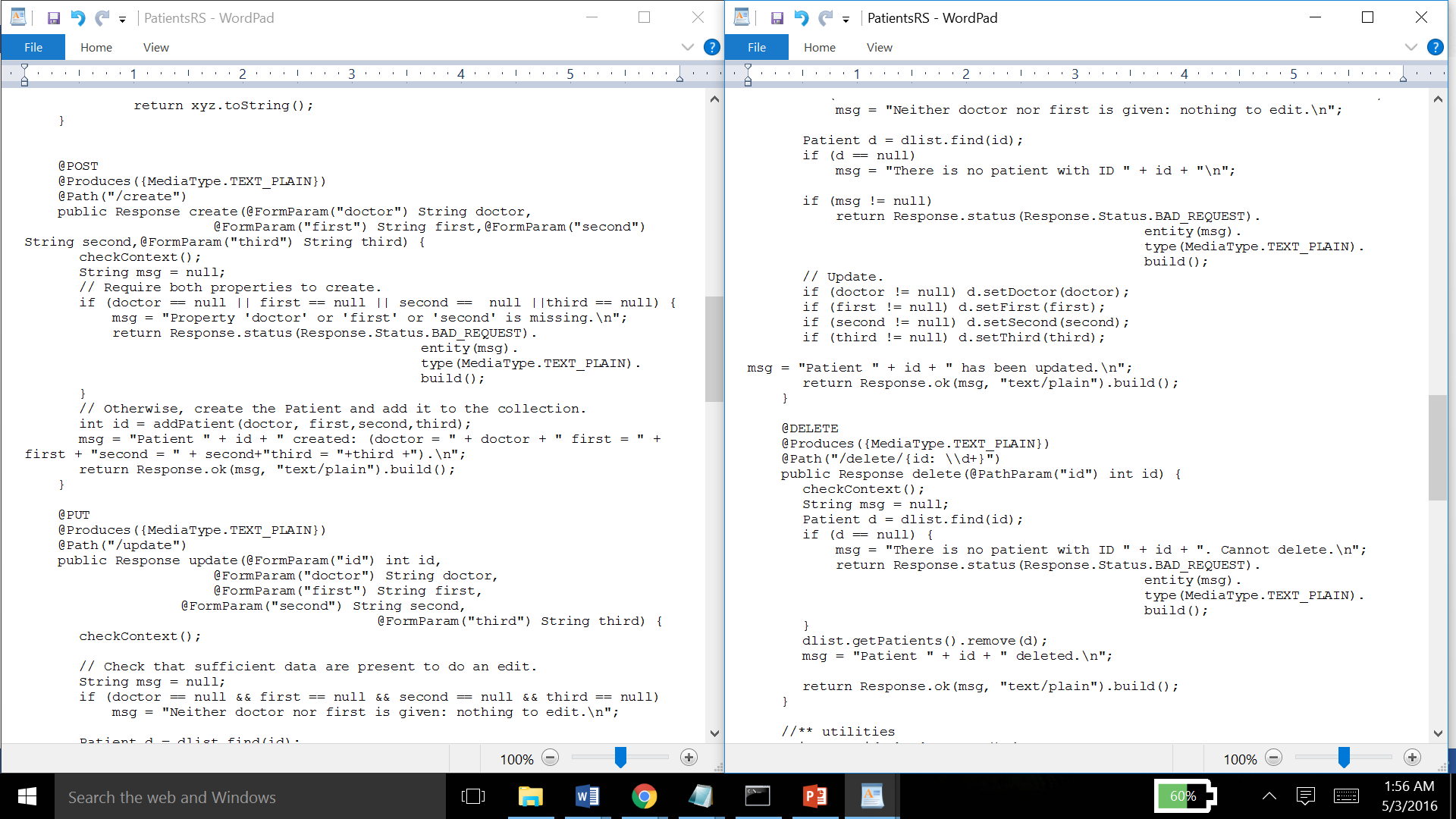
Next, the PatientsList.java file. This program creates a list of the doctor and his/her patients and uses the methods of Patient class to set it’s properties in the patient list.



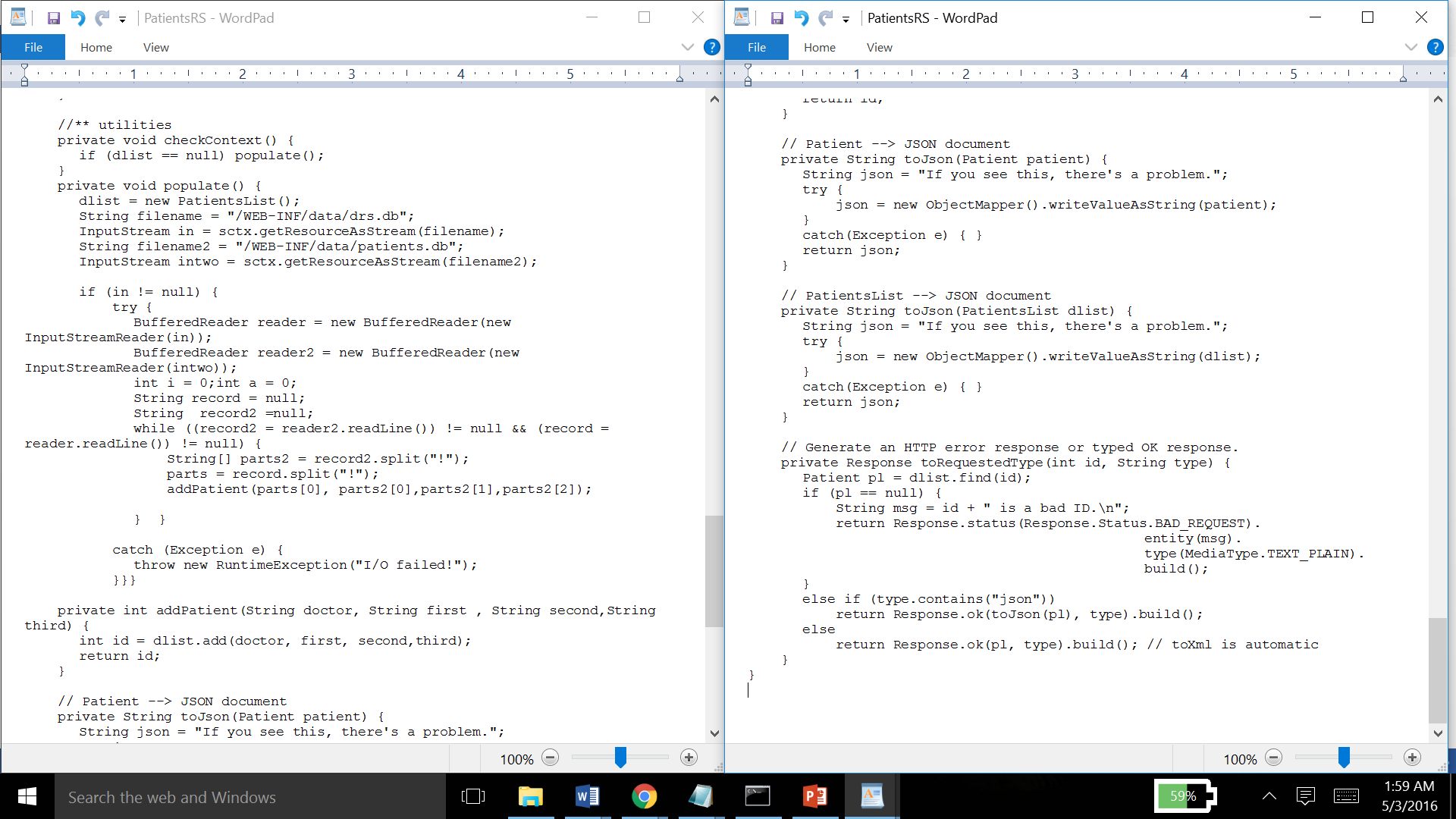
After that, is the PatientsRS.java program. It is a fairly large one. This program takes input from the given DB files, separates the words according to the specified separator, stores them in an array, and adds them to a PatientsList through it’s methods, hence creating multiple PatientsLists ,which will be outputted on the screen , giving each doctor three patients along with their insurance numbers.



(screenshot of the continued program)

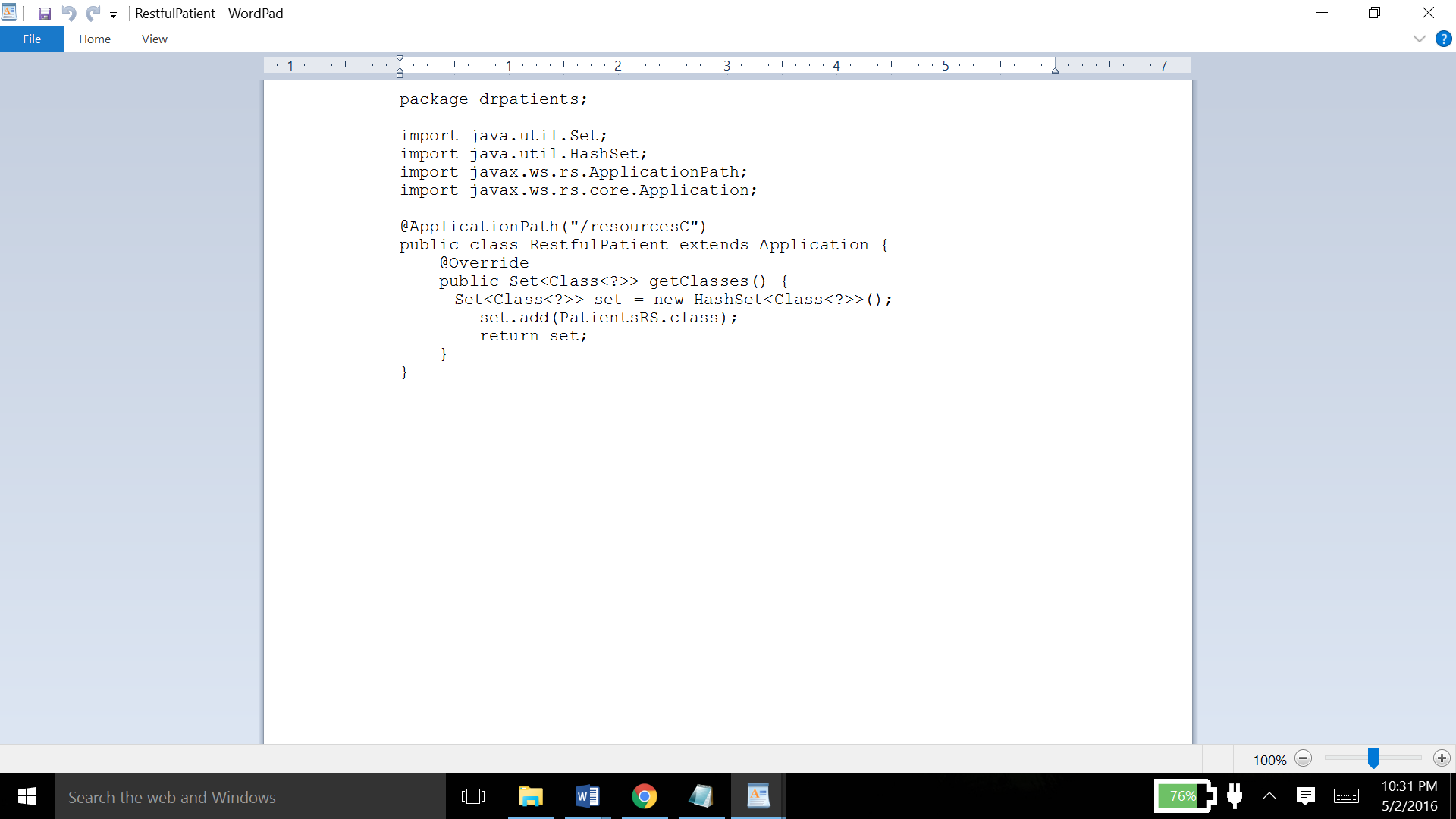


(screenshot of the continued program)

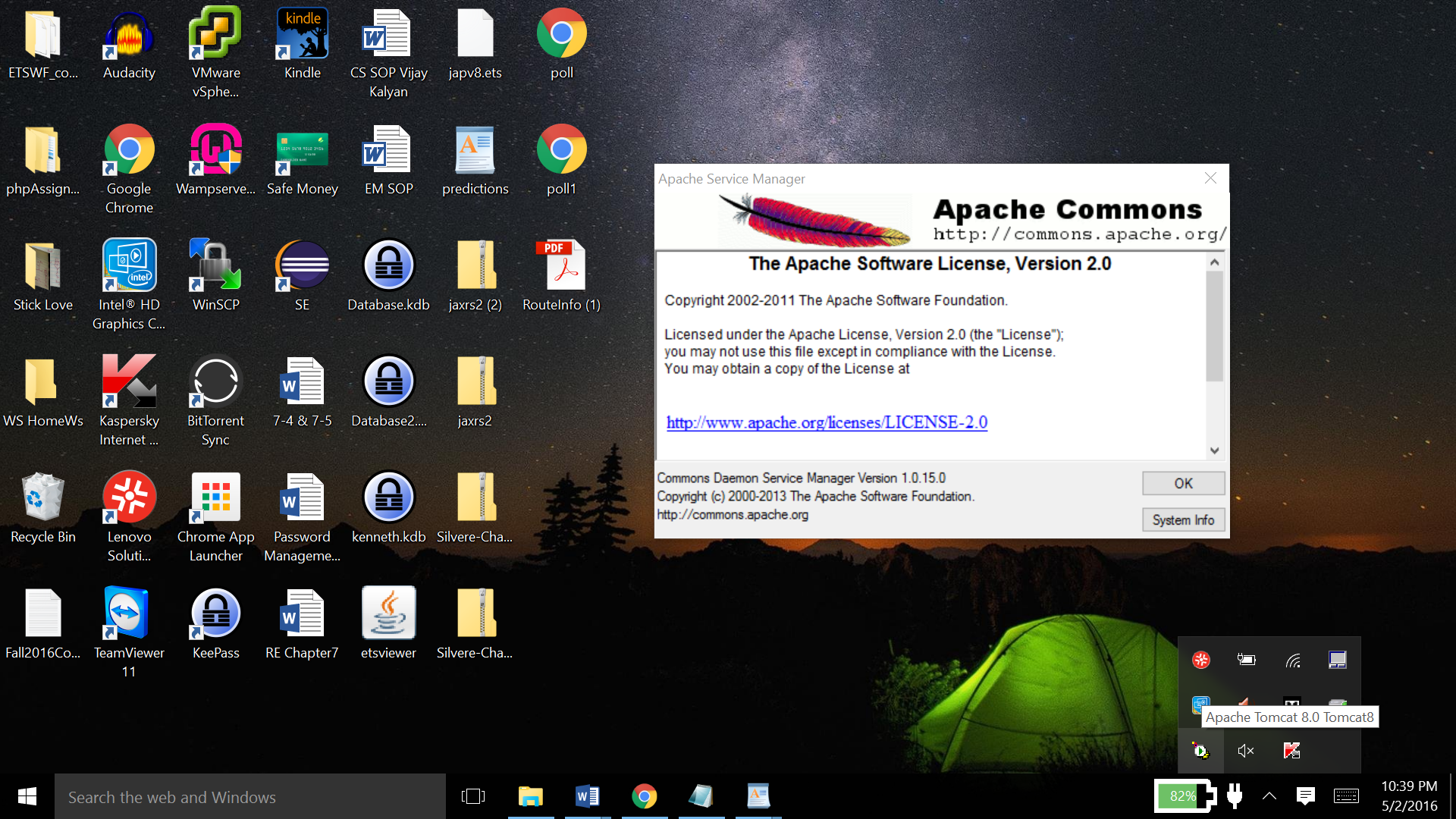


Finally, there is the RestfulPatient.java program

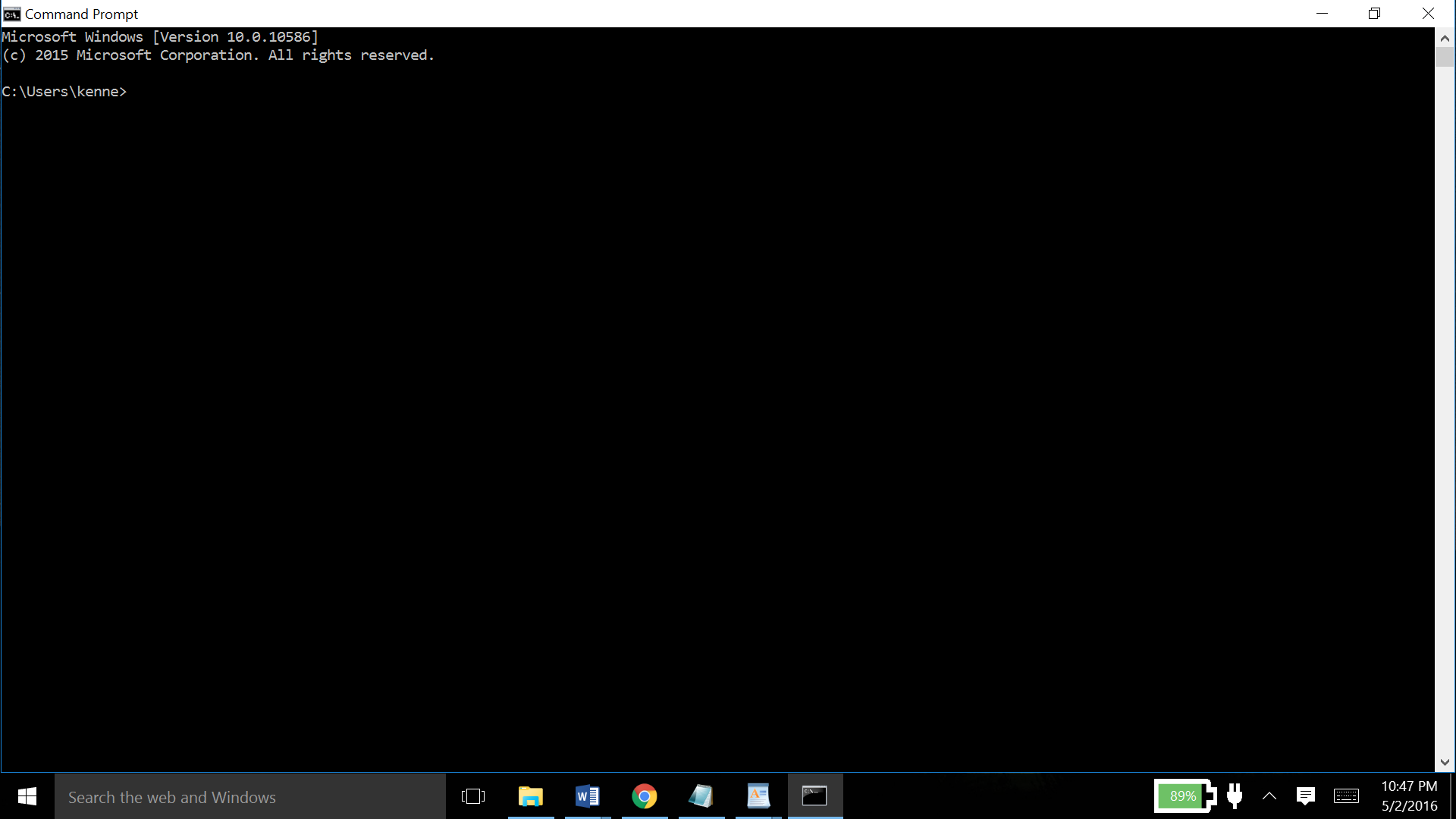
This forward all the lists of doctors and their patients (PatientRS.class) , as an HashSet object.



Operations

The JAX-RS API was the WS API used in this project. The Jersey client is the client used here, and the Apache Tomcat 8.0 server was used as the server to this client.

The command prompt was the interface used to perform these operations



The CURL utility was used for performing CRUD operations in this service.

**curl** is a **tool** to transfer data from or to a server, using one of the supported protocols  (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). The command is designed to work without user interaction.

Following are some of the CRUD commands that I have used in CURL :

curl --request DELETE localhost:8080/drpatients/resourcesC/delete/7

curl --request GET localhost:8080/drpatients/resourcesC/json/4

curl --request POST --data "doctor=Evelyn Silvere" --data "first=Anita" --data "second=Van Jake" --data "third=Derrick" localhost:8080/drpatients/resourcesC/create

curl --request PUT --data "id=8" --data "doctor=Dr.Silvere" localhost:8080/drpatients/resourcesC/update

All the programs were edited in a normal text editor.

A lot of testing was done in the process of making this project. Initially, there were many logic errors in getting data from two db files and passing them to a single function.

There were a few trouble shooting cases, mostly where proper input was not given to the programs and the project would not compile. I designed this service to have one doctor with three patients each. So, if there were unequal number of patients in the patients.db file, an input error would have been caused.

Another scenario was where Servlet Context was used many times, but there was no Servlet Context class present. An attempt was made to import that class but there was no resource with that class. So I had to include a servlet-api.jar in the src folder. I could have downloaded it from the internet, but I already had it in the lib subdirectory of my Tomcat Server directory.