1 Source Code

```
/* This program is free software: you can redistribute it and/or
 modify it under the terms of the GNU Lesser General Public License
 as published by the Free Software Foundation, either version 3 of
 the License, or (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program.  If not, see <http://www.gnu.org/licenses/>. */

package org.opentripplanner.standalone;

import com.beust.jcommander.JCommander;
import com.beust.jcommander.ParameterException;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.MissingNode;
import org.opentripplanner.common.MavenVersion;
import org.opentripplanner.graph_builder.GraphBuilder;
import org.opentripplanner.routing.graph.Graph;
import org.opentripplanner.routing.impl.DefaultStreetVertexIndexFactory;
import org.opentripplanner.routing.impl.GraphScanner;
import org.opentripplanner.routing.impl.InputStreamGraphSource;
import org.opentripplanner.routing.impl.MemoryGraphSource;
import org.opentripplanner.routing.services.GraphService;
import org.opentripplanner.scripting.impl.BSFOTPScript;
import org.opentripplanner.scripting.impl.OTPScript;
import org.opentripplanner.visualizer.GraphVisualizer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

/**
 * This is the main entry point to OpenTripPlanner. It allows both
   building graphs and starting up an OTP server
 * depending on command line options. OTPMain is a concrete class making
   it possible to construct one with custom
 * CommandLineParameters and use its graph builder construction method
   from web services or scripts, not just from the
 * static main function below.
 *
 * TODO still it seems fairly natural for all of these methods to be
   static.
```

```
47  */
public class OTPMain {

    private static final Logger LOG = LoggerFactory.getLogger(OTPMain.
    class);

    private final CommandLineParameters params;
53  public OTPServer otpServer = null;
    public GraphService graphService = null;

    /** ENTRY POINT: This is the main method that is called when running
    otp.jar from the command line. */
57  public static void main(String[] args) {

59      /* Parse and validate command line parameters. */
        CommandLineParameters params = new CommandLineParameters();
61      try {
            JCommander jc = new JCommander(params, args);
63          if (params.version) {
                System.out.println(MavenVersion.VERSION.
    getLongVersionString());
65              System.exit(0);
            }
67          if (params.help) {
                System.out.println(MavenVersion.VERSION.
    getShortVersionString());
69              jc.setProgramName("java -Xmx[several]G -jar otp.jar");
                jc.usage();
71              System.exit(0);
            }
73          params.infer();
        } catch (ParameterException pex) {
75          System.out.println(MavenVersion.VERSION.getShortVersionString
    ());
            LOG.error("Parameter error: {}", pex.getMessage());
77          System.exit(1);
        }
79
        if (params.build == null && !params.visualize && !params.server &&
     params.scriptFile == null) {
81          LOG.info("Nothing to do. Use --help to see available tasks.");
            System.exit(-1);
83      }

85      OTPMain main = new OTPMain(params);
        main.run();
87
    }
89
    /* Constructor. */
91  public OTPMain(CommandLineParameters params) {
        this.params = params;
```

```
93      }
95      /**
          * Making OTPMain a concrete class and placing this logic an instance
        method instead of embedding it in the static
97        * main method makes it possible to build graphs from web services or
        scripts, not just from the command line.
          */
99      public void run() {
101         // TODO do params.infer() here to ensure coherency?
103         /* Create the top-level objects that represent the OTP server. */
            makeGraphService();
105         otpServer = new OTPServer(params, graphService);
107         /* Start graph builder if requested */
            if (params.build != null) {
109             GraphBuilder graphBuilder = GraphBuilder.forDirectory(params,
        params.build); // TODO multiple directories
                if (graphBuilder != null) {
111                 graphBuilder.run();
                    /* If requested, hand off the graph to the server as the
        default graph using an in-memory GraphSource. */
113                 if (params.inMemory || params.preFlight) {
                        Graph graph = graphBuilder.getGraph();
115                     graph.index(new DefaultStreetVertexIndexFactory());
                        // FIXME set true router IDs
117                     graphService.registerGraph("", new MemoryGraphSource("
        ", graph));
                    }
119             } else {
                    LOG.error("An error occurred while building the graph.
        Exiting.");
121                 System.exit(-1);
                }
123         }
125         /* Scan for graphs to load from disk if requested */
            // FIXME eventually router IDs will be present even when just
        building a graph.
127         if ((params.routerIds != null && params.routerIds.size() > 0) ||
        params.autoScan) {
                /* Auto-register pre-existing graph on disk, with optional
        auto-scan. */
129             GraphScanner graphScanner = new GraphScanner(graphService,
        params.graphDirectory, params.autoScan);
                graphScanner.basePath = params.graphDirectory;
131             if (params.routerIds != null && params.routerIds.size() > 0) {
                    graphScanner.defaultRouterId = params.routerIds.get(0);
133             }
                graphScanner.autoRegister = params.routerIds;
```

```
135             graphScanner.startup();
        }
137
        /* Start visualizer if requested */
139     if (params.visualize) {
            Router defaultRouter = graphService.getRouter();
141         defaultRouter.graphVisualizer = new GraphVisualizer(
    defaultRouter);
            defaultRouter.graphVisualizer.run();
143         defaultRouter.timeouts = new double[] {60}; // avoid timeouts
    due to search animation
        }
145
        /* Start script if requested */
147     if (params.scriptFile != null) {
            try {
149             OTPScript otpScript = new BSFOTPScript(otpServer, params.
    scriptFile);
                if (otpScript != null) {
151                 Object retval = otpScript.run();
                    if (retval != null) {
153                     LOG.warn("Your script returned something, no idea
    what to do with it: {}", retval);
                    }
155             }
            } catch (Exception e) {
157             throw new RuntimeException(e);
            }
159     }

161     /* Start web server if requested */
        if (params.server) {
163         GrizzlyServer grizzlyServer = new GrizzlyServer(params,
    otpServer);
            while (true) { // Loop to restart server on uncaught fatal
    exceptions.
165             try {
                    grizzlyServer.run();
167                 return;
                } catch (Throwable throwable) {
169                 LOG.error("An uncaught {} occurred inside OTP.
    Restarting server.",
                            throwable.getClass().getSimpleName(),
    throwable);
171             }
            }
173     }

175 }

177 /**
```

```
     * Create a cached GraphService that will be used by all OTP
    components to resolve router IDs to Graphs.
179  * If a graph is supplied (graph parameter is not null) then that
    graph is also registered.
     * TODO move into OTPServer and/or GraphService itself, eliminate
    FileFactory and put basePath in GraphService
181  */
     public void makeGraphService () {
183      graphService = new GraphService(params.autoReload);
         InputStreamGraphSource.FileFactory graphSourceFactory =
185              new InputStreamGraphSource.FileFactory(params.
    graphDirectory);
         graphService.graphSourceFactory = graphSourceFactory;
187      if (params.graphDirectory != null) {
             graphSourceFactory.basePath = params.graphDirectory;
189      }
     }
191
     /**
193  * Open and parse the JSON file at the given path into a Jackson JSON
    tree. Comments and unquoted keys are allowed.
     * Returns null if the file does not exist,
195  * Returns null if the file contains syntax errors or cannot be parsed
     for some other reason.
     *
197  * We do not require any JSON config files to be present because that
    would get in the way of the simplest
     * rapid deployment workflow. Therefore we return an empty JSON node
    when the file is missing, causing us to fall
199  * back on all the default values as if there was a JSON file present
    with no fields defined.
     */
201  public static JsonNode loadJson (File file) {
         try (FileInputStream jsonStream = new FileInputStream(file)) {
203          ObjectMapper mapper = new ObjectMapper();
             mapper.configure(JsonParser.Feature.ALLOW_COMMENTS, true);
205          mapper.configure(JsonParser.Feature.ALLOW_UNQUOTED_FIELD_NAMES
    , true);
             JsonNode config = mapper.readTree(jsonStream);
207          LOG.info("Found and loaded JSON configuration file '{}'", file
    );
             return config;
209      } catch (FileNotFoundException ex) {
             LOG.info("File '{}' is not present. Using default
    configuration.", file);
211          return MissingNode.getInstance();
         } catch (Exception ex) {
213          LOG.error("Error while parsing JSON config file '{}': {}",
    file, ex.getMessage());
             System.exit(42); // probably "should" be done with an
    exception
215          return null;
```

```
            }
217     }

219

}
```

Listing 1: code1 zu sehen

```
for i:=maxint to 0 do
2 begin
j:=square(root(i));
4 end;
```

Listing 2: ein paar Zeilen code

```
hier kommt dann der Quellcode...
```

hier kommt der Quellcode...

2 Pseudo Code

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
  read current;
  if *understand* **then**
    go to next section;
    current section becomes this one;
  else
    go back to the beginning of current section;
  end
end

Algorithm 1: How to write algorithms