```java
package net.wiredclub.translation;

import ch.qos.logback.classic.Level;
import ch.qos.logback.classic.Logger;
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.DefaultParser;
import org.apache.commons.cli.HelpFormatter;
import org.apache.commons.cli.Option;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import static net.wiredclub.translation.TranslationStatusCode.STATUS_HELP;
import static net.wiredclub.translation.TranslationStatusCode.STATUS_INVALID_ARGUMENT;

/**
 * A helper tool for command line arguments.
 * For specified options either the default value or the given
 * command line argument will be used for TranslatorConfig.
 */
public class CommandLineHelper {

    public static final String DEFAULT_TRANSLATION_DIRECTORY = "translations";
    public static final String DEFAULT_SOURCE_LANGUAGE = "en";
    public static final String DEFAULT_REPOSITORY_PATH = ".";

    private final DeepLHelper deepLHelper;
    private final FileHelper fileHelper;

    public CommandLineHelper() {
        this.deepLHelper = new DeepLHelper(new JsonHelper());
        this.fileHelper = new FileHelper();
    }

    public CommandLineHelper(DeepLHelper deepLHelper, FileHelper fileHelper) {
        this.deepLHelper = deepLHelper;
        this.fileHelper = fileHelper;
    }

    public TranslationConfig getTranslationConfig(String[] args) throws TranslationException, IOException {
        Options options = defineOptions();

        try {
            return parseArguments(options, args);
        } catch (ParseException e) {
            displayHelp(options);
            throw new TranslationException("Error: Arguments could not be parsed. " + e.getMessage() + "\n",
                    STATUS_INVALID_ARGUMENT);
        }
    }

    /**
     * Options for command line.
     *
     * @return Options
     */
    Options defineOptions() {
        Options options = new Options();

        Option sourceOption = new Option("s", "source", true,
                "Source language (default is '" + DEFAULT_SOURCE_LANGUAGE + "')");
        options.addOption(sourceOption);

        Option targetOption = new Option("t", "target", true,
                "Target language (default all directories of path argument except source)");
        options.addOption(targetOption);

        Option pathOption = new Option("p", "path", true,
                "Path to translations directory from repo root (default is '" + DEFAULT_TRANSLATION_DIRECTORY + "')");
        options.addOption(pathOption);

        Option repoOption = new Option("r", "repo", true,
                "Path of repository (default is '" + DEFAULT_REPOSITORY_PATH + "')");
        options.addOption(repoOption);

        Option verbose = new Option("v", "verbose", false, "Turn on more output (default is off)");
        options.addOption(verbose);

        Option help = new Option("h", "help", false, "This help");
        options.addOption(help);

        return options;
    }

    /**
     * Parse command line arguments. If mandatory arguments are missing ParseException is thrown.
     *
     * @param options command line options which are allowed
     * @param args    command line arguments given by the user
     * @return the configuration of translation tool
     * @throws ParseException       if an argument is invalid, ParseException will be thrown
     * @throws TranslationException if json cannot be parsed wraps JsonProcessingException
     * @throws IOException          if request to deepl.com cannot be completed
     */
    TranslationConfig parseArguments(Options options, String[] args)
            throws ParseException, TranslationException, IOException {
        CommandLineParser parser = new DefaultParser();
        CommandLine cmd = parser.parse(options, args);

        if (cmd.hasOption("help")) {
            displayHelp(options);
            throw new TranslationException(STATUS_HELP);
        }

        Logger root = (Logger) LoggerFactory.getILoggerFactory().getLogger("ROOT");
        root.setLevel(Level.toLevel(cmd.hasOption("verbose") ? "ALL" : "INFO"));

        String repositoryDirectory = cmd.getOptionValue("repo", DEFAULT_REPOSITORY_PATH).trim();
        String translationsDirectory = cmd.getOptionValue("path", DEFAULT_TRANSLATION_DIRECTORY).trim();

        String sourceLanguage = cmd.getOptionValue("source", DEFAULT_SOURCE_LANGUAGE).trim();
        List<String> sourceLanguages = deepLHelper.sourceLanguages();
        if (!sourceLanguages.contains(sourceLanguage)) {
            throw new ParseException("Source language '" + sourceLanguage + "' is not allowed. "
                    + "Possible values are: " + sourceLanguages);
        }

        String targetLanguage = cmd.getOptionValue("target");
        Set<String> targetLanguages;
        if (targetLanguage == null) {
            // find in parent directory all other directories which are not the source directory
            targetLanguages = fileHelper.discoverLanguageDirectories(repositoryDirectory + "/" + translationsDirectory);
            targetLanguages.remove(sourceLanguage);
        } else {
            targetLanguages = Set.of(targetLanguage.trim());
        }

        List<String> possibleLanguages = deepLHelper.targetLanguages();
        if (!new HashSet<>(possibleLanguages).containsAll(targetLanguages)) {
            throw new ParseException("Some target languages " + targetLanguages + " are not allowed. "
                    + "Possible target languages are: " + possibleLanguages);
        }

        return new TranslationConfig(sourceLanguage, targetLanguages, translationsDirectory, repositoryDirectory);
    }

    /**
     * Display an explanation of the translation tool.
     *
     * @param options command line options which are allowed
     */
    void displayHelp(Options options) {
        String header =
                "\nTranslationTool reads all keys from source language, finds differences to each target file and "
                        + "differences to previous commit of source. Changed translations are then retrieved from "
                        + "https://www.deepl.com/translator and stored for each target language(s). The order of "
                        + "existing keys is preserved, but new keys are added at the end of the target file.\n\n";

        String footer = "\nExample: translation-tool -s en -t de -p \"translations\" -r .";

        HelpFormatter formatter = new HelpFormatter();
        formatter.setOptionComparator(null);
        formatter.printHelp("translation-tool", header, options, footer, true);
    }
}

```