```java
package net.wiredclub.translation;

import com.fasterxml.jackson.databind.JsonNode;
import org.apache.http.client.fluent.Form;
import org.apache.http.client.fluent.Request;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.Set;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

/**
 * A helper class that executes requests to DeepL-API, e.g.
 * it retrieves allowed source and target languages, usage stats,
 * or triggers the translation of a text.
 */
public class DeepLHelper {

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

    private static final String AUTH_KEY = "bddf179b-b8b6-d1a3-2a96-11c7cc8ac50a:fx";

    private static final String DEEPL_BASE_URI_FREE = "https://api-free.deepl.com";
    private static final String DEEPL_USAGE = "/v2/usage";
    private static final String DEEPL_LANGUAGES = "/v2/languages";
    private static final String DEEPL_TRANSLATE = "/v2/translate";

    private static final String XML_TAG_TO_EXCHANGE_CURLY_BRACKETS = "donut";

    private final JsonHelper jsonHelper;

    public DeepLHelper() {
        this.jsonHelper = new JsonHelper();
    }

    public DeepLHelper(JsonHelper jsonHelper) {
        this.jsonHelper = jsonHelper;
    }

    public DeepLUsage usage() throws IOException, TranslationJsonProcessingException {
        String response = Request.Post(DEEPL_BASE_URI_FREE + DEEPL_USAGE)
                .bodyForm(Form.form()
                        .add("auth_key", AUTH_KEY)
                        .build())
                .execute().returnContent().asString();

        JsonNode json = jsonHelper.convertStringToJson(response);

        long characterCount = json.get("character_count").asLong();
        long characterLimit = json.get("character_limit").asLong();

        return new DeepLUsage(characterCount, characterLimit);
    }

    public List<String> sourceLanguages() throws IOException, TranslationJsonProcessingException {
        String response = Request.Post(DEEPL_BASE_URI_FREE + DEEPL_LANGUAGES)
                .bodyForm(Form.form()
                        .add("auth_key", AUTH_KEY)
                        .add("type", "source")
                        .build())
                .execute().returnContent().asString();

        JsonNode json = jsonHelper.convertStringToJson(response);
        Set<String> languages = jsonHelper.extractLanguages(json);
        return languages.stream().map(String::toLowerCase).sorted().collect(Collectors.toList());
    }

    public List<String> targetLanguages() throws IOException, TranslationJsonProcessingException {
        String response = Request.Post(DEEPL_BASE_URI_FREE + DEEPL_LANGUAGES)
                .bodyForm(Form.form()
                        .add("auth_key", AUTH_KEY)
                        .add("type", "target")
                        .build())
                .execute().returnContent().asString();

        JsonNode json = jsonHelper.convertStringToJson(response);
        Set<String> languages = jsonHelper.extractLanguages(json);
        return languages.stream().map(String::toLowerCase).sorted().collect(Collectors.toList());
    }

    public String translate(String textToTranslate, String sourceLanguage, String targetLanguage)
            throws IOException, TranslationJsonProcessingException {
        String response = Request.Post(DEEPL_BASE_URI_FREE + DEEPL_TRANSLATE)
                .bodyForm(Form.form()
                        .add("auth_key", AUTH_KEY)
                        .add("text", wrapTextToTranslate(textToTranslate))
                        .add("source_lang", sourceLanguage)
                        .add("target_lang", targetLanguage)
                        .add("tag_handling", "xml")
                        .add("ignore_tags", XML_TAG_TO_EXCHANGE_CURLY_BRACKETS) // xml tag for disabling translation
                        .build())
                .execute().returnContent().asString(StandardCharsets.UTF_8);

        JsonNode json = jsonHelper.convertStringToJson(response);
        // LOG.debug("Translated '{}' to '{}'", textToTranslate, translation);
        String translation = jsonHelper.extractTranslation(json, textToTranslate);
        return unwrapTranslation(translation);
    }

    private static final Pattern CURLY_BRACKETS_START = Pattern.compile("\\{\\{");
    private static final Pattern CURLY_BRACKETS_END = Pattern.compile("}}");

    String wrapTextToTranslate(String text) {
        String wrap = CURLY_BRACKETS_START.matcher(text).replaceAll("<" + XML_TAG_TO_EXCHANGE_CURLY_BRACKETS + ">{{");
        return CURLY_BRACKETS_END.matcher(wrap).replaceAll("}}</" + XML_TAG_TO_EXCHANGE_CURLY_BRACKETS + ">");
    }

    private static final Pattern DONUT_START = Pattern.compile("<" + XML_TAG_TO_EXCHANGE_CURLY_BRACKETS + ">");
    private static final Pattern DONUT_END = Pattern.compile("</" + XML_TAG_TO_EXCHANGE_CURLY_BRACKETS + ">");

    String unwrapTranslation(String translation) {
        String unwrap = DONUT_START.matcher(translation).replaceAll("");
        return DONUT_END.matcher(unwrap).replaceAll("");
    }

    record DeepLUsage(long characterCount, long characterLimit) {
    }
}

```