3. For RESTCONF, use Java's HttpClient or a third-party library like Apache HttpComponents to make REST API calls

The program includes methods for performing `GET`, `POST`, and `DELETE` requests on a RESTCONF API.

```
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.net.http.HttpHeaders;
import java.util.Base64;

public class RESTCONFManager {

    private static final String BASE_URL = "http://192.168.1.1:8080/restconf";  // Adjust base URL
and port as needed
    private static final String USERNAME = "username";
    private static final String PASSWORD = "password";

    private final HttpClient client;

    public RESTCONFManager() {
        this.client = HttpClient.newHttpClient();
    }

    // Perform GET request to retrieve data
    public String getConfig(String resource) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(BASE_URL + resource))
                .header("Authorization", "Basic " + encodeCredentials())
                .header("Accept", "application/yang-data+json")
                .GET()
                .build();

        HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
        return response.body();
    }

    // Perform POST request to edit or add configuration
    public void editConfig(String resource, String jsonBody) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(BASE_URL + resource))
                .header("Authorization", "Basic " + encodeCredentials())
                .header("Content-Type", "application/yang-data+json")
                .POST(HttpRequest.BodyPublishers.ofString(jsonBody))
                .build();

        HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
        System.out.println("Response: " + response.statusCode() + " - " + response.body());
    }

    // Perform DELETE request to remove configuration
    public void deleteConfig(String resource) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(BASE_URL + resource))
                .header("Authorization", "Basic " + encodeCredentials())
```

```
                .header("Accept", "application/yang-data+json")
                .DELETE()
                .build();

        HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
        System.out.println("Response: " + response.statusCode() + " - " + response.body());
    }

    // Helper method for encoding credentials in Basic Auth format
    private String encodeCredentials() {
        String credentials = USERNAME + ":" + PASSWORD;
        return Base64.getEncoder().encodeToString(credentials.getBytes());
    }

    public static void main(String[] args) {
        try {
            RESTCONFManager manager = new RESTCONFManager();

            // Get configuration (replace with desired RESTCONF resource path)
            String getResource = "/data/ietf-interfaces:interfaces";
            String getConfigResponse = manager.getConfig(getResource);
            System.out.println("GET Response: " + getConfigResponse);

            // Edit configuration (replace JSON body with your configuration details)
            String postResource = "/data/ietf-interfaces:interfaces/interface=eth0";
            String jsonBody = """
                {
                  "ietf-interfaces:interface": {
                    "name": "eth0",
                    "description": "Configured via RESTCONF",
                    "type": "iana-if-type:ethernetCsmacd",
                    "enabled": true
                  }
                }
            """;
            manager.editConfig(postResource, jsonBody);

            // Delete configuration (adjust to target specific resource)
            String deleteResource = "/data/ietf-interfaces:interfaces/interface=eth0";
            manager.deleteConfig(deleteResource);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```