HTTP Methods and Requests

In Java, you can send HTTP requests to interact with web services or APIs using various HTTP methods. HTTP methods are actions you can perform on a resource, and they include GET, POST, PUT, DELETE, and more. Here's a detailed explanation of some commonly used HTTP methods and examples in Java.

HTTP Methods:

1. **GET**: Used to retrieve data from a specified resource. It should not have any side effects on the server.

2. **POST**: Used to submit data to be processed to a specified resource. It can have side effects on the server.

3. **PUT**: Used to update a resource or create a new resource if it does not exist at the specified URI.

4. **DELETE**: Used to request the removal of a resource.

Now, let's provide code examples for each of these methods:

HTTP GET Request in Java:

To send an HTTP GET request in Java, you can use the `HttpURLConnection` class. Here's an example:

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class HttpGetExample {
    public static void main(String[] args) {
        try {
            URL url = new URL("https://api.example.com/resource");
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");

            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuilder response = new StringBuilder();
```

```
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            System.out.println("Response: " + response.toString());
            connection.disconnect();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

This code sends an HTTP GET request to "https://api.example.com/resource" and retrieves the response.

HTTP POST Request in Java:

To send an HTTP POST request in Java, you can use the `HttpURLConnection` class as well. Here's an example:

```java
import java.io.DataOutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class HttpPostExample {
    public static void main(String[] args) {
        try {
            URL url = new URL("https://api.example.com/resource");
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("POST");
            connection.setDoOutput(true);

            String requestBody = "key1=value1&key2=value2";

            try (DataOutputStream out = new DataOutputStream(connection.getOutputStream())) {
                out.writeBytes(requestBody);
            }

            int responseCode = connection.getResponseCode();
            System.out.println("Response Code: " + responseCode);
```

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

This code sends an HTTP POST request to "https://api.example.com/resource" with a request body containing key-value pairs.

These are basic examples, and in practice, you may want to handle exceptions and error responses more gracefully. Additionally, consider using libraries like Apache HttpClient or OkHttp for more complex HTTP interactions in Java.