Java: Try Me First

Find a ready-to-run code sample that is a good starting point for accessing the PostalMethods API via Java in the *pmjava/firstLetter* folder.

First thing you'll want to do is edit the file *pmjava/postalmethods.properties* and update the username and password values to match your account username and password.

The Java code we will be running is *FirstLetter.java*. This code sample will send a sample PDF letter using the PostalMethods API. To compile and run it do the following:

On Windows:

```
% set PM_HOME=<path to pmjava directory>
% run.bat
```

On Linux/Unix:

```
% export PM_HOME=<path to pmjava directory>
% ./run.sh
```

This command will compile and run the FirstLetter.java program and the output will look something like this:

```
PM_HOME is <path to pmjava directory>
Building FirstLetter.java ...
FirstLetter.java has been built.
Running the FirstLetter program ...
File to send: <path to pmjava directory>/misc/sample-letter.pdf
Making call to SendLetter...
Service call completed:
ResultCode:
       code: 1022100
    message: Letter has been successfully received.
description: Letter has been received and is being processed. The numeric code
returned by this call is your letter id.
FirstLetter program has run.
```

Congratulations! You've sent your first letter using the PostalMethods API.

Java: Send A Letter With SendLetter()

Find this sample in directory */test/com/postalmethods/client*

This sample demonstrates how to send a letter through PostalMethods using the Address Inside method.

```
package com.postalmethods.client;

import static org.junit.Assert.assertTrue;

import org.junit.Before;
```

```
import org.junit.Test;

import com.postalmethods.client.api.SendLetterResponse;

/**
 * Integration test that calls the SendLetter web method.
 * <p>
 * For details about PostalMethods.SendLetter see
 * <a href="http://www.postalmethods.com/method/2009-02-26/SendLetter">the
PostalMethods.SendLetter reference</a>
 */
public class SendLetterTest {

    private PostalMethodsClient client;

    @Before
    public void setUp() {
        client = new PostalMethodsClientFactory().build();
    }

    @Test
    public void testSendLetter() {
        System.out.println("Making call to SendLetter...");
        SendLetterResponse response = client.sendLetter("Sending test letter via JUnit
integration test.", TestConstants.SAMPLE_LETTER_FILENAME);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertTrue(code > 0);

        System.out.println("New letter id is: " + code);
    }
}
```

Java: Send A Letter with SendLetterAndAddress()

Find this sample in directory */test/com/postalmethods/client*

This sample demonstrates how to send a letter through PostalMethods using the Address Outside method.

```
package com.postalmethods.client;

import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.SendLetterAndAddressResponse;
import com.postalmethods.client.model.Address;
import com.postalmethods.client.model.LetterContent;
import com.postalmethods.client.model.LocalLetterContent;

/**
```

```
 * Integration test that calls the SendLetterAndAddress web method.
 * <p>
 * For details about PostalMethods.SendLetterAndAddress see
 * <a href="http://www.postalmethods.com/method/2009-02-26/SendLetterAndAddress">the
PostalMethods.SendLetterAndAddress reference</a>
 */
public class SendLetterAndAddressWithLocalLetterContent {

    private PostalMethodsClient client;

    @Before
    public void setUp() {
        client = new PostalMethodsClientFactory().build();
    }

    @Test
    public void testSendLetterAndAddress() {

        Address address = new Address("George Washington", "", "The White House",
"1600 Pennsylvania Ave", "", "Washington", "DC", "20500", "USA");
        String description = "Test letter with address.";

        LetterContent letterContent = new LocalLetterContent(
TestConstants.SAMPLE_LETTER_LOCAL_FILENAME );

        System.out.println("Making call to SendLetterAndAddress...");
        SendLetterAndAddressResponse response =
client.sendLetterAndAddress(description, letterContent, address);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertTrue(code > 0);

        System.out.println("New letter id is: " + code);
    }
}
```

Java: Send A Letter By Providing The Document Content

Find this sample in directory */test/com/postalmethods/client*

This sample demonstrates how to send a letter through PostalMethods using the Address Outside method when the content of the document are provided directly instead of reading the document from the file system.

```
package com.postalmethods.client;

import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.SendLetterAndAddressResponse;
import com.postalmethods.client.model.Address;
```

```java
import com.postalmethods.client.model.LetterContent;
import com.postalmethods.client.model.StringLetterContent;

/**
 * Integration test that calls the SendLetterAndAddress web method.
 * <p>
 * For details about PostalMethods.SendLetterAndAddress see
 * <a href="http://www.postalmethods.com/method/2009-02-26/SendLetterAndAddress">the
PostalMethods.SendLetterAndAddress reference</a>
 */
public class SendLetterAndAddressWithStringLetterContent {

    private PostalMethodsClient client;

    @Before
    public void setUp() {
        client = new PostalMethodsClientFactory().build();
    }

    @Test
    public void testSendLetterAndAddressAsString() {

        Address address = new Address("George Washington", "", "The White House",
"1600 Pennsylvania Ave", "", "Washington", "DC", "20500", "USA");
        String description = "Test letter with address.";

        LetterContent letterContent = new StringLetterContent( "<html><body><h2>I just
called to say 'I Love You'</h2></body></html>.", "HTML" );

        System.out.println("Making call to SendLetterAndAddress...");
        SendLetterAndAddressResponse response =
client.sendLetterAndAddress(description, letterContent, address);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertTrue(code > 0);

        System.out.println("New letter id is: " + code);
    }
}
```

Java: Send A Letter With Binary Content In Memory

Find this sample in directory */test/com/postalmethods/client*

This sample demonstrates how to send a letter through PostalMethods using the Address Outside method when the document to be sent is stored in memory as binary data as opposed to being read from the file system. This example is using an HTML file and shows how to encode is as binary but this sample is most useful when you create a binary document, such as a PDF, in memory.

```java
package com.postalmethods.client;

import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;
```

```
import com.postalmethods.client.api.SendLetterAndAddressResponse;
import com.postalmethods.client.model.Address;
import com.postalmethods.client.model.BytesLetterContent;
import com.postalmethods.client.model.LetterContent;

/**
 * Integration test that calls the SendLetterAndAddress web method.
 * <p>
 * For details about PostalMethods.SendLetterAndAddress see
 * <a href="http://www.postalmethods.com/method/2009-02-26/SendLetterAndAddress">the
PostalMethods.SendLetterAndAddress reference</a>
 */
public class SendLetterAndAddressWithBytesLetterContent {

    private PostalMethodsClient client;

    @Before
    public void setUp() {
        client = new PostalMethodsClientFactory().build();
    }

    @Test
    public void testSendLetterAndAddressAsBytes() {

        Address address = new Address("George Washington", "", "The White House",
"1600 Pennsylvania Ave", "", "Washington", "DC", "20500", "USA");
        String description = "Java: test letter with address";

        // Create document in memory.
        // This example uses a hard-coded HTML document.  You can create your
        // document any way you like as long as you provide a byte[] as shown
        // below.
        String documentType = "HTML";
        String document = "<html><body><h2>I just called to say 'I Love
You'</h2></body></html>.";
        LetterContent letterContent = new BytesLetterContent( document.getBytes(),
documentType );

        System.out.println("Making call to SendLetterAndAddress...");
        SendLetterAndAddressResponse response =
client.sendLetterAndAddress(description, letterContent, address);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertTrue(code > 0);

        System.out.println("New letter id is: " + code);
    }
}
```

Java: Send A Letter Using A Template

Find this sample in directory */test/com/postalmethods/client*

Instructions

1. Download the PostalMethods Java client. Simply download the jar file and place it in your classpath.
2. In order to send a letter using a template, you must first upload a template file using the PostalMethods UploadFile method. For this sample, download the invoice template sample (.MHT) and upload it to your account using the UploadFile method (see code sample). Remember the name you use to identify your uploaded file as you will need it below.
3. Below is a working Java program. There are three static String variables that you will need to provide values for - TEMPLATE_NAME, USERNAME, and PASSWORD.
4. Compile and run the class. If the call succeeds, you will see your letter in the PostalMethods Control Panel. If the response is a negative number, check the Web Service Status Codes section.

That is all that is required to send a letter using a pre-uploaded template and providing merge data. Using the other methods available in the PostalMethods API is just as easy.

```
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.postalmethods.client.PostalMethodsClient;
import com.postalmethods.client.PostalMethodsClientFactory;
import com.postalmethods.client.api.SendLetterAndAddressResponse;
import com.postalmethods.client.model.Address;
import com.postalmethods.client.model.LetterContent;
import com.postalmethods.client.model.TemplateLetterContent;

public class SendTemplateLetter {

    private static final String TEMPLATE_NAME = "CHANGE ME";
    private static final String USERNAME = "CHANGE ME";
    private static final String PASSWORD = "CHANGE ME";

    public static void main( String[] args ) {

        PostalMethodsClient client = new PostalMethodsClientFactory().build(USERNAME,
PASSWORD);

        Address address = new Address("George Washington", "", "The White House",
"1600 Pennsylvania Ave", "", "Washington", "DC", "20500", "USA");
        String description = "Invoice letter using a template.";

        System.out.println("Making call to SendLetterAndAddress...");
        List<String> remoteFilenames = Collections.emptyList();  // none for this test

        Map<String,Object> mergeData = new HashMap<String,Object>();
        mergeData.put( "ReturnAddress", "ABC Hardware Inc.<br/>93S. Jackson
Street<br/>Seattle  WA 98104-2818<br/>United States" );
        mergeData.put( "Date", "July 09, 2009" );
        mergeData.put( "InvoiceID", "8830023-RRX-3200985" );
        mergeData.put( "Payment", "Net 30" );
        mergeData.put( "MakeFor", "ABC Hardware" );
        mergeData.put( "ProductID1", "ED-2343" );
        mergeData.put( "Description1","Paslode Cordless Framing Nailer, 30 Degree
Paper Collated" );
        mergeData.put( "Quantity1","2" );
        mergeData.put( "Price1", "379.21" );
        mergeData.put( "Amount1", "758.42" );
```

```
        mergeData.put( "ProductID2", "FR-9485" );
        mergeData.put( "Description2", "Porta-Nails 2 In. Flooring Nails Master Carton
(1000 items)" );
        mergeData.put( "Quantity2", "10" );
        mergeData.put( "Price2", "11.20" );
        mergeData.put( "Amount2", "112.020" );

        LetterContent letterContent = new TemplateLetterContent( TEMPLATE_NAME,
mergeData, remoteFilenames );

        SendLetterAndAddressResponse response =
client.sendLetterAndAddress(description, letterContent, address);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        if(code > 0) {
            System.out.println("New letter id is: " + code);
        } else {
            System.out.println("Error, result code is: " + code);
        }
    }
}
```

Java: Get The Generated Letter with GetPDF()

Find this sample in directory */test/com/postalmethods/client*

```
package com.postalmethods.client;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.GetPdfResponse;
import com.postalmethods.client.api.SendLetterResponse;

/**
 * Integration test that calls the GetPDF web method.
 * <p>
 * For details about PostalMethods.GetPDF see
 * <a href="http://www.postalmethods.com/method/2009-02-26/GetPDF">the
PostalMethods.GetPDF reference</a>
 */
public class GetPdfTest {

    private PostalMethodsClient client;

    private int letterId;

    private String outFilename;

    @Before
    public void setUp() {
        client = new PostalMethodsClientFactory().build();
```

```
        // Send a sample letter so we have a letterId to check status on.
        SendLetterResponse sendResponse = client.sendLetter("Junit test letter",
TestConstants.SAMPLE_LETTER_FILENAME);
        int code = sendResponse.getResult().getCode();
        assertTrue(code > 0);
        this.letterId = code;
        System.out.println("Letter id is: " + this.letterId);

        String tmpDir = System.getProperty("java.io.tmpdir");
        String pathSeparator = System.getProperty("file.separator");
        outFilename = tmpDir + pathSeparator + new String(letterId + ".pdf");

        // This is a bit of a hack, but it does seem like we need to give the
        // PostalMethods API some time to process our test report before we
        // turn around and ask it for a rendered PDF.
        try {
            System.out.println("Waiting a bit to give PostalMethods a chance to
process our test letter.");
            Thread.sleep(20000);
            System.out.println("OK, should be ready now.");
        } catch (Exception e) {
        }
    }

    @Test
    public void testGetPdf() {

        System.out.println("Making call to GetPDF...");
        GetPdfResponse response = client.getPdf(letterId, outFilename);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertEquals(-3000, code);

        System.out.println("PDF saved to: " + outFilename);
    }
}
```

Java: Get The Status Of Letters And Postcards with GetStatus()

```
package com.postalmethods.client;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import java.util.List;
```

```java
import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.GetStatusResponse;
import com.postalmethods.client.api.SendLetterResponse;
import com.postalmethods.client.model.LetterStatus;

/**
 * Integration test that calls the GetStatus web method.
 * <p>
 * For details about PostalMethods.GetStatus see
 * <a href="http://www.postalmethods.com/method/2009-02-26/GetStatus">the
PostalMethods.GetStatus reference</a>
 */
public class GetStatusTest {

    private PostalMethodsClient client;

    private String letterId;

    @Before
    public void setUp() {

        client = new PostalMethodsClientFactory().build();

        // Send a sample letter so we have a letterId to check status on.
        SendLetterResponse sendResponse = client.sendLetter("Junit test letter",
TestConstants.SAMPLE_LETTER_FILENAME);
        int code = sendResponse.getResult().getCode();
        assertTrue(code > 0);
        this.letterId = Integer.toString(code);

        try {
            Thread.sleep(6000); // GetStatus calls should be called no more than
                                // once every six seconds.
        } catch (Exception e) {
        }
    }

    @Test
    public void testGetStatus() {

        System.out.println("Making call to GetStatus...");

        // GetStatus accepts an "ID" string which can include a single
        // item, multiple items, or a range of items.  Read the relevant documentation
        // for details (http://www.postalmethods.com/method/2009-02-26/GetStatus)
        // We will be looking for a single letter id, the letter created in
        // the setUp() method of this integration test.
        GetStatusResponse response = client.getStatus(letterId);
        System.out.println("Service call completed:\n" +
response.getResult().toString());
        int code = response.getResult().getCode();
        assertEquals(-3000, code); // success

        List<LetterStatus> letterStatusList = response.getLetterStatusList();
        assertEquals(1, letterStatusList.size());
        System.out.println(letterStatusList.get(0).toString());
    }
}
```

Java: Get Letters And Postcards' Delivery Details with GetDetails()

```
package com.postalmethods.client;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import java.util.List;

import org.junit.Before;
import org.junit.Test;

import  com.postalmethods.client.api.GetDetailsResponse;
import  com.postalmethods.client.api.SendLetterResponse;
import com.postalmethods.client.model.LetterDetails;

/**
 * Integration test that calls the GetDetails web method.
 * <p>
 * For details about PostalMethods.GetDetails see
 * <a href="http://www.postalmethods.com/method/2009-02-26/GetDetails">the
PostalMethods.GetDetails reference</a>
 */
public class GetDetailsTest {

    private PostalMethodsClient client;

    private String letterId;

    @Before
    public void setUp() {

        client = new PostalMethodsClientFactory().build();

        // Send a sample letter so we have a letterId to check status on.
        SendLetterResponse sendResponse = client.sendLetter("Junit test letter 1",
TestConstants.SAMPLE_LETTER_FILENAME);
        int code = sendResponse.getResult().getCode();
        assertTrue(code > 0);
        this.letterId = Integer.toString(code);
    }

    @Test
    public void testGetDetails() {

        System.out.println("Making call to GetDetails...");

        // GetDetails accepts an "ID" string which can include a single
        // item, multiple items, or a range of items.  Read the relevant documentation
        // for details (http://www.postalmethods.com/method/2009-02-26/GetDetails)
        // We will be looking for a single letter id, the letter created in
        // the setUp() method of this integration test.
        GetDetailsResponse response = client.getDetails(letterId);
        System.out.println("Service call completed:\n" +
response.getResult().toString());
```

```
        int code = response.getResult().getCode();
        assertEquals(-3000, code); // success

        List<LetterDetails> letterDetailsList = response.getLetterDetailsList();
        assertEquals(1, letterDetailsList.size());
        System.out.println(letterDetailsList.get(0).toString());
    }
}
```

Find this sample in directory */test/com/postalmethods/client*

```
package com.postalmethods.client;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import java.util.List;

import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.GetDetailsExtendedResponse;
import com.postalmethods.client.api.SendLetterResponse;
import com.postalmethods.client.model.LetterDetailsExtended;

/**
 * Integration test that calls the GetDetailsExtended web method.
 * <p>
 * For details about PostalMethods.GetDetailsExtended see
 * <a href="http://www.postalmethods.com/method/2009-02-26/GetDetailsExtended">the
PostalMethods.GetDetailsExtended reference</a>
 */
public class GetDetailsExtendedTest {

    private PostalMethodsClient client;

    private String letterId;

    @Before
    public void setUp() {

        client = new PostalMethodsClientFactory().build();

        // Send a sample letter so we have a letterId to check status on.
        SendLetterResponse sendResponse = client.sendLetter("Junit test letter 1",
TestConstants.SAMPLE_LETTER_FILENAME);
        int code = sendResponse.getResult().getCode();
        assertTrue(code > 0);
        this.letterId = Integer.toString(code);
    }

    @Test
    public void testGetDetailsExtended() {

        System.out.println("Making call to GetDetailsExtended...");

        // GetDetailsExtended accepts an "ID" string which can include a single
        // item, multiple items, or a range of items.  Read the relevant documentation
```

```
        // for details (http://www.postalmethods.com/method/2009-02-
26/GetDetailsExtended)
        // We will be looking for a single letter id, the letter created in
        // the setUp() method of this integration test.
        GetDetailsExtendedResponse response = client.getDetailsExtended(letterId);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertEquals(-3000, code); // success

        List<LetterDetailsExtended> letterDetailsList =
response.getLetterDetailsList();
        assertEquals(1, letterDetailsList.size());
        System.out.println(letterDetailsList.get(0).toString());
    }
}
```

Java: Cancel Delivery Of A Letter with CancelDelivery()

Find this sample in directory */test/com/postalmethods/client*

```
package com.postalmethods.client;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.CancelDeliveryResponse;
import com.postalmethods.client.api.SendLetterResponse;

/**
 * Integration test that calls the CancelDelivery web method. This method is
 * documented
 * <p>
 * For details about PostalMethods.CancelDelivery see
 * <a href="http://www.postalmethods.com/method/2009-02-26/CancelDelivery">the
PostalMethods.CancelDelivery reference</a>
 */
public class CancelDeliveryTest {

    private PostalMethodsClient client;

    private int letterId;

    @Before
    public void setUp() {

        client = new PostalMethodsClientFactory().build();

        // Send a sample letter so we have a letterId to check status on.
        SendLetterResponse sendResponse = client.sendLetter("Junit test letter",
TestConstants.SAMPLE_LETTER_FILENAME);
        int code = sendResponse.getResult().getCode();
        assertTrue(code > 0);
```

```
        this.letterId = code;
    }

    @Test
    public void testCancelDelivery() {

        System.out.println("Making call to CancelDelivery...");
        CancelDeliveryResponse response = client.cancelDelivery(letterId);
        int code = response.getResult().getCode();
        System.out.println("Service call completed:\n" +
response.getResult().toString());
        assertEquals(-3000, code); // success
    }
}
```

Java: Delete an uploaded file using DeleteUploadedFile()

Find this sample in directory */test/com/postalmethods/client*

```
package com.postalmethods.client;

import static org.junit.Assert.assertEquals;

import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.DeleteUploadedFileResponse;
import com.postalmethods.client.api.UploadFileResponse;

/**
 * Integration test that calls the DeleteUploadedFile web method.
 * <p>
 * For details about PostalMethods.DeleteUploadFile see
 * <a href="http://www.postalmethods.com/method/2009-02-26/DeleteUploadedFile">the
PostalMethods.DeleteUploadedFile reference</a>
 */
public class DeleteUploadedFileTest {

    private PostalMethodsClient client;

    @Before
    public void setUp() {
        client = new PostalMethodsClientFactory().build();

        // Upload a sample file so we have something to delete.
        System.out.println("Making call to UploadFile...");

        String description = "This is only a test.";
        UploadFileResponse response =
client.uploadFile(TestConstants.SAMPLE_UPLOAD_FILE,
TestConstants.SAMPLE_POSTCARD_IMAGE_FRONT_LOCAL, "User", description, true);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertEquals(-3000, code); // success
    }
```

```
    @Test
    public void testDeleteUploadedFile() {
        System.out.println("Making call to DeleteUploadedFile...");

        DeleteUploadedFileResponse response =
client.deleteUploadedFile(TestConstants.SAMPLE_UPLOAD_FILE);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertEquals(-3000, code); // success
    }
}
```

Java: Get Details Of Uploaded Files Using GetUploadedFileDetails()

Find this sample in directory */test/com/postalmethods/client*

```
package com.postalmethods.client;

import static org.junit.Assert.assertEquals;

import java.util.List;

import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.GetUploadedFileDetailsResponse;
import com.postalmethods.client.model.FileDetails;

/**
 * Integration test that calls the GetUploadedFileDetails web method.
 * <p>
 * For details about PostalMethods.GetUploadFileDetails see
 * <a href="http://www.postalmethods.com/method/2009-02-26/GetUploadedFileDetails">the
PostalMethods.GetUploadedFileDetails reference</a>
 */
public class GetUploadedFileDetailsTest {

    private PostalMethodsClient client;

    @Before
    public void setUp() {
        client = new PostalMethodsClientFactory().build();
    }

    @Test
    public void testGetUploadedFileDetails() {
        System.out.println("Making call to GetUploadedFileDetails...");

        GetUploadedFileDetailsResponse response = client.getUploadedFileDetails();
```

```
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        List<FileDetails> fileDetails = response.getFileDetailsList();
        for( FileDetails curr : fileDetails ) {
            System.out.println( curr );
        }

        int code = response.getResult().getCode();
        assertEquals(-3000, code); // success
    }
}
```

Java: Upload A File For Future Use Using UploadFile()

Find this sample in directory */test/com/postalmethods/client*

```
package com.postalmethods.client;

import static org.junit.Assert.assertEquals;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import com.postalmethods.client.api.DeleteUploadedFileResponse;
import com.postalmethods.client.api.UploadFileResponse;

/**
 * Integration test that calls the UploadFile web method.
 * <p>
 * For details about PostalMethods.UploadFile see
 * <a href="http://www.postalmethods.com/method/2009-02-26/UploadFile">the
PostalMethods.UploadFile reference</a>
 */
public class UploadFileTest {

    private PostalMethodsClient client;

    @Before
    public void setUp() {
        client = new PostalMethodsClientFactory().build();
    }

    @After
    public void tearDown() {

        // Delete the file we uploaded during our test.
        System.out.println("Making call to DeleteUploadedFile...");

        DeleteUploadedFileResponse response =
client.deleteUploadedFile(TestConstants.SAMPLE_UPLOAD_FILE);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertEquals(-3000, code); // success
```

```
    }

    @Test
    public void testUploadFile() {
        System.out.println("Making call to UploadFile...");

        String description = "This is only a test.";
        UploadFileResponse response =
client.uploadFile(TestConstants.SAMPLE_UPLOAD_FILE,
TestConstants.SAMPLE_POSTCARD_IMAGE_FRONT_LOCAL, "User", description, true);
        System.out.println("Service call completed:\n" +
response.getResult().toString());

        int code = response.getResult().getCode();
        assertEquals(-3000, code); // success
    }
}
```