Technical Challenge

Background

Imagine that we have a website at http://pdfbuilder.luminos.software

On this site, the user can:

- Upload an Excel spreadsheet (.xls format).
- Upload a PDF template which has form fields on it.
- Create “mapping instructions” that associate columns in the spreadsheet with form fields in the PDF.
- Generate multiple PDFs by filling out a copy of the PDF template with data extracted from each row of the uploaded Excel file by clicking a ‘Generate’ button.

Architecture

Because the PDF generation might take a long time (there could be hundreds of rows in the Excel), when the ‘Generate’ button is clicked, a document generation process will be added to a queue where execution will commence at some later time. The user will be informed “Your PDFs are generating, you will receive an email when they’re done”.

Objective

Your task is to build the tool that the document generation process will invoke to generate the PDFs from the Excel spreadsheet.

Specifically:
- You will write a command line Java application called **`XlsPdfGenerator`**

Overview

`XlsPdfGenerator` is an application to generate multiple PDFs by filling out a template PDF with data from an Excel spreadsheet.

```
usage: XlsPdfGenerator srcXls srcPdf srcJson tgtDir

  srcXls - full path to a source Excel file (.xls format)
  srcPdf - full path to a template PDF form
  srcJson - full path to a JSON file of mapping instructions
  tgtDir - the target directory where generated PDF documents will be saved
```

We will assume the following:

- The source Excel and PDF files have been uploaded to the local filesystem somewhere.
- The spreadsheet will contain (a) a header row of column labels, and then (b) a bunch of data.
- The template PDF will be ready configured with form fields that can be programmatically populated.

- The application will output generated PDFs to the local filesystem (in **tgtDir**).

Behaviour

In order to define how the PDF should be populated, the website allows user to create “mapping rules”. When the background job runs, the mapping rules that the user created will be passed into **`XlsPdfGenerator`** as a string. For simplicity, we’ll assume that a file containing an array of JSON objects can be found on the filesystem.

There are 6 types of “mapping rules” that a user can create:

- **fill** - fills out a named form field in the PDF with a string value.
- **check** - adds a check mark to a named form field on the PDF.
- **fillFromExcel** - fills out a named form field in the PDF using the value found in the named Excel column.
- **checkFromExcel** - adds a check mark to a named checkbox on the PDF using the value found in the named Excel column (the value found in Excel must be ‘true’, ’yes’ or 1 in order to check the checkbox).
- **conditionalFillFromExcel** - fills out a named form field in the PDF IF a condition is true. For simplicity we will limit the predicates used for the condition to ‘equals’ and ‘contains’. Equals is a literal equality test, contains looks for a substring.
- **conditionalCheckFromExcel** - adds a check mark to a named form field IF a condition is true. For simplicity we will limit the predicates used for the condition to ‘equals’ and ‘contains’. Equals is a literal equality test, contains looks for a substring.

Here are some example JSON fragments for each of the mapping rules:

```
/* Fill
 * ——
 * this will fill the PDF field ‘textFirstName’ with the value ‘Calin’.
 *
 */
{
  "action" : "fill",
  "fieldName" : "textFirstName",
  "value" : "Calin"
}
```

```
/* Check
 * ——
 * this will check the PDF checkbox ‘isCool’.
 *
 */
{
  "action" : "check",
  "fieldName" : "isCool"
}
```

```
/* Fill from Excel
 * ———
 * this will fill the PDF field 'textFirstName' with the content found in
 * the Excel sheet column labelled 'firstName'.
 * /
{
  "action" : "fillFromExcel",
  "colName" : "firstName",
  "fieldName" : "textFirstName"
}
```

```
/* Check From Excel
 * ———
 * this will check the PDF checkbox 'isSMS' if the column labelled 'sms'
 * in the Excel file equals 'true', 'yes' or 1.
 * /
{
  "action" : "checkFromExcel",
  "colName" : "sms",
  "fieldName" : "isSms"
}
```

```
/* Conditional Fill From Excel
 * ———
 * this will fill the PDF field 'theGroovyField' with the value 'We are
 * groovy!' if the value found in the Excel column labelled
 * 'myColumnName' contains the string "groovy".
 * /
{
  "action" : "conditionalFillFromExcel",
  "colName" : "myColumnName",
  "predicate" : "contains",
  "testVal" : "groovy",
  "value" : "We are groovy!",
  "fieldName" : "theGroovyField"
}
```

```
/* Conditional Check From Excel
 * ———
 * this will check the PDF checkbox 'isGroovy' if the value found in the
 * Excel column labelled 'myColumnName' equals the string "groovy".
 * /
{
  "action" : "conditionalCheckFromExcel",
  "colName" : "myColumnName",
  "predicate" : "equals",
  "testVal" : "groovy",
  "fieldName" : "isGroovy"
}
```

Resources

Attached to this challenge is a source Excel and PDF file to be used for completing this challenge.

- challenge-source.xls
- challenge-source.pdf

In the **annex** you will find the mapping instructions that must be executed to complete this challenge. (Hint: create a JSON file of mapping instructions based on what you find in the annex).

What we're looking for

- Clean, elegantly structure code, appropriately commented with an eye to easy future reuse.
- Appropriate use of core Java classes for data structures.
- Appropriate use of error-handling.
- Appropriate use of third-party libraries (hint: you'll find some useful libraries for working with PDFs on apache.org)
- Some automated testing.

Annex

Excel column	PDF field	Notes
transactionid	regristreringsnummer	Fill this field from Excel.
	ortOchDatum	Always fill this form field “NOT APPLICABLE”
consultant	idNummer	Fill this field from Excel.
bill_to_firstname	fornamn	Fill this field from Excel.
bill_to_lastname	efternamn	Fill this field from Excel.
bill_to_street	adress	Fill this field from Excel.
	postnummer	Always fill this form field “ZIP UNKNOWN”
city	ort	Fill this field from Excel.
bill_to_email	epost	Fill this field from Excel.
bill_to_phonefast	telefonMobil	Fill this field from Excel.
bill_to_bank	bankensNamn	Fill this field from Excel.
bill_to_clearingnummer	clearingnummer	Fill this field from Excel.
bill_to_kontonummer	kontonummer	Fill this field from Excel.
	personnummer	Always fill this form field “NOT APPLICABLE”
	kontrolluppgifterHamtadeFran	Always fill this form field “NOT APPLICABLE”
Bill_to_news_SMS	checkSms	Check this checkbox if Bill_to_news_SMS contains the string ‘Ja tack’
Bill_to_news_mail	checkEpost	Check this checkbox if Bill_to_news_mail contains the string ‘Ja tack’
product	100kr	Always check this checkbox.
product	150kr	Check this checkbox if product equals ‘150kr’
product	200kr	Check this checkbox if product equals ‘200kr’
product	250kr	Check this checkbox if product equals ‘250kr’
product	300kr	Check this checkbox if product equals ‘300kr’