Instruction: Implement the following user story.

Business Value

So that ABC consumers will be able to receive accounting line details for a booking

As an accounting line detail service provider

I should be able to capture information and populate it in the corporate accounting line payload

Scope

Limited to accounting line only

Acceptance Criteria

1. Capture accounting line details from the value returned by XYZ identification field

	ABC Path	**XYZ source**
	accountingLine/accountingLineID	AccountingLine.XYZ/id
	accountingLine/accountingLineStatus	set to ACTIVE upon parsing
	accountingLine/accountingVendorCode	AccountingLine.XYZ/AccountingVendorCode
	accountingLine/airlineCode	
	accountingLine/chargeCategoryCode	AccountingLine.XYZ /ChargeCategoryCoded
	accountingLine/formattedReceiptNumber	
	accountingLine/invoiceNumber	AccountingLine.XYZ/OriginalInvoice
	accountingLine/linkCode	AccountingLine.XYZ/LinkCode
	accountingLine/numberOfConjunctedDocuments	AccountingLine.XYZ/NumberOfConjunctedDocuments *numberOfConjunctedDocuments should be parsed based on pattern of 1 or more digits followed by "-" and any character ie. "2-NN"*
	accountingLine/originalTicketNumber	AccountingLine.XYZ/OriginalTicketNumber
	accountingLine/receiptNumber	AccountingLine.XYZ /DocumentNumber
0..*	accountingLine/segmentRefIDList	AccountingLine.XYZ/SegmentNumber

	accountingLine/travelerName	AccountingLine.XYZ/PassengerName
0..*	accountingLine/travelerRefIDList	1
	accountingLine/typeIndicator	AccountingLine.XYZ/TypeIndicator
	accountingLine/elementNumber	AccountingLine.XYZ/index
	accountingLine/fareApplication	AccountingLine.XYZ/FareApplication
	accountingLine/baseFare	AccountingLine.XYZ/BaseFare
	accountingLine/taxAmount	AccountingLine.XYZ/TaxAmount
	accountingLine/totalTaxAmount	TaxAmount + totalTaxSurcharge
	accountingLine/totalTaxSurcharge	GSTAmount + QSTAmount
	accountingLine/gstAmount	AccountingLine.XYZ/GSTAmount
	accountingLine/gstCode	AccountingLine.XYZ/QSTCode
	accountingLine/qstAmount	AccountingLine.XYZ/QSTAmount
	accountingLine/qstCode	AccountingLine.XYZ/GSTCode
	accountingLine/commission/amount	AccountingLine.XYZ/CommissionPercentage
	accountingLine/commission/percentage	AccountingLine.XYZ/CommissionAmount
	accountingLine/freeFormText	AccountingLine.XYZ/FreeFormText

2. Exceptions encountered during parsing and/or data manipulation to capture details should be handled and logged
    1. When information is not available or an error occurs, exception should be handled accordingly and should be able to proceed to the next flow
3. Captured accounting line details should be exposed by API and can be verified using a REST client

Acceptance Test:

Given an accounting line payload wherein accounting line details is present in XYZ
When ABC process the accounting line
Then accounting line details should be populated in ABC accounting line payload

Given the ABC accounting line payload from the conversion
When ABC process the accounting line
Then accounting line details should be viewed via API call using a REST client

Dev Notes:

1. jar/zip file will be provided for model classes
2. Create a Spring Boot application using gradle
3. Create a Builder class that does data conversion
4. Expose a Spring REST API that returns JSON AccountingLine given AccountingLineXYZ xml (see sampleAccountingLineXYZ.xml)
5. Implement Best Practices: Junit, Error Handling, Proper Convention, Use Java 8 Features
6. Package it to jar file

BA Notes:

Attached sample XYZ payload:

```
<AccountingLine id="572" index="1" op="C" elementId="PNR-572">

        <TypeIndicator>TYPE-01</TypeIndicator>

        <FareApplication>ONE</FareApplication>

        <FormOfPaymentCode>FormOfPaymentCode0</FormOfPaymentCode>

        <LinkCode>01</LinkCode>

        <AccountingVendorCode>ABC</AccountingVendorCode>

        <ChargeCategoryCoded>TEST</ChargeCategoryCoded>

        <AirlineDesignator>AA</AirlineDesignator>

        <DocumentNumber>3889081143</DocumentNumber>

        <CommissionPercentage>2</CommissionPercentage>

        <CommissionAmount>2</CommissionAmount>

        <BaseFare>1252.00</BaseFare>

        <TaxPercentage>1</TaxPercentage>
        <TaxAmount>201.38</TaxAmount>

        <TaxSurchargeCode2>VAT</TaxSurchargeCode2>

        <GSTCode>GST</GSTCode>

        <GSTAmount>10</GSTAmount>

        <GSTPercent>1</GSTPercent>

        <QSTCode>QST</QSTCode>

        <QSTAmount>10</QSTAmount>

        <QSTPercent>1</QSTPercent>

        <CreditCardNumber>123456789</CreditCardNumber>

        <CreditCardCode>AX0</CreditCardCode>

        <PassengerName>Thanos</PassengerName>

        <NumberOfConjunctedDocuments>15-FEE-EXP$08</NumberOfConjunctedDocuments>

        <NumberOfCoupons>1</NumberOfCoupons>

        <OriginalTicketNumber>123456</OriginalTicketNumber>

        <OriginalDateOfIssue>2021-01-28</OriginalDateOfIssue>

        <OriginalPlaceOfIssue>MNL</OriginalPlaceOfIssue>

        <FullPartialExchangeIndicator>PART</FullPartialExchangeIndicator>

        <OriginalInvoice>00001</OriginalInvoice>

        <TarriffBasis>TarriffBasis0</TarriffBasis>

        <FreeFormText>HELLO WORLD</FreeFormText>

        <CurrencyCode>USD</CurrencyCode>

        <SegmentType>AIR</SegmentType>

        <SegmentNumber>1</SegmentNumber>

</AccountingLine>
```

* This story will cover the correct capture logic for baseFare, totalTaxAmount, and taxAmount.