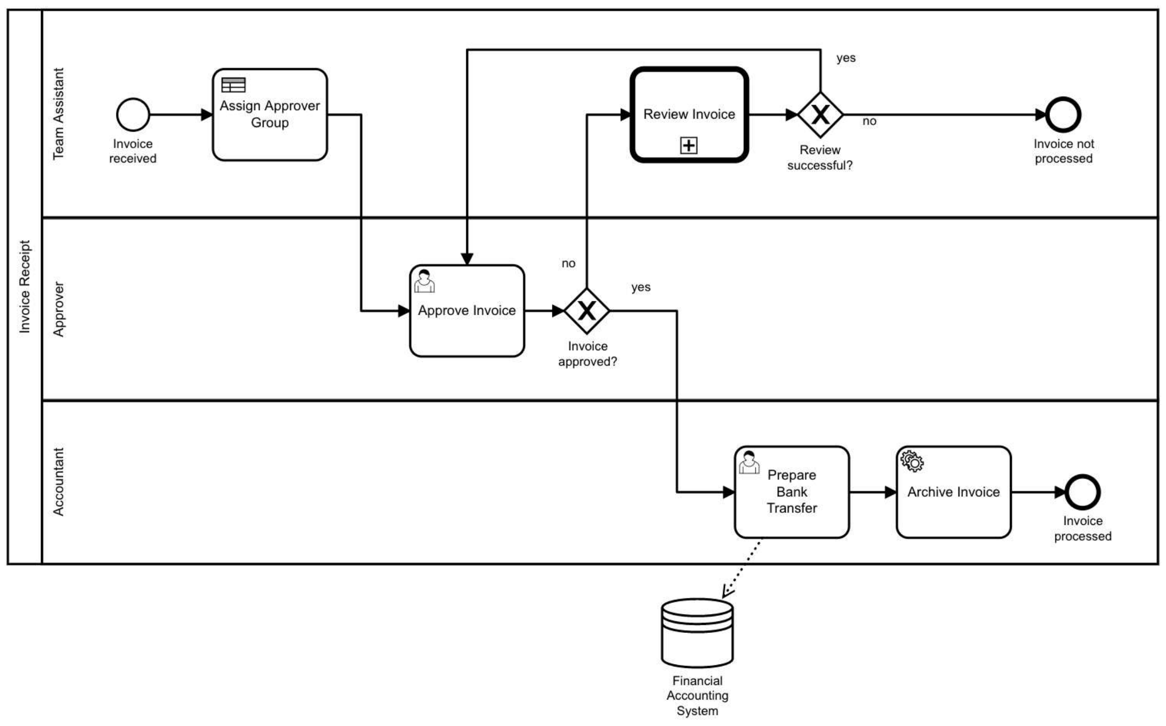
Introduction

This coding challenge is based on a feature we actually implemented in one of our products (but it is, of course, a very simplified version thereof). Hence it allows you to get a glimpse of our domain and the challenges you can expect when working at Camunda. And it allows us to get a sense of your programming style as well. Win / win! :-)

At Camunda we create software that helps our customers automate their business processes. We built our products on the foundation of BPMN, a simple but very powerful notation for describing and executing workflows. Here is a BPMN diagram for an exemplary 'invoice approval' workflow:



As you can see, a workflow is made of different kinds of elements, which we call *flow nodes,* and connections between them, which we call *sequence flows*. In that sense, a BPMN diagram can be described as a *directed graph*, which can be traversed programmatically.

BPMN diagrams are stored as XML files, which comply with the BPMN 2.0 specification released by the Object Modeling Group (OMG). You do not need to know anything about BPMN and the underlying XML structure to complete this coding challenge!

Your task

Your task is to develop a Java program that does the following:

- It fetches the XML representation of the exemplary ‘invoice approval’ BPMN diagram depicted above from a remote server.
- It parses the XML into a traversable data structure.
- It finds one possible path on the graph between a given start node and a given end node.
- It prints out the IDs of all nodes on the found path to System.out.

Hints

- There is no need to handle any BPMN-specific aspect of the graph. You can simply treat it as an ordinary graph that consists of *nodes* and *edges*.

- To fetch the XML representation of the BPMN file, do a GET request to https://n35ro2ic4d.execute-api.eu-central-1.amazonaws.com/prod/engine-rest/process-definition/key/invoice/xml

  - The response contains a JSON object with two string attributes *id* and *bpmn20Xml*. The attribute named *bpmn20Xml* contains the XML representation of the diagram.

- Do not parse the XML yourself but rather use the *Model API* that we provide.

  - To use it, add the following dependency to your project:
    `groupId: org.camunda.bpm, artifactId: camunda-engine, version: 7.9.0`

  - Have a look at the documentation on how to read the model/diagram from a stream.

- Implement a main method which will be invoked with two arguments: One is the *start flow node ID* and the other one the *end flow node ID*. Remember that the diagram can be described as a graph. Each ID represents a node in the graph. Use the two node IDs provided to find one possible path between them and print out the path you found to System.out. You don’t necessarily need to find the *shortest* path, but your path is not allowed to contain loops (e.g. contain the same node more than once). Again, consult the documentation to find out how to get a flow node for an ID and how to traverse the graph. On the last page of this document, we provide the exemplary *invoice approval* diagram with all flow nodes and their IDs for your reference.

- Invoking your program in the following way

  ```
  java -jar YourApp.jar approveInvoice invoiceProcessed
  ```

  Should result in *exactly* the following output (with no line-breaks):

  ```
  The path from approveInvoice to invoiceProcessed is:
  [approveInvoice, invoice_approved, prepareBankTransfer,
  ServiceTask_1, invoiceProcessed]
  ```
- Errors and no found path *must* quit your program with exit code -1.

Good to know before you start

- You are free to choose any libraries to achieve the goals (e.g. REST client, JSON Library, ...).
- Your application can be as simplistic as possible.
- Don't write any test cases! (We usually value tests very much, but for this exercise, we don't require them)
- Documentation is also not required.
- The code should be clean but do not over-engineer it. If everything fits in one class, that's totally fine.

Additional Questions

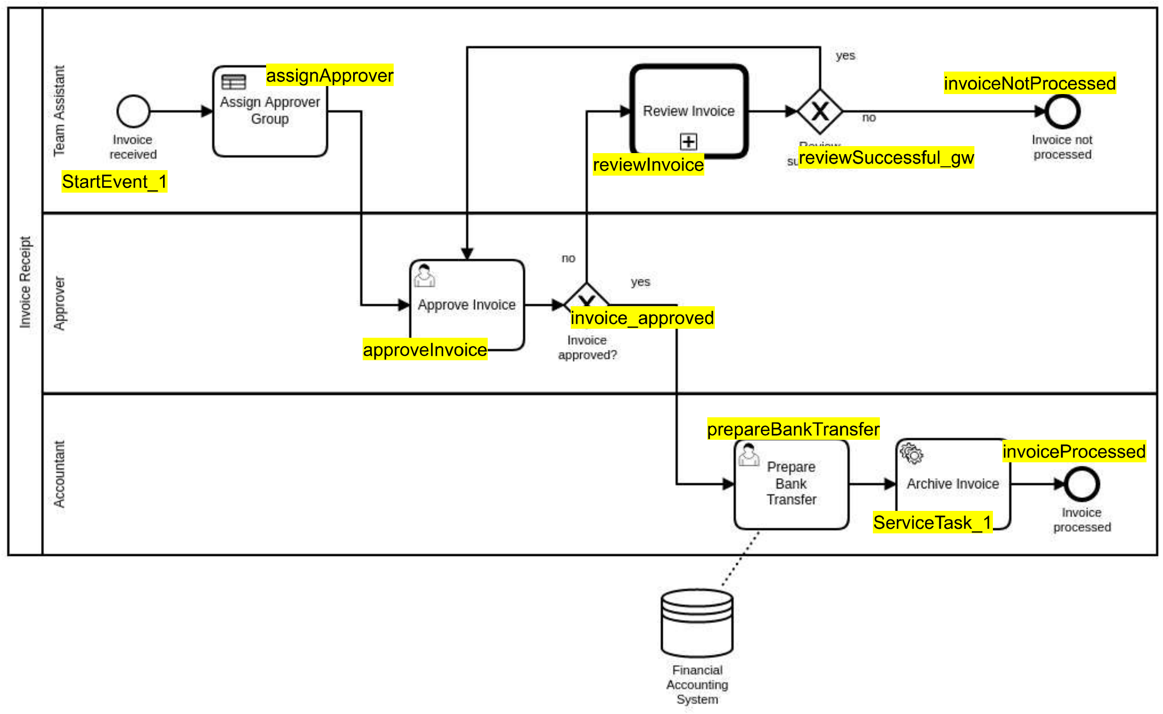
1) How long did it take you to solve the exercise? (Please be honest, we evaluate your answer to this question based on your experience.)
2) What are some edge cases that you might not have considered yet?
3) What kind of problems/limitations can you think of in terms of your implementation?

Submit your solution

When you are done, please create a .zip archive with your code (no need for the .jar file) and email it back to us as an attachment or upload it to a file hosting service (e.g. Dropbox or Google Drive) and provide us the link. Please include the answers to the above questions in your email body.

Happy coding!

Appendix: BPMN diagram with flow node IDs


Invoice Receipt
Team Assistant
Approver
Accountant
Invoice received
StartEvent_1
Assign Approver Group
assignApprover
Review Invoice
reviewInvoice
yes
no
reviewSuccessful_gw
invoiceNotProcessed
Invoice not processed
Approve Invoice
approveInvoice
no
yes
invoice_approved
Invoice approved?
prepareBankTransfer
Prepare Bank Transfer
Archive Invoice
ServiceTask_1
invoiceProcessed
Invoice processed
Financial Accounting System