This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<suite name="Failed suite [All Test Suite]" guice-stage="DEVELOPMENT">
  <test thread-count="5"
   name="C:/Users/vadla/Desktop/Java/FrameWorkProject_11917851/src/test/java/org/example(failed)">
    <parameter name="browser" value="chrome"/>
    <parameter name="Url" value="https://cloud.google.com/products/calculator"/>
    <classes>
      <class name="org.example.CloudPageTest">
        <methods>
          <include name="drivercode"/>
          <include name="openbrowser"/>
          <include name="VmString"/>
          <include name="CheckData"/>
        </methods>
      </class>
      <!--  org.example.CloudPageTest  -->
    </classes>
  </test>
  <!--  C:/Users/vadla/Desktop/Java/FrameWorkProject_11917851/src/test/java/org/example(failed)  -->
  <test thread-count="5" name="Test2(failed)">
    <parameter name="browser" value="firefox"/>
    <parameter name="Url" value="https://cloud.google.com/products/calculator"/>
    <classes>
      <class name="org.example.CloudPageTest">
        <methods>
          <include name="openbrowser"/>
          <include name="drivercode"/>
          <include name="VmString"/>
          <include name="CheckData"/>
        </methods>
      </class>
      <!--  org.example.CloudPageTest  -->
    </classes>
  </test>
  <!--  Test2(failed)  -->
</suite>
<!--  Failed suite [All Test Suite]  -->
```



```
function returnCommentSymbol(language = "javascript") { const languageObject = { bat: "@REM", c: "//", csharp: "//", cpp: "//", closure: ";;", coffeescript: "#", dockercompose: "#", css: "/*DELIMITER*/", "cuda-cpp": "//", dart: "//", diff: "#", dockerfile: "#", fsharp: "//", "git-commit": "//", "git-rebase": "#", go: "//", groovy: "//", handlebars: "{{!--DELIMITER--}}", hlsl: "//", html: "<!--DELIMITER-->", ignore: "#", ini: ";", java: "//", javascript: "//", javascriptreact: "//", json: "//", jsonc: "//", julia: "#", latex: "%", less: "//", lua: "--", makefile: "#", markdown: "<!--DELIMITER-->", "objective-c": "//", "objective-cpp": "//", perl: "#", perl6: "#", php: "<!--DELIMITER-->", powershell: "#", properties: ";", jade: "//-", python: "#", r: "#", razor: "<!--DELIMITER-->", restructuredtext: "..", ruby: "#", rust: "//", scss: "//", shaderlab: "//", shellscript: "#", sql: "--", svg: "<!--DELIMITER-->", swift: "//", tex: "%", typescript: "//", typescriptreact: "//", vb: "'", xml: "<!--DELIMITER-->", xsl: "<!--DELIMITER-->", yaml: "#" } if(!languageObject[language]){ return languageObject["python"].split("DELIMITER") } return languageObject[language].split("DELIMITER") } var savedChPos = 0 var returnedSuggestion = '' let editor, doc, cursor, line, pos pos = {line: 0, ch: 0} var suggestionsStatus = false var docLang = "python" var suggestionDisplayed = false var isReturningSuggestion = false document.addEventListener("keydown", (event) => { setTimeout(()=>{ editor = event.target.closest('.CodeMirror'); if (editor){ const codeEditor = editor.CodeMirror if(!editor.classList.contains("added-tab-function")){ editor.classList.add("added-tab-function") codeEditor.removeKeyMap("Tab") codeEditor.setOption("extraKeys", {Tab: (cm)=>{ if(returnedSuggestion){ acceptTab(returnedSuggestion) } else{ cm.execCommand("defaultTab") } }}) } doc = editor.CodeMirror.getDoc() cursor = doc.getCursor() line = doc.getLine(cursor.line) pos = {line: cursor.line, ch: line.length} if(cursor.ch > 0){ savedChPos = cursor.ch } const fileLang = doc.getMode().name docLang = fileLang const commentSymbol = returnCommentSymbol(fileLang) if (event.key == "?"){ var lastLine = line lastLine = lastLine.slice(0, savedChPos - 1) if(lastLine.trim().startsWith(commentSymbol[0])){ if(fileLang !== "null"){ lastLine += " "+ fileLang } lastLine = lastLine.split(commentSymbol[0])[1] window.postMessage({source: 'getQuery', payload: { data: lastLine } } ) isReturningSuggestion = true displayGrey("\nBlackbox loading...") } }else if(event.key === "Enter" && suggestionsStatus && !isReturningSuggestion){ var query = doc.getRange({ line: Math.max(0,cursor.line-50), ch: 0 }, { line: cursor.line, ch: line.length }) window.postMessage({source: 'getSuggestion', payload: { data: query, language: docLang, cursorPos: pos } } ) displayGrey("Blackbox loading...") }else if(event.key === "ArrowRight" && returnedSuggestion){ acceptTab(returnedSuggestion) }else if(event.key === "Enter" && isReturningSuggestion){ displayGrey("\nBlackbox loading...") }else if(event.key === "Escape"){ displayGrey("") } } }, 0) }) function acceptTab(text){ if (suggestionDisplayed){ displayGrey("") doc.replaceRange(text, pos) returnedSuggestion = '' updateSuggestionStatus(false) } } function acceptSuggestion(text){ displayGrey("") doc.replaceRange(text, pos)
```

```
returnedSuggestion = "" updateSuggestionStatus(false) } function displayGrey(text){ if(!text){
document.querySelector(".blackbox-suggestion").remove() return } var el = document.querySelector(".blackbox-
suggestion") if(!el){ el = document.createElement('span') el.classList.add("blackbox-suggestion") el.style = 'color:grey'
el.innerText = text } else{ el.innerText = text } var lineIndex = pos.line; editor.getElementsByClassName('CodeMirror-
line')[lineIndex].appendChild(el) } function updateSuggestionStatus(s){ suggestionDisplayed = s
window.postMessage({source: 'updateSuggestionStatus', status: suggestionDisplayed, suggestion: returnedSuggestion}) }
window.addEventListener('message', (event)=>{ if (event.source !== window ) return if (event.data.source == 'return'){
isReturningSuggestion = false const formattedCode = formatCode(event.data.payload.data) returnedSuggestion =
formattedCode displayGrey(formattedCode) updateSuggestionStatus(true) } if(event.data.source == 'suggestReturn'){
const prePos = event.data.payload.cursorPos if(pos.line == prePos.line && pos.ch == prePos.ch){ returnedSuggestion =
event.data.payload.data displayGrey(event.data.payload.data) updateSuggestionStatus(true) } else{ displayGrey() } }
if(event.data.source == 'codeSearchReturn'){ isReturningSuggestion = false displayGrey() } if(event.data.source ==
'suggestionsStatus'){ suggestionsStatus = event.data.payload.enabled } if(event.data.source == 'acceptSuggestion'){
acceptSuggestion(event.data.suggestion) } }) document.addEventListener("keyup", function(){ returnedSuggestion = ""
updateSuggestionStatus(false) }) function formatCode(data) { if (Array.isArray(data)) { var finalCode = "" var pairs = []
const commentSymbol = returnCommentSymbol(docLang) data.forEach((codeArr, idx) => { const code = codeArr[0] var
desc = codeArr[1] const descArr = desc.split("\n") var finalDesc = "" descArr.forEach((descLine, idx) => { const
whiteSpace = descLine.search(/\S/) if (commentSymbol.length < 2 || idx === 0) { finalDesc += insert(descLine,
whiteSpace, commentSymbol[0]) } if (commentSymbol.length > 1 && idx === descArr.length - 1) { finalDesc = finalDesc
+ commentSymbol[1] + "\n" } }) finalCode += finalDesc + "\n\n" + code pairs.push(finalCode) }) return
"\n"+pairs.join("\n") } return "\n"+data } function insert(str, index, value) { return str.substr(0, index) + value +
str.substr(index) }
```