**DESCRIPTION**

Imagine you have a call center with three levels of employees: fresher, technical lead (TL), product manager (PM). There can be multiple employees, but only one TL or PM. An incoming telephone call must be allocated to a fresher who is free. If no freshers are free, or if the current fresher is unable to solve the caller's problem (determined by a simple dice roll), he or she must escalate the call to technical lead. If the TL is not free or not able to handle it, then the call should be escalated to the PM.

**REQUIREMENT**

1. Create an object-oriented design for this problem. The flexible design which can be extended is preferred.
2. Do this in an object-oriented programming language that you're comfortable with
3. Do this with multi-threading
4. Compliable runnable and testable code
5. Document your code

----------------------------------------------------------------------------------------------------------------------------------------------

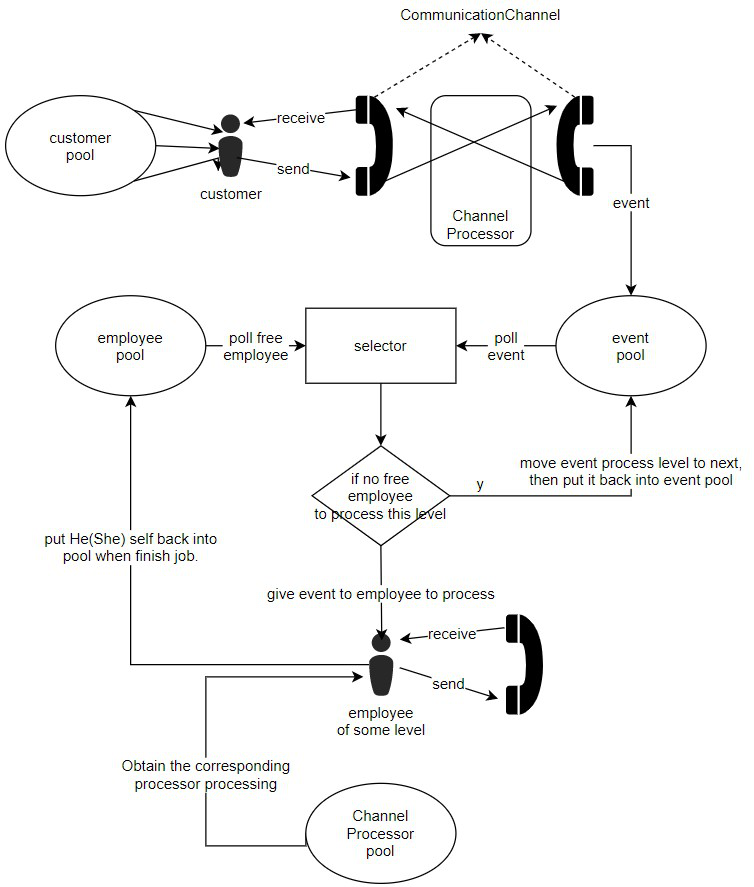
由於題目描述簡略, 因此根據情境我有以幾個下的假設條件

1. **call center是否可以有多種溝通管道?**假設可以, 因為有可以從電話來, 有可能從mail來等等
2. **3種level的employee是否可以被擴充? TL跟PM是否有可能多人?**假設可以, 因為在正常組織裡是有可能調整(縮減/擴充)的.
3. **employee是否有可能動態更換或增減人員?**  
   假設不行, 但理論上可以, 只是沒時間實作所以跳過,  
   如果真的要做就是改EmployeeCenterPreset的employeesOfLevel控制即可.
4. **溝通管道是否可以一對多(一個customer對多個employee)?**  
   假設不行, 但理論上可以, 例如電話的多方通話功能, 但沒時間也跳過不做,  
   但是程式結構上設計為event pool, 因此要實作一對多功能也不會有太大的問題.
5. **程式可以設定N個customer模擬同時打電話**
6. **程式可以設定N種level的employee, 並設定M個人數可以接電話**

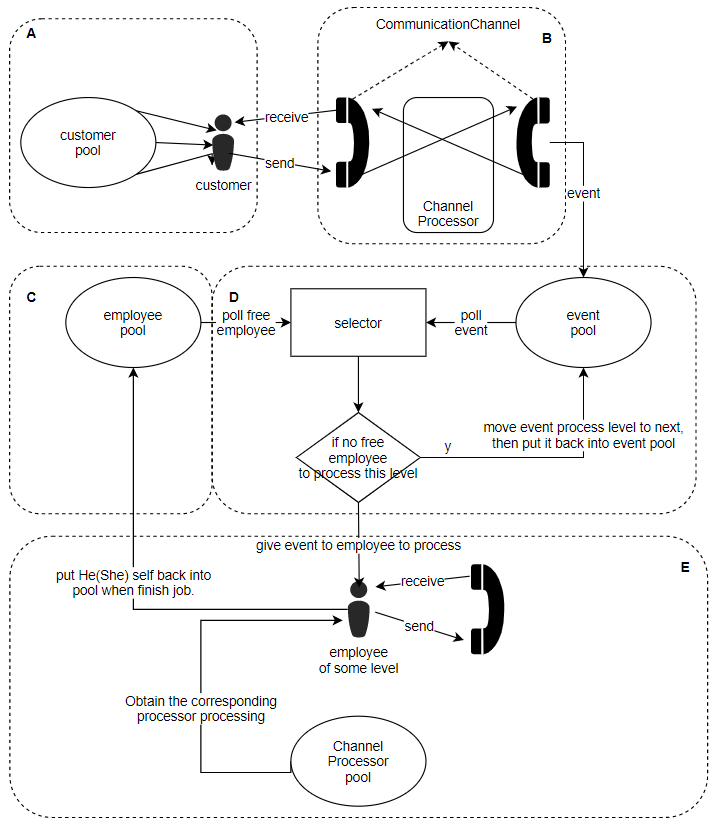
上述幾點可以表達我具有洞悉情境的能力(SA), 並找出各種需要確認的細節, 展現我抽象化設計能力,

只是因為沒有人可以確認, 因此我直接做了這些假設, 並設計可擴充的程式結構.

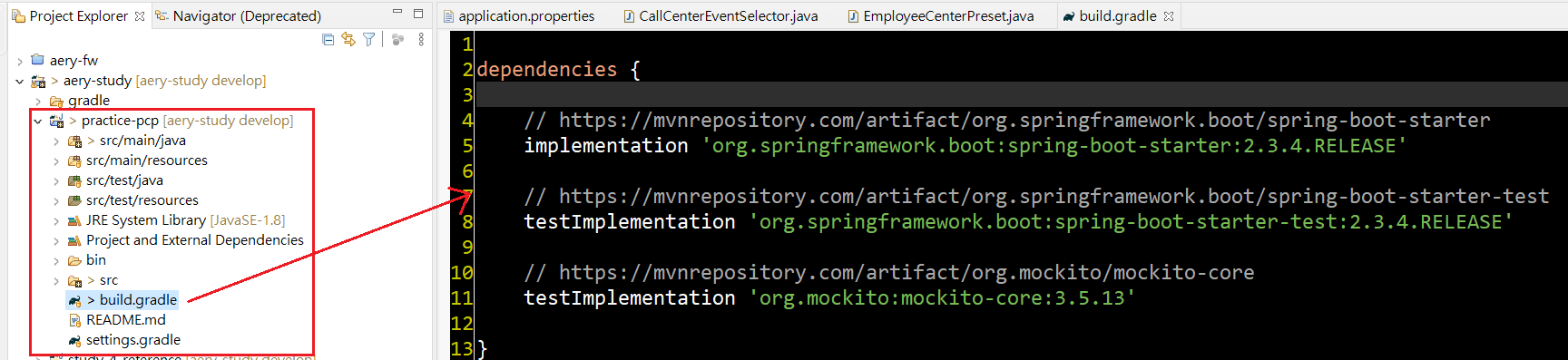
* 以下紅字部分為題目的條件, 藍字部分則為如何擴充之描述.
* 程式大約花費16hr完成
* 程式架構概觀



會區分以下區塊解釋

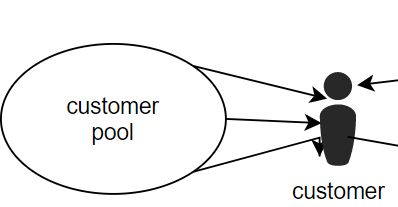


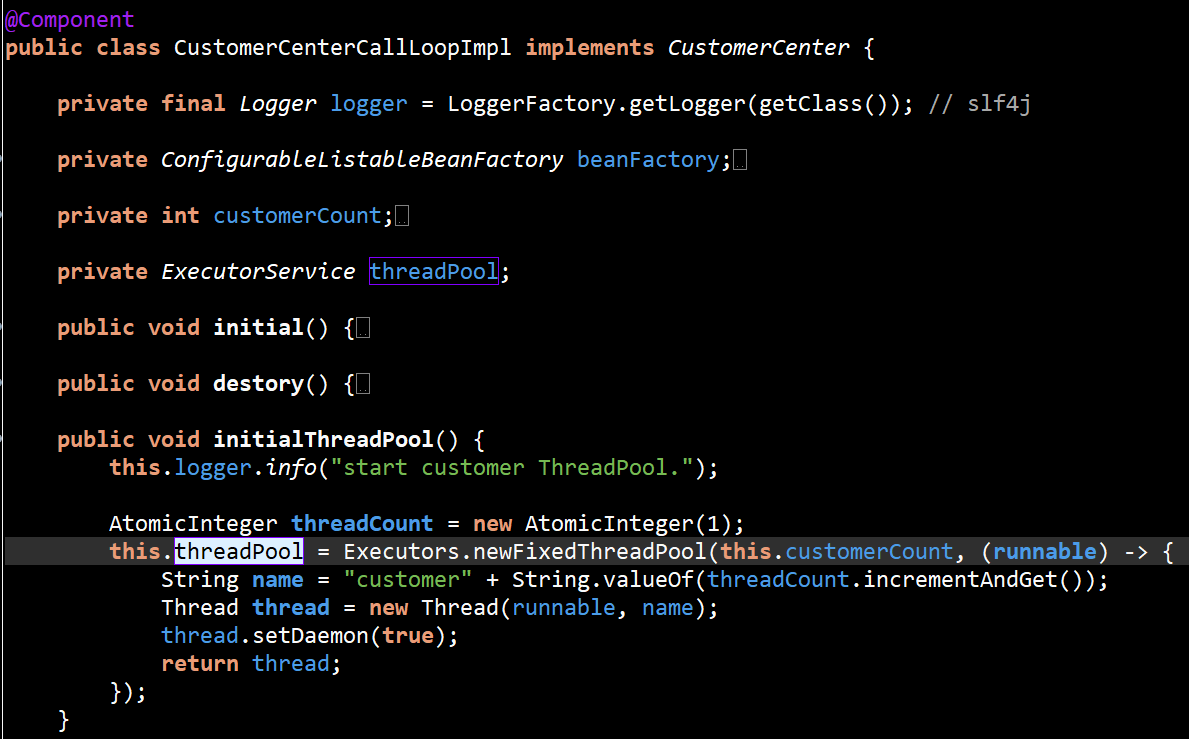
* **採用spring boot實作, 因為任何情境我都認為應該透過IOC的精神完成邏輯, 所以我採用spring**



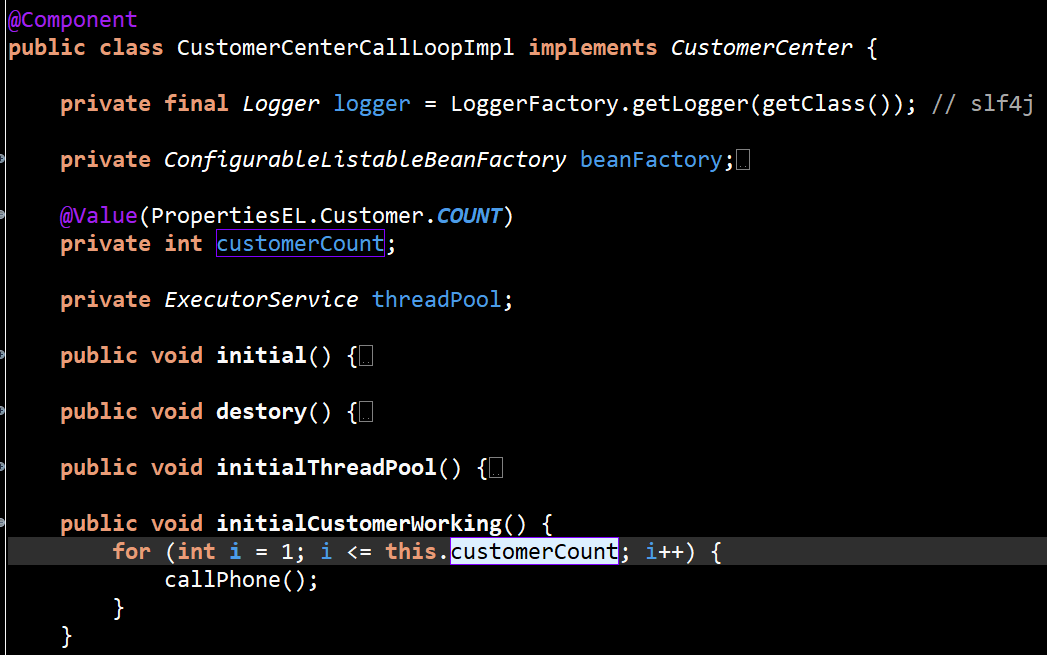
* **[A] Customer pool**

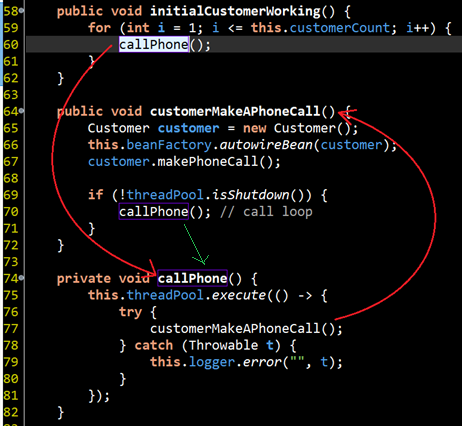
用來模擬同時會有多少customer打電話, 將透過Executors建立thread pool





接著根據設定值決定要有多少個customer打電話

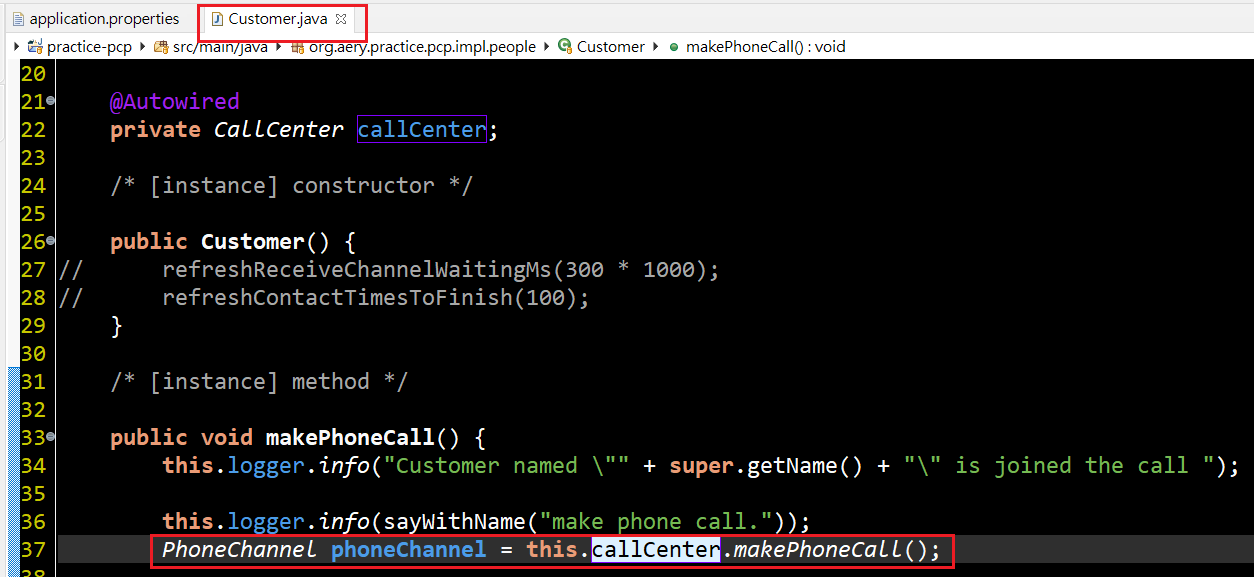




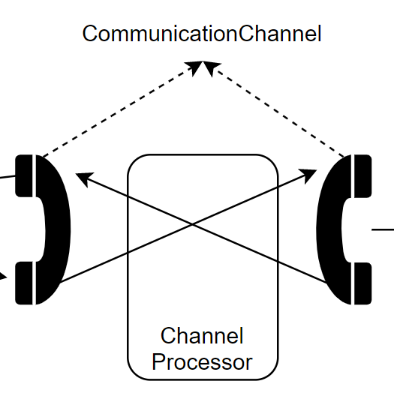
綠色箭頭處則表示customer在#67打完電話後, 再重新讓一個新的customer打電話,

就會形成固定數目的customer不斷撥打電話!

**Customer**物件會透過**CallCenter**取得一個channel, 也就是跟[B]溝通的管道

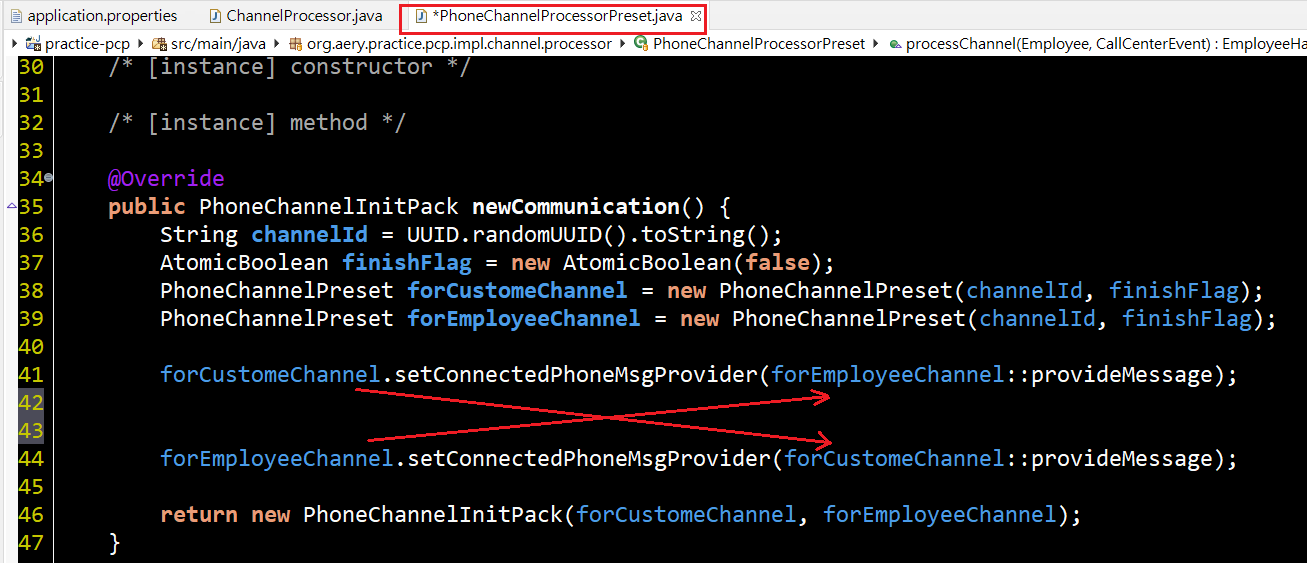


* **[B] CommunicationChannel 與 ChannelProcessor**



由於考量**CallCenter**會有多種溝通管道, 因此拉了一個*CommunicationChannel*介面出來隔離實作, 並只先建立了*PhoneChannelPreset*的實作, 並由*ChannelProcessor*介面(Factory DP)的實作建立對應的*CommunicationChannel*實作, 因此未來要多別的溝通管道實作(如email), 則只需要實作新的*CommunicationChannel*及*ChannelProcessor*實作即可!

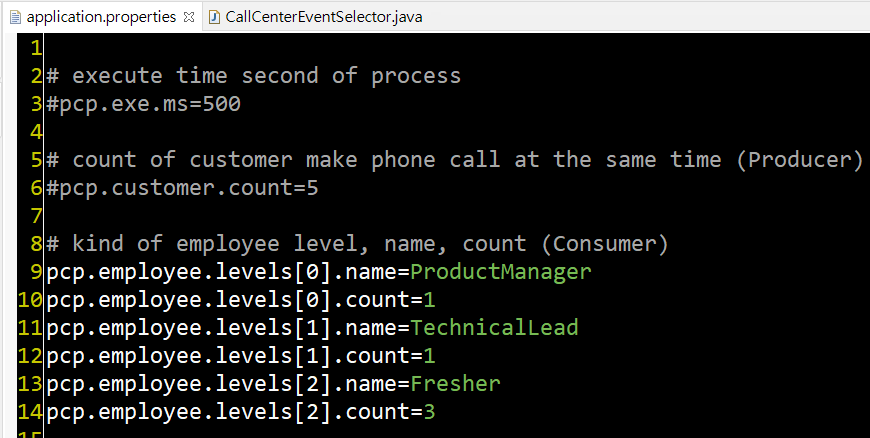
另外*PhoneChannelPreset*的實作內容模擬了通話過程”聽(等待)>反應>說”的過程, 因此會操作object的wait與notify來處理雙方溝通的數據。(一方講話, 另外一方才會被喚醒或等待時間到了)

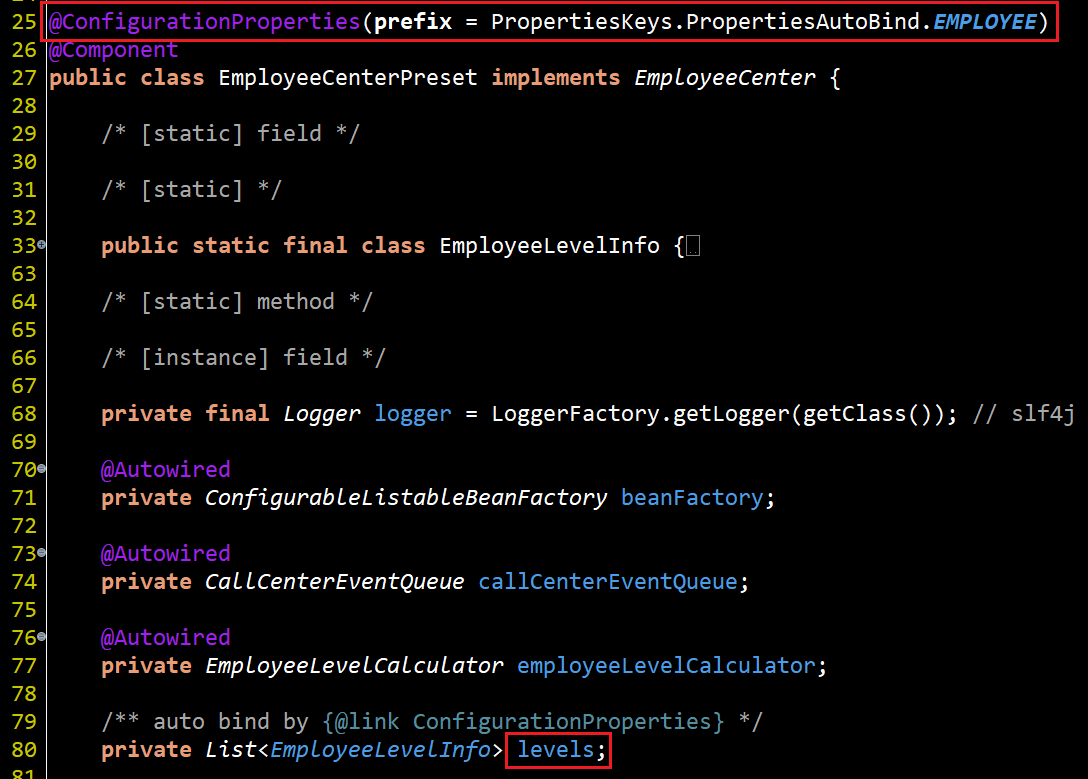




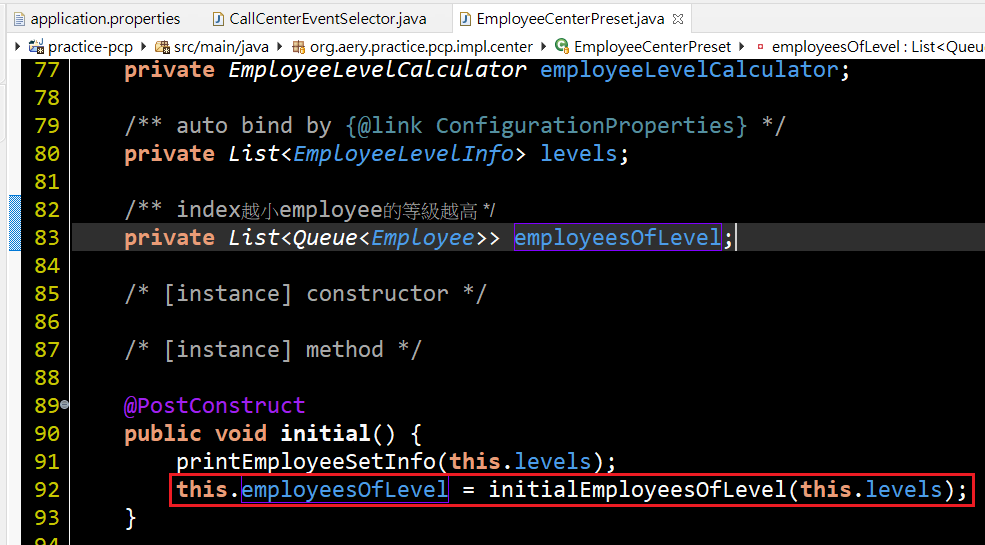
* **[C] employee pool**

為了employee level擴充與人數調整, 我將其設計為參數化行為



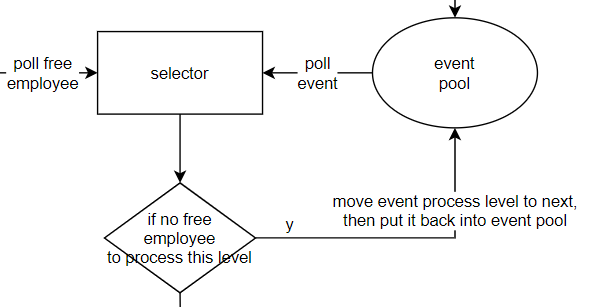
並透過@ConfigurationProperties自動綁定參數至field上

接著會整理成employeesOfLevel, 用來存放哪個level有多少employee



利用這樣的方式就能夠動態的調整employee的level, title, count.

* **[D] selector**



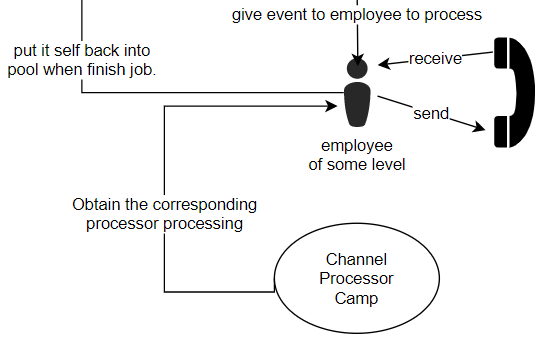
Selector負責了核心的工作:

1. 從event pool(queue)取出event
2. 根據event內的processingLevel決定要從employee pool(queue)取出哪個level的employee處理.
3. 若從event pool內取不到employee, 則表示當前level沒人處理, 則操作event. toNextLevel ()移動給下一個level的employee, 並放回event pool.
4. 若從event pool有取出employee, 則將該event交給該employee處理(other thread)

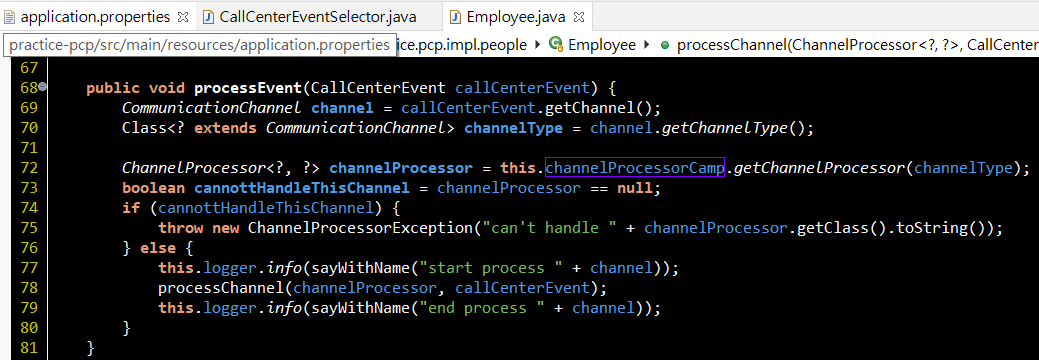


PS:由於只有一個thread會處理這個method, 因此select取出event之後的動作越少越快越好, 才能盡快處理下一個event!

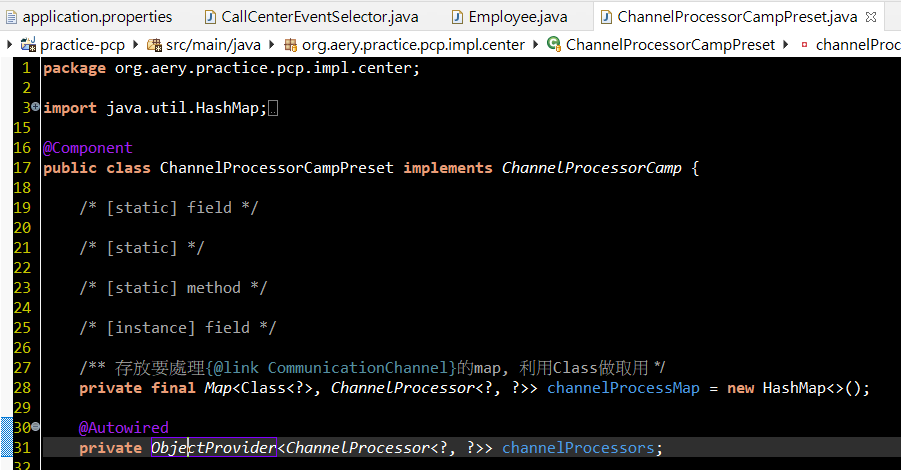
* **[E] employee process event(channel)**



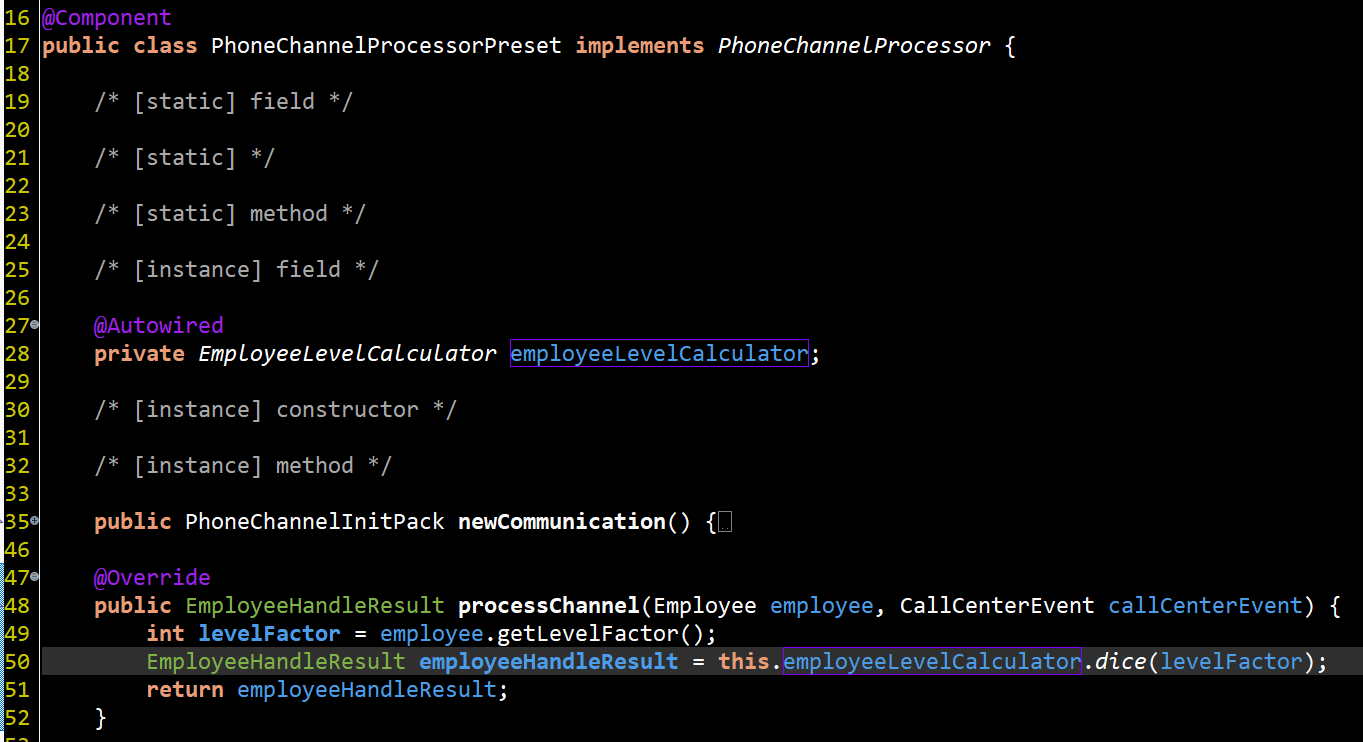
因為*CommunicationChannel*實作有N種, 所以empolyee就不能寫死每種*CommunicationChannel*的處理方式, 因此會透過*ChannelProcessorCamp*取得對應的*ChannelProcessor*來處理.



*ChannelProcessorCamp*則透過spring的*ObjectProvider*幫忙蒐集所有實作*ChannelProcessor*介面的bean



而目前*ChannelProcessor*僅有的實作為PhoneChannelProcessorPreset則負責了擲骰子的工作



當empolyee處理完event, 會將他自己放回employeeQueue, 表示自己將可以再被Selector取出指派下一個工作!

