LE, Nhat Hung - 260793376 2018.03.05

COMP 303 - Assignment 4 - Question 1 Part B

The parts unfit for class diagrams are: punching in, widget production history generation, and punching out. First, when punching in, an employee can optionally set widgetCount in their would-be-instantiated PunchTransaction. This choice can be given by through UI (user input) in Menu.punchIn(), and can be represented in an activity diagram. Second, when generating the widget production history, the manager can also specify a time range for the report. Furthermore, by design choice, Database.generateWidgetProdHis() creates this report by reading an external file which holds all punch transactions in pretty print, sorting them by employee while respecting the time range. The report generation process can thus also be represented by an activity diagram. Finally, moving on to our last part: punching out. I mentioned an external file housing all punch transactions in pretty print. This file is populated each time Menu.punchOut() is called, through a series of other method calls. By choice, punchOut() first validates an employee's ID (Database.isEmployee()), verifies the corresponding Employee object is already punched in (Employee.getIsPunchedIn()), adds the end time to Employee's CurPunchTransaction, write its pretty print (PunchTransaction.getPunchTransaction()) into the external file, and set CurPunchTransaction to null. Those last 3 steps happen in Employee.endCurPunchTransaction(). This isn't a process easily represented by a class diagram. A sequence diagram would be more fitting. In summary, punching in and generating manager's report can be represented by activity diagrams, and a sequence diagram can represent punching out.