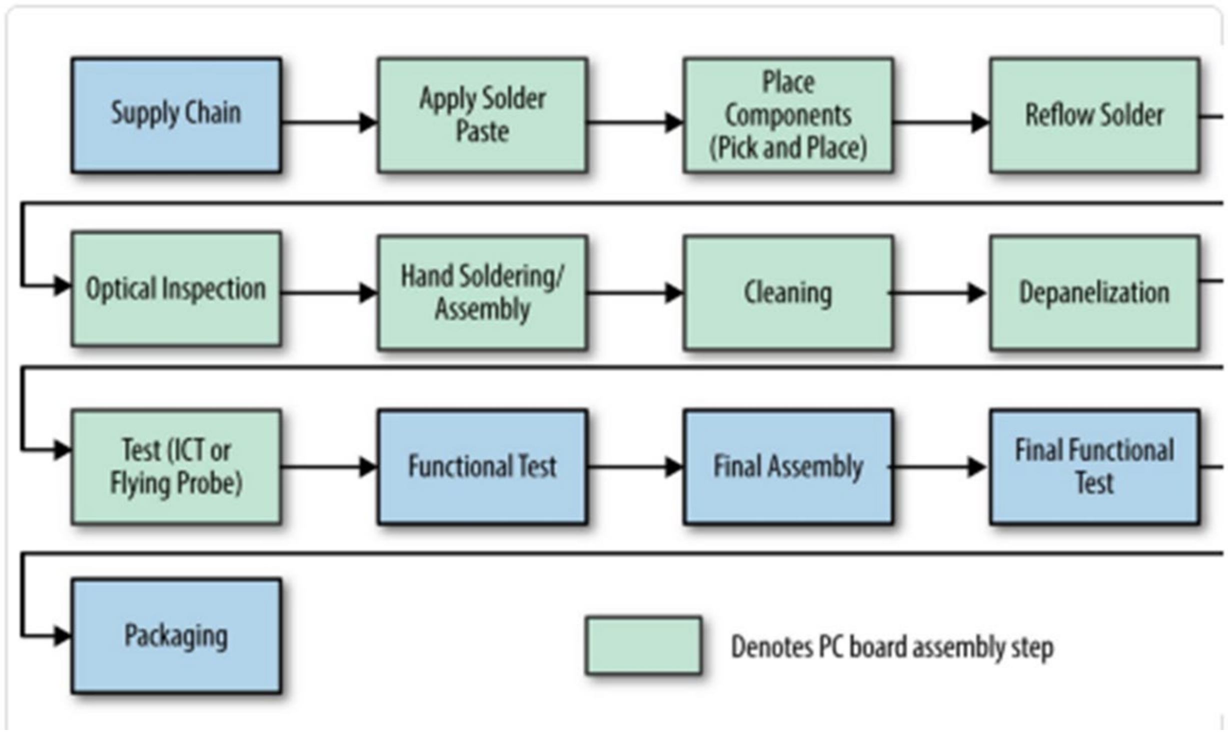
Project 2: OO Design and Code Exercise: Dependency Injection

This a 15 point project, graded by peer review only, based on your PDF submission described below.

Problem Description

This is a typical flow for the creation of a printed circuit (PC) board based product.



(Image from Prototype to Product, Cohen, 2015, O'Reilly)

In project 1, you were asked to model a simulation of the PC board assembly above. You are now being asked to create a Java code solution that performs the simulation. The same description of the system applies to this design and development effort.

Simulate the PC board assembly steps (in green) in the process above. In this model, PCBs (printed circuit boards) arrive at the first station, named above as Apply Solder Paste. The final station in the board assembly is Test (ICT or Flying Probe) (ICT = In-Circuit Test). There are several stations that have a chance of detecting a defect and declaring a failure for a PCB. The stations where there will be a normal percentage chance of rejecting a PCB for a defect are:

- Place Components
- Optical Inspection
- Hand Soldering/Assembly
- Test (ICT or Flying Probe)

A given PCB board type will have unique failure percentage likelihoods for each of these four stations. In addition, each of the eight assembly stations has a unique normal percentage chance of a station failure. If the station fails, the PCB in process at that point will be discarded as if it had a defect.

Your simulation should run 1000 PCBs in each simulation run. At each station that has a PCB defect chance and/or a station failure chance, you should randomly check whether the board or station has failed or not. If either type of failure occurs, you should remove that board from the production line and keep track of the type of failure that has occurred. Station failures should be checked prior to PCB defect checks.

At runtime for the simulation, you will use dependency injection to set the type of PCB used in a simulation run. There will be three types of PCB boards for assembly to begin with: a test board, a sensor board, and a gateway board. Each PCB board type has its own set of unique chances of being rejected at each of the four PCB process defect detection stations. The chance of a station failure is the same for all boards and all stations = 0.2 % (i.e. 99.8% of the time, a given station will function correctly).

Failure chances by PCB board type

Failure Type/PCB Type	Test Board	Sensor Board	Gateway Board
Place Components	5%	0.2%	0.4%
Optical Inspection	10%	0.2%	0.4%
Hand Solder/Assembly	5%	0.4%	0.8%
Test (ICT or Flying Probe)	10%	0.4%	0.8%

At the end of the simulation you should be able to provide for the simulation run:

- Type of PCB
- Number of PCBs submitted for assembly
- Number of failures (PCBs discarded) at each of the four stations due to detected PCB defects
- Number of failures (PCBs discarded) at each station due to a station failure
- Number of PCBs that pass the final test station (completed PCB assemblies)

Project Deliverables

Create a UML Class Diagram for the simulation using a UML tool of your choice. Your OO design for this simulation should **include at least one OO Design Pattern** application. Your UML should show clearly where the pattern is applied and where dependency injection is used.

Implement the simulated production line run using Java. Your code should include the implementation of the OO Design Pattern selected in your UML diagram. The pattern code should be clearly identified in code comments.

You will also need to **implement a dependency injection (DI) approach**. You may select one of these three DI approaches:

- The pseudo dependency injection discussed in class as "A Plain Java Example" (Example at https://www.codejava.net/coding/what-is-dependency-injection-with-java-code-example)

- Google Guice (see https://www.baeldung.com/guice for tutorial information).
- Spring-based DI (Example at https://www.geeksforgeeks.org/spring-dependency-injection-with-example/)

At runtime, your code should use the PCB type set in the DI metadata (for Spring and Guice) or in code (for Plain Java). **You need to run your application against all three PCB board types, and capture the resulting failure report** in a text-based table similar to the following output:

```
PCB type:               Sensor Board
PCBs run:               1000

Station Failures
Apply Solder Paste:           2
Place Components:             3
Reflow Solder:                0
Optical Inspection:           1
Hand Soldering/Assembly:      2
Cleaning:                     1
Depanelization:               3
Test (ICT or Flying Probe):   2

PCB Defect Failures
Place Components              3
Optical Inspection            2
Hand Soldering/Assembly       5
Test (ICT or Flying Probe)    4

Final Results
Total failed PCBs:            28
Total PCBs produced:          972
```

For your own testing, you may want to print messages for pass/fail results for stations or PCBs, but that is not required in final output.

You will not have any code submission required for a Coursera coding lab, **there will be no autograding of this application.** An ungraded coding lab will be provided if you'd like to work on the project development in the Coursera environment, but no example code will be provided. You may find it easier (or necessary) to implement your code in a Java environment local to your own personal computer where you can decide on an IDE and install any supplemental libraries or tools you may need.

Deliverables for project should be submitted as a PDF with each section clearly labeled. Your PDF submission for peer review should include:

1) Full UML Class Diagram for this model of the simulation, including clear identification and annotation of the type of Dependency Injection being implemented and the OO Design Pattern(s) that are part of this design. Your UML Diagram does not have to show accessability or multiplicity, but it should clearly show key methods, references, and

inheritance between classes and subclasses. You can also include any design assumptions or interpretations here.

2) Your Java code, developed in an OO fashion, well commented with particular annotation of OO Design Pattern(s) and the Dependency Injection elements
3) The results of three simulation runs for the three types of PCBs (test, sensor, gateway)

It may be easiest to combine your annotated UML diagram, code, and output in a Word document to then export or print a single PDF containing all deliverables. Please clearly label all three sections for your peer reviewers to examine.

Design/Development Interpretation

Unlike the prior course assignments, Course 3 projects are intended for you to explore OO design and related tools and technologies, exercising your own research, design, and development skills. This mirrors previous on-campus projects in OOAD that require students to develop solutions from scratch. The projects intentionally do not contain starting or example code and will require your work to develop complete solutions.

You may find that some content for project deliverables may be unclear or incomplete, as requirements for software often are. If you find a question in your assessment of the project description or deliverables, you should document any assumption or interpretation you make and use your best effort to meet the spirit of the project content as presented. Please include any discussion of your assumptions regarding the project deliverables in your submitted project PDF.

Grading Rubric

After reviewing your submitted PDF, peer reviewers will assign a grade of up to 15 points for your work, as follows:

UML Class Diagram

5 points – Thorough and complete work on UML class diagram

4 points – Minor issues in UML diagram

3 points - Major issues or obvious missing elements in the UML diagram

0 points – No UML Class Diagram submission

Java Code

5 points – Thorough and complete work on Java code, clear annotation comments identifying the DI approach in use and at least one OO Design Pattern

4 points – Minor issues in code or missing comments for implemented DI approach or Design Pattern(s)

3 points – Missing implementation of DI approach OR Design Pattern(s)

2 points - Missing implementation of DI approach AND Design Pattern(s)

0 points – No Java Code submission

Run Results

5 points – Thorough and complete output reports for each of three simulated PCB runs of 1000 boards, clearly identified for test board, sensor board, and gateway board.

4 points – Minor issues in reports of three PCB simulation runs

3 points – Missing one or more runs OR major issues in supplied output

0 points – No output supplied