Dependency Injection

APPLICATION	HIGH LEVEL CLASS	LOW LEVEL CLASS

main() {
   …
   HLC h = new HLC();
   h.foo();
   …
}
	class HLC {
	
	   void foo() {
	      …
	      LLC l = new LLC();
	      …
	      l.doSomething();
	      …
	   }
	}
		class LLC {
		
		   void doSomething() {
		      …
		   }
		}

DIAGRAM-1

Q1. What is a MAJOR problem with the HLC class for

a) Maintainers?

HLC needs the concrete class LLC. This is akin to "hardcoding CONSTANTS". If there is a need to change the concrete class to something else (say LLC2), the maintainer will need to modify the HLC code, recompile it, and retest it.

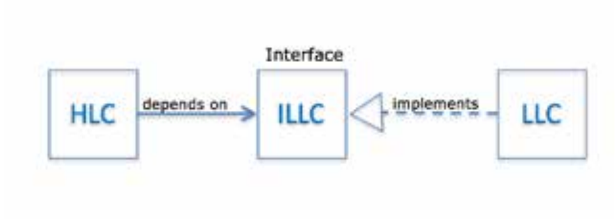
Say LLC represents mysql database. Anytime HLC needs to refer to a DIFFERENT database (say sqlite), its code will need to be changed. Instead, it will be much better if the database change is made only in a configuration text file – without HLC code needing to be changed at all.

b) Testers?

In the current situation, testing of HLC with LLC stubbed or mocked cannot be done without changing the source code for HLC. In general, testing of code (say A) should NOT need the tester to change the source code of A!

It is important to be able to mock/stub dependent classes so that development and testing of a unit can proceed without having to wait for availability of the dependent classes. In addition, it is also advisable to mock dependent classes to reduce the COMPILE/RUN/DEBUG cycle times.

Q2. Draw diagram of solution approach. What's the idea?



The idea is to remove the direct dependency between HLC and LLC by introducing an interface. Now, HLC depends on the interface and not on LLC.

Q3. Fix the problem in Diagram-1. Describe each step!

Step 1: Make an interface for the LLC

```
interface ILLC {
  void doSomething();
}
```

Step 2. Make LLC implement interface

```
class LLC implements ILLC {
   void doSomething() { ..... }
}
```

Step 3: Make HLC use the Interface. There should be ZERO references to LLC in HLC code. LLC can be injected either via constructor OR via a setter method. In the example, we used the constructor.

```
class HLC {
  ILLC l;
  HLC (ILLC l) { this.l  = l; }

  void foo () {
     l.doSomething();
  }
}
```

Step 4: In application code, create a new LLC object and INJECT into HLC code (in this case using constructor of HLC).

```
main() {
  ...
  HLC h = new HLC (new LLC());
  h.foo();
  ...
}
```

Q4. Explain how the maintainers and testers work are facilitated by these changes!

You should be able to do this now!