Alexander Martin

Dr. King

CS 3354 – 255

Dependency Injection

One of the greatest struggles I encountered in Objected Oriented Programming was the ability to test my programs while implementing 3 or more classes. I found it difficult to test each class individually without relying on the other classes working to the specification of the assignment. Early in the course we discussed design patterns such as tightly and loosely coupled class, which when implemented made it much easier to test. Dependency Injection is one step further and helps overcome the need for class coupling at all.

The rules for working with OOP are essentially, your highest level of classes shouldn’t rely on the lower level classes and they should be connected through abstraction or interfaces. In short Dependency injection is the implementation of the Inversion Principle. With the help of Dependency Injection objects depended on specific classes can be outside of any given class that may depend on a higher class than its own.

During my research I came across Miguel Castro who describes in detail how to overcome coupling with the use of Dependency Injection Containers. He describes the container as a “tool that turns DI into architectural patterns that lets you satisfy a types dependencies … it is a repository that typically associates interfaces with concrete types.” In practice these containers will act as placeholders for the instance of the concrete type within the called class. This is what is called Resolving. This can be done as many times as necessary. This is especially useful when working on the larger programs done in this class, as it doesn’t require the classes to need other lower classes for compilation, by using faux dependencies it allows the user to test a number of classes individually.

At times larger projects may call for implementing a Dependency Injection framework, which Miguel Castro says can “making troubleshooting difficult and time consuming”. Although the advantages of an automatic framework greatly outweigh any disadvantages. It can help keep a project consistent especially when working in a larger group of collaborators, and the ability to implement faux dependencies can greatly speed up the work load, allowing key components in the application to be tested.

Overall, dependency injection can help save so much time and effort with how easy it makes testability and routine maintenance, whether you’re using the automatic frameworks such as Pico container, or are manually doing it.

Work Cited

* https://www.codemag.com/Article/1210031/Understanding-Dependency-Injection-and-Those-Pesky-Containers