1. What advantages are of the 3 layered approach to building applications.

The three tiered approach to building applications is relatively simple. There are three layers, somewhat like the layers of a cake, that each take a different job in the application. The first layer is responsible for the U.I. of the application and is known as the presentation layer (IBM. 2018). The second tier is known as the business or object layer, this is needed to manage the objects or business logic of the application and can access the third tier. The presentation layer only has access to the methods and attributes declared as public in this layer. Finally, the third layer is the data layer. This layer accesses the data stores by the application, for example a database. It allows the other layers to be configured without directly having access to the database libraries (Microsoft. 2012). The main advantage of using the three layered architecture is maintainability, this means that changes in one layer have no effect on other layers. This means that if the presentation layer has a problem it can be fixed without having to re-code any of the other layers. A second advantage is reliability. For example, if the database supporting the application is split over multiple servers, multiple levels of redundancy can be added to the data layer in case one of the servers fails, allowing the application to keep running with minimal interruption. This backs up the first point as if that were to happen only the data layer would need to be dealt with as none of the other layers can access the database (Tony Marston. 2012). A final advantage is that a three tier architecture makes collaboration very simple as each layer can only access other layers by the methods and properties defined in that layer. This means that they can be worked on by different people entirely as they don’t need to know the inner workings of each layer. Overall the main advantage to using this architecture is its reliability, which stems from the fact that all the layers are isolated from each other so if there is a problem with one it doesn’t affect any other layers and can be fixed without having to look at other layers.

References

IBM. (2018). Three-tier architectures. Retrieved From: <https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/covr_3-tier.html> (Accessed on 07/12/2018)

Microsoft. (2018). Using a Three-Tier Architecture Model. Retrieved From: <https://docs.microsoft.com/en-us/windows/desktop/cossdk/using-a-three-tier-architecture-model> (Accessed on 07/12/2018)

Tony Marston. (2012). What is the 3-Tier Architecture?. Tony Marston’s Blog. Retrieved From:

<https://www.tonymarston.net/php-mysql/3-tier-architecture.html> (Accessed on 07/12/2018)

1. With an example, explain why using design patterns can make the design of an oo system easier to understand.

A design pattern is a framework to help programmers design how an applications classes interact with each other. They can be categorized in three groups, creational, structural and behavioral (Dofactory. 2018). One example of a design pattern is singleton, which is an example of a creational pattern. In order to adhere to this pattern a class can only have one instance and have a global point of access to it (Source Making. 2012). Singleton makes an object oriented system easy to understand by only instantiating the class once. This means that code can be followed easily as all the references to the class will be referencing the same object. This also means that the object can only be accessed by its pre-defined methods meaning that it is simple to use. This helps the system overall to be easily understood by reducing the number of objects created allowing a programmer to follow each one through the code without getting lost or being confused as to which object is being dealt at a time. In the example the class is instantiated twice (Figure 2), by using the pre-defined method to instantiate the class (Figure 1). However, both classes are identical, showing that the class follows the pattern as only one version of the class can be created. Overall singleton makes an object oriented system easy to understand by making sure that the class can only be instantiated once meaning that all references to that object can be followed easily.

Diagram

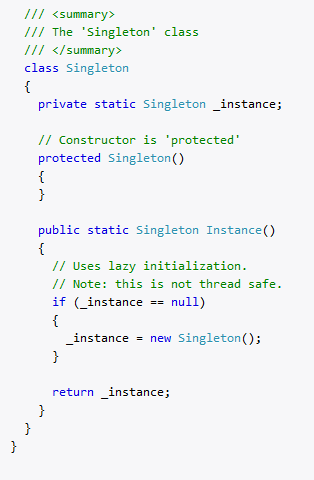


Figure 1, an example singleton class (Dofactory. 2018)

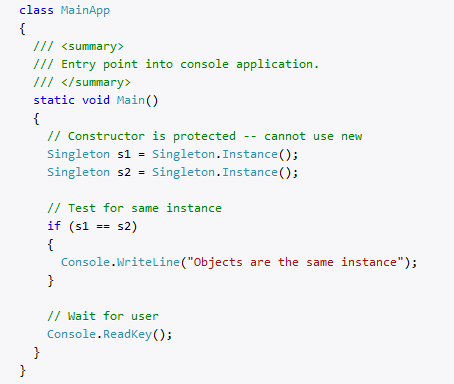


Figure 2, the singleton class being put to use in a main method. The output will say ‘Objects are the same instance’. (Dofactory. 2018)

References

Dofactory. (2018). .Net Design Patterns. Retrieved From:

<https://www.dofactory.com/net/design-patterns> (Accessed on 07/12/2018).

Source Making. (2018). Singleton Design Pattern. Retrieved From:

<https://sourcemaking.com/design_patterns/singleton> (Accessed on 07/12/2018).

DoFactory. (2018). Singleton. (Used for the example). Retrieved From:

<https://www.dofactory.com/net/singleton-design-pattern> (Accessed on 07/12/2018).