UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

NAME: **NKURAIJA GARY MATTHEW**

REGISTRATION NO.: **S19B23/537**

ACCESS NO.: **A87132**

COURSE: **Bachelor of Science in Computer Science**

COURSE UNIT: **SOFTWARE CONSTRUCTION**

LECTURE: **Mr. Simon Lubambo**

QUESTION: **Write short notes on Facade Design Patterns**

A structural design pattern called Facade offers a streamlined user interface to a complicated system of classes, making it simpler to use. When a collection of classes, such a library or framework, have a complicated interaction or relationship, the Facade design is frequently used to give a higher-level interface.

The Facade approach is intended to provide an easier access to the underlying functionality while encapsulating a group of classes. When a complicated system of classes must be employed but the developer does not want to reveal every aspect of the system to the client, this can be helpful. The Facade pattern can simplify the user interface, making the system simpler to comprehend and operate.

Often, the Facade approach is implemented as a single class that stands between the client and the intricate class hierarchy. While leaving the intricate interactions to the underlying system of classes, the Facade class gives the client a straightforward interface. The interaction is made easier and more logical for the client by using the Facade class rather than the underlying classes directly.

There are numerous circumstances where the Facade pattern can be helpful. For instance, it can be used to make it easier to utilize a library or framework that has a difficult-to-navigate interface. The Facade approach can minimize the learning curve for new users and make it simpler to use the library or framework by offering a streamlined interface.

The Facade design can also be used to group together a number of interconnected classes with intricate interactions. When the interplay between the classes is intricate and the client doesn't need to be aware of every nuance, this can be helpful. The Facade approach can lessen the likelihood of errors and problems in the client code by offering a streamlined interface that makes it simpler to use the collection of classes.

A single point of access to a group of related classes can be offered through the facade pattern. This can be helpful when the client wants to access several classes to complete a certain activity. The Facade pattern can streamline communication between the client and the collection of classes and lessen the complexity of the client code by offering a single point of access.

The Facade approach may have the drawback of adding an additional layer of abstraction to the system, which could make it more challenging to comprehend and debug. Also, if the Facade class is poorly built, it may create a performance bottleneck for the system.

The developer should determine the group of related classes that need to be enclosed and build a client-facing interface that is as simple as possible before implementing the facade pattern. The Facade class should be in charge of handing off complicated interactions to the underlying system of classes and giving the client a cleaner interface. In order to prevent the Facade class from turning into a system bottleneck, it must also be effectively planned and implemented.

Ultimately, the Facade design pattern is a practical technique for streamlining client-complex class system communication. The Facade approach can lessen the possibility of mistakes and faults in the client code by offering a streamlined interface that is simpler to comprehend and use. To prevent a system bottleneck and to make sure that the abstraction layer does not grow overly complex, the developer must take care when designing the Facade class.