Question 4

Design patterns used

Singleton design pattern

Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under Creational pattern as this pattern provides one of the best ways to create an object. This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

Why we use it?

As we know that singleton pattern has very benefit on common objects that can be used in each section of the system. If we look at Database Connection class, singleton pattern can be very suitable for this. Because we can manage Db Connection via one instance so we can control load balance, unnecessary connection, shared db connection management, control data inconsistency etc.

Prototype design pattern

Prototype pattern refers to creating duplicate object while keeping performance in mind. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object. This pattern involves implementing a prototype interface which tells to create a clone of the current object. This pattern is used when creation of object directly is costly. For example, an object is to be created after a costly database operation. We can cache the object, returns its clone on next request and update the database as and when needed thus reducing database calls.

Why we use it?

Let's consider the case in which we need to make a number of queries to an extensive database to construct an answer. Once we have this answer as a table or Result set, we might want to manipulate it to produce other answers without having to issue additional queries. As such reason After we fetch the data from database to be sorted in different context we don’t need to fetch the same data for those tasks we just fetch the data for the first time then by cloning it we can manipulate our task as we need it.

State design pattern

State design pattern is used when an object change it’s behavior based on it’s internal state, encapsulates varying behavior for the same object, based on its internal state. If we have to change the behavior of an object based on its state, we can have a state variable in the object and use if-else condition block to perform different actions based on the state.

Why we use it?

In our project we used state pattern in order to fetch grade information of the student from different tables and allow us to access the classes corresponding to his/her grade level.

It allow us to get the result information for the different grade levels by passing the grade value to the a class

Iterator design pattern

An aggregate object such as a list should give you a way to access its elements without exposing its internal structure. Moreover, you might want to traverse the list in different ways, depending on what you need to accomplish.

Why we use it?

We have many data in a database, that needs to be fetched and these data might come in group of objects making it necessary to use iterators.