Design Patterns Description

Class name : Scheduler Notifier

Design Pattern : Singleton

Problem :

The system has Auctions everyday and every auction has many sessions , the system has to manage the notification that will be sent to Bidders and Sellers for specific session(s) , the announcement will be sent for (the winner of the session) and the invoice for the (winner) bidder and the seller, and the notifications will be sent for (start the session,end the session,bids in the session) , the Problem that all notifications and announcements depend on the time of the system(so it has to be unified)

Solution :
Using singleton Design pattern will guarantee that only one unified time in the system , that can't be changed .

Anti-Design Patterns :
using inheritance will avoid Unified time in the whole system

Class name : DBInterface "DataBase"

Design Pattern : Singleton

Problem :

DataBase overhead may lead to low Performance , inconsistency , keep opening connections without manage results overhead , the problem that we want only one connection to the database exists

Solution :

Using singleton Design pattern will guarantee that only one Connection to the database to (store,retrieve ,delete ,update) data will exist.

Anti-Design Patterns :

using inheritance will allow to open more than one connection while will result overhead and may occur fail in the data.

Class name : Data Manager

Design Pattern : Delegator

Problem :
Most of the classes deal with the database (inserting , deleting ,updating ,or retrieving) data. The implementation of this methods changes from one class to another ,but it’s the same operations
The problem we want to reuse the methods without Duplication the code.

Solution :
Using Delegator will allow us to use the methods from this class and change the implementation to be suitable for its need.

Anti-Design Patterns :
using inheritance will allow duplication of code .