**Practical 4**

**Objective:** Identify the design principle that is being violated in relation to thegiven scenario.

**Note:** A good object oriented design not only meets the specified requirements butalso addresses implicit requirements. There are five design principles which address most of the implicit requirements:

**1.Abstraction:** Focus on solving a problem by considering the relevant detailsand ignoring the irrelevant.

**2.Encapsulation** ​Wrapping the internal details, thereby making these detailsinaccessible. Encapsulation separates interface and implementation, specifying only the public interface to the clients, hiding the details of Implementation.

**3.Decomposition and Modularization:** ​Dividing the problem into

smaller, independent, interactive subtasks for placing different functionalities in different components.

**4.Coupling & Cohesion:** Coupling​is the degree to which modules aredependent on each other. Cohesion is the degree to which a module has a single, well defined task or responsibility. A good design is one with loose coupling and strong cohesion.

**5.Sufficiency, Completeness and Primitiveness:** ​Design should



ensure the completeness and sufficiency with respect to the given specifications in a very simple way as possible.

**Problem:** Which of the following design principle(s) have been violated in thefollowing scenarios?

1. Abstraction
2. Decomposition and Modularization
3. Coupling & Cohesion
4. Encapsulation
5. Sufficiency, Completeness and Primitiveness
6. All

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Principle Violated** |
|  |  |  |
| 1 | Important information of a module is directly |  |
|  | accessible by other modules | Encapsulation |
|  |  |  |
| 2 | Too many global variables in the program after |  |
|  | implementing the design | Coupling & Cohesion |
|  |  |  |
| 3 | Code breaks in unexpected places | All |
|  |  |  |
| 4 | Unfulfilled requirements in the code after the design |  |
|  | has been implemented | All |
|  |  |  |
| 5 | Cyclic dependency among classes |  |
|  |  | Decomposition and Modularization,Coupling & Cohesion |
| 6 | Huge class doing too many unrelated | Abstraction , Decomposition and Modularization |
|  | operations |  |
|  |  |  |
| 7 | Several un-related functionalities/tasks are carried |  |
|  | out by a single module | Decomposition and Modularization |
|  |  |  |
| 8 | All data of all classes in public | Encapsulation |
|  |  |  |
| 9 | Design resulting in spaghetti code | Suffieciency , Completeness and Primitiveness |
|  |  |  |
| 10 | An algorithm documented as part of design is not |  |
|  | understandable by the programmers | Sufficiency , Completeness and Primitiveness |
|  |  |  |