PART - A

1. What is GRASP?

General Responsibility Assignment Software Patterns (or Principles), abbreviated GRASP, consists of guidelines for assigning responsibility to classes and objects in object-oriented design.

2. What is Responsibility-Driven Design?

A popular way of thinking about the design of software objects and also larger scale Components 3 are in terms of responsibilities, roles, and collaborations. This is part of a larger approach called responsibility-driven design or RDD.

3. What is Responsibilities?

The UML defines a responsibility as “a contract or obligation of a classifier”. Responsibilities are related to the obligations or behavior of an object in terms of its role.

4. What are the two responsibilities?

The responsibilities are of the following two types: doing and knowing.

Doing responsibilities of an object include:

• doing something itself, such as creating an object or doing a calculation

• initiating action in other objects

• controlling and coordinating activities in other objects

Knowing responsibilities of an object include:

• knowing about private encapsulated data

• knowing about related objects

• knowing about things it can derive or calculate

5. Define Pattern?

A pattern is a named problem/solution pair that can be applied in new context, with advice on how to apply it in novel situations and discussion of its trade-offs

6. What are the GRASP patterns?

They describe fundamental principles of object design and responsibility assignment. expressed as patterns.

7. How to Apply the GRASP Patterns?

The following sections present the first five GRASP patterns:

. Information Expert

. Creator

. High Cohesion

. Low Coupling

. Controller

8. Define Creator?

Creation of objects is one of the most common activities in an object-oriented system. Which class is responsible for creating objects is a fundamental property of the relationship between objects of particular classes.

9. What is Controller?

The Controller pattern assigns the responsibility of dealing with system events to a non-UI class that represent the overall system or a use case scenario. A Controller object is a non-user interface object responsible for receiving or handling a system event.

10. Define Low Coupling?

Low Coupling is an evaluative pattern, which dictates how to assign responsibilities to support:

• low dependency between classes;

• low impact in a class of changes in other classes;

• high reuse potential;

11. Define High Cohesion?

High Cohesion is an evaluative pattern that attempts to keep objects appropriately focused, manageable and understandable. High cohesion is generally used in support of Low Coupling. High cohesion means that the responsibilities of a given element are strongly related and highly focused. Breaking programs into classes and subsystems is an example of activities that increase the cohesive properties of a system.

12. What is Information Expert?

Information Expert is a principle used to determine where to delegate responsibilities. These responsibilities include methods, computed fields and so on. Using the principle of Information Expert a general approach to assigning responsibilities is to look at a given responsibility, determine the information needed to fulfill it, and then determine where that information is stored. Information Expert will lead to placing the responsibility on the class with the most information required to fulfill it

13. What is singleton pattern?

The singleton pattern is a design pattern used to implement the mathematical concept of a singleton, by restricting the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system.

14. What is adapter pattern?

The adapter pattern is a design pattern that translates one interface for a class into a compatible interface. An adapter allows classes to work together that normally could not because of incompatible interfaces, by providing its interface to clients while using the original interface. The adapter is also responsible for transforming data into appropriate forms.

15. What is Facade Pattern?

A facade is an object that provides a simplified interface to a larger body of code, such as a class library. A facade can:

• make a software library easier to use, understand and test, since the facade has convenient methods for common tasks;

• make code that uses the library more readable, for the same reason;

• reduce dependencies of outside code on the inner workings of a library, since most code uses the facade, thus allowing more flexibility in developing the system;

• Wrap a poorly-designed collection of APIs with a single well-designed API (as per task needs).

16. What is Observer pattern?

The observer pattern (a subset of the publish/subscribe pattern) is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. It is mainly used to implement distributed event handling systems.

Part –B (16 Marks)

1. Explain GRASP: Designing objects with responsibilities

2. Explain GoF DESIGN PATTERNS

3. Explain Creator and Information Expert with an Example?

4. Explain Low Coupling and Controller with an Example?

5. Explain adapter and singleton with an example?

6. Explain factory and observer patterns.