**1. What is a design pattern?**

A design pattern describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context. An example of "best practice" in OO design. A description of a common solution to a common problem Design patterns are a "vocabulary" for designers to better communicate design ideas.

Can be characterized as a three-part rule which

Expresses a relation between a certain **context,** A problem; a solution.

Context allows readers to understand the environment in which the problem resides and what solution might be appropriate within that environment

A good design pattern:

 Captures hard-earned pragmatic design knowledge in a way that enables others to reuse that knowledge

 Saves you from “reinventing the wheel” or “inventing a new wheel that is out of shape”

**2. Why do we give them names?**

Pattern name—express the essence of the pattern succinctly. A good name is vital, because it will become part o your design vocabulary.

The names of design patterns should be chosen with care. One of the key technical problems in pattern-based design is the inability to find existing patterns when hundreds or thousands of candidate patterns exist. The search for the “right” pattern is helped immeasurably by a meaningful pattern name.

**3. Describe the following design patterns:** **Observer,** **Iterator,** **Composite,** **Proxy, Facade,** **Strategy. Give an example of the use of each pattern using words and code segments.**

Observer: Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. ( that is, a subscription service, or event notification service.)

Iterator: An Iterator provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation. (LinkedList and ArrayList iterator simplifies traversing through the collection’s elements}

Composite: Composite lets clients treat individual objects and compositions of objects uniformly. (a graphical drawing editor that allows you to build an image out of components, and also allows you to use images you’ve constructed as new components)

Proxy : A proxy is an object used as a substitute or placeholder for an object in order to control access to it. (a firewall proxy that runs on network switch, connecting a company network to the Internet. It filters out client requests and server results that may be inconsistent with company policies)

Facade: A facade provides a uniform interface to a set of interfaces in a subsystem. The Facade defines a higher-level interface that makes the subsystem easier to use. (a PuzzleMaker that creates, initializes, and contains the Puzzle, then provides methods for manipulating the Puzzle. Clients, such as the Driver, doesn’t have to manipulate Puzzle directly, instead uses the methods provided by the PuzzleMaker )

Strategy: A strategy object encapsulates an algorithm, so that the algorithm can vary independently from the clients that use it. (a calculator having one strategy for each arithmetic operation)