Exercise 3 | Gerald Kloimstein | ai22m040

What is the difference between a dynamic linked library and a static library?

static library

- object files are linked with the program at compile time
- code from the library is included directly into the program's executable
- program can run independently without any external dependencies
- any changes made to the library require a recompilation of the program

dynamic link library

- DLL is a separate file that is loaded into the program's memory at runtime
- program links to the library dynamically → library can be updated independently of the program.
- multiple programs can share the same DLL in memory
- individual executable is smaller

Describe a logging system with your own words:

A logging system records events like errors, warnings, system events, user actions, audit information, etc. A logging system shall show where in the source code the error occurred and what the severity of the event is. The information collected by a logging system may be used for error analysis, auditing, monitoring an application's health status, identifying unusual system behaviour, getting information for scaling decisions, etc.

A logging system might also contain features for log compression and policies for rotation and long-term retention.

Describe an event system with your own words (was probably meant):

An event system is a design pattern that allows loose coupling between software components. It deals with the following aspects:

- event sources: any application or software component that generates events and publishes them to the event systems ("fire and forget")
- event objects: data structures that contain information about the event like event type and time stamp
- event listeners: listeners can subscribe themselves with the event system in order to receive all or certain types of events. When an event source publishes an event, the event system delivers it to all subscribed listeners.

Event publishers do not need to know anything about the subscribers who consume their events, and vice versa. Therefore, each software component can be changed independently.

What is a Design Pattern?

A design pattern is a reusable solution to a commonly occurring problem in software design. One can think of it as a set of best practices and techniques that have been developed and

refined by software developers over time. It is like a template for solving a particular problem that can be adapted and reused in different contexts.

Benefits

- reuse of proven solution approaches to common problems – no re-invention of the wheel
- higher productivity in the coding process
- more stable software products
- better code organization – easier to maintain and update

Name and describe 2 Design Patterns (except façade or adapter):

- Singleton pattern: ensures that a class has only one instance and provides a global point of access to that instance. You apply this pattern when you have to ensure that only one instance of a class is created (e.g. database connection or logging service). Typically, you have a private constructor for the class, a private static instance variable, and a public static method that returns the instance.
- Factory pattern: A factory in software design produces class instances like a normal factory produces goods. Instead of instantiating objects of a certain class directly, you delegate this responsibility to a factory method that decides which class to instantiate based on the given criteria. This might be useful when you want to create objects (instances) at runtime given a certain set of conditions or configuration.