Questions and Answers

Tuesday, April 3, 2018 18:52

What is KISS?
Keep it simple, stupid! - You must make you code simple, as simple as possible.
What is DRY or DIE?
Don't repeat yourself OR Duplication Is Evil - If you have piece of code which is repeated several times - you should move it to common file and import it wherever it needed.
What is YAGNI?
You aren't gonna need it! - Think twice when you trying to add any functionality - may be you already have it or you not need it at all.
What is SSOT? (optional)
Single source of truth - data element should be store only in one place and only once.
What is SOLID?
S - single responsibility principle - class should have only one responsibility and because of it we should have only one reason to change this class.
O - open/closed principle - class should be opened for extends and closed for modification.
L - Liskov substitution principle - child class should not change expected behaviour of parent class, methods which use parent class as parameter could use child class without any modifications.
I - Interface segregation principle - better to have lots of simple interfaces splitted by functionality area than one large and complex which contains all methods we have - it allow to avoid of stubing or implementation of useless method.
D - Dependency inversion principle - abstraction should not depend on details but details should depend on abstractions
What is GRASP? (I think this is useful information but I don't think that anybody will be asked to explain it on interview - too large, complex and abstract)
This nine principles and patterns listed by Craig Larman in his book about UML and design patterns.
General Responsibility Assignment Software Patterns

- Information expert
- Controller
- Creator
- Low coupling
- High cohesion
- Indirection
- Polymorphism
- Protected variation
- Pure fabrication

Information expert - Information expert (also expert or the expert principle) is a principle used to determine where to delegate responsibilities.
Controller - A controller object is a non-user interface object responsible for receiving or handling a system event.
Creator - The class must instantiate the classes that it can:
Contain or aggregate;
Write;
Use;
Initialize with the required data.
Low coupling - Coupling is a measure of how strongly one element is connected to, has knowledge of, or relies on other elements.
High cohesion - 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
Indirection - The indirection pattern supports low coupling (and reuse potential) between two elements by assigning the responsibility of mediation between them to an intermediate object.
Polymorphism - this is the provision of a single interface to entities of different types. Responsibility

of defining the variation of behaviors based on type is assigned to the type for which this variation happens

Protected variation - The protected variations pattern protects elements from the variations (modifications) on other elements (objects, systems, subsystems) by wrapping the focus of instability with an interface and using polymorphism to create various implementations of this interface.

Pure fabrication - A pure fabrication is a class that does not represent a concept in the problem domain, specially made up to achieve low coupling, high cohesion, and the reuse potential. This kind of class is called a "service" in domain-driven design.