Турсунов Баходурхон

1.  Как принцип единой ответственности (SRP) помогает улучшить разработку программного обеспечения и в чем его основная идея?

В целом принципы Solid – это не паттерны, их нельзя назвать какими-то определенными догмами, которые надо обязательно применять при разработке, однако их использование позволят улучшить код программы, упростить возможные его изменения и поддержку.

Принцип единой ответственности (Single Responsibility Principle) можно сформулировать так:

Каждый компонент должен иметь одну и только одну причину для изменения.

В C# в качестве компонента может выступать класс, структура, метод. А под обязанностью здесь понимается набор действий, которые выполняют единую задачу. То есть суть принципа заключается в том, что класс/структура/метод должны выполнять одну единственную задачу. Весь функционал компонента должен быть целостным, обладать высокой связностью.

2.  Почему SRP иногда трудно реализовать, особенно когда обязанности класса не сразу ясны?

3.  Как принцип открытости/закрытости (OCP) способствует расширению программных модулей, не требуя их модификации?

4.  Почему в соответствии с OCP важно, чтобы модули были открыты для расширения, но закрыты для модификации?

5.  Какая фундаментальная концепция лежит в основе принципа замещения Лискова (LSP), когда речь идет о подклассах и базовых классах?

6.  Как принцип замещения Лискова (LSP) гарантирует, что родительские экземпляры могут заменять дочерние экземпляры без неожиданного поведения?

Practice:

Реализуйте небольшой программный проект или модифицируйте существующий, чтобы продемонстрировать применение принципов SOLID, уделяя особое внимание SRP, OCP и LSP. Ниже приведены шаги для практического занятия:

1.  Выберите небольшой программный проект или модуль, над которым вы уже работали, или создайте новый, если это необходимо.

2.  Определите класс или компонент в проекте, который мог бы выиграть от следования принципу единой ответственности (SRP). Переделайте этот класс так, чтобы он в большей степени соответствовал SRP, разделив его обязанности на более мелкие, более сфокусированные классы.

3.  Если возможно, найдите возможность применить в проекте принцип открытости/закрытости (OCP). Внесите изменения, чтобы обеспечить расширяемость без изменения существующего кода.

4.  Проверьте соблюдение принципа замещения Лискова (LSP) в иерархии классов вашего проекта. Убедитесь, что подклассы могут быть заменены на базовые классы, не вызывая неожиданного поведения.