Вопросы по теме ООП

1. Принципы SOLID
2. Что такое ООП
3. Основные принципы ООП
4. Основные понятия ООП
5. Преимущества ООП
6. Недостатки ООП
7. Что подразумевается под «является» и «имеет» в плане ООП

Принципы SOLID:

Принцип единой ответственности (Single Responsibility Principle) – никогда не должно быть больше одной причины изменить класс. Т.е. один класс должен решать только одну какую-то задачу. Класс может содержать сколько угодно методов, но все эти методы должны быть направлены на решение этой задачи.

Принцип открытости-закрытости (Open-Closed Principle) – программные сущности должны быть открыты для расширения, но закрыты для изменения. Т.е. мы должны реализовывать класс так, чтобы можно было изменять его внешнее поведение, не внося изменения в сам класс.

Принцип подстановки Барбары Лисков (Liskov Substitution Principle) – объекты в программе можно заменить их наследниками без изменения свойств программы. Цель этого принципа заключаются в том, чтобы классы-наследники могли бы использоваться вместо родительских классов, от которых они образованы, не нарушая работу программы.

Принцип разделения интерфейса (Interface Segregation Principle) – клиенты не должны быть вынуждены реализовывать интерфейс, который они не будут использовать. Т.е. не нужно создавать интерфейс, который выполняет много функций. Лучше разбить его на много одно-функциональных, чтобы разработчик при необходимости подключал к разрабатываемому классу только то, что ему необходимо.

Принцип инверсии зависимостей (Dependency Inversion Principle) – все зависимости между классами должны строится на уровне абстракций.

1. Модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба должны зависеть от абстракции.
2. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

Что такое ООП. Основные принципы ООП.

ООП – набор правил, принципов, понятий в программировании, направленный на проецирование объектов реального мира и бизнес-процессов в программный код.

Основные принципы ООП:

Инкапсуляция – контроль доступа к полям и методам объекта (сокрытие данных с помощью модификаторов доступа).

Наследование – создание новой сущности на базе существующей.

Полиморфизм – способность иметь много реализаций одного интерфейса.

Абстракция – выделение главных характеристик объекта.

Основные понятия ООП.

Класс – программный способ описания реальной сущности (чертеж объекта)

Объект – отдельный представитель сущности (экземпляр класса)

Интерфейс – указывает, что именно должен делать класс, но не как это делать. Способ реализации выпирает сам класс. Предназначен для снижения уровеня зависимости сущностей друг от друга, добавив больше абстракции.

Преимущества и недостатки ООП (по сравнению с процедурными языками)

Преимущества:

1. Сложные приложения проще создавать, т.к. все легко разбивается на мелкие задачи, которые удобно разделить между разными программистами / отделами.
2. Приложения, основанные на принципах ООП, более просты для модификаций и расширения.
3. Возможность использования одного кода много раз (в промышленных масштабах, подразумевается использование существующих библиотек и фреймворков).
4. Высокая начальная скорость разработки продукта (за счет использования большого количества библиотек и фреймворков).

Недостатки:

1. Много ненужного кода в больших приложениях (за счет использования большого количества библиотек и фреймворков).
2. Большое потребление памяти и снижение производительности (из-за большого количества ненужного кода).
3. В больших приложениях сложная иерархия наследования.

Что подразумевается под «является» и «имеет» в плане ООП.

1. **Является** – подразумевает наследование (класс Автомобиль является наследником класса Транспортное средство)
2. **Имеет** – подразумевает ассоциацию (класс Автомобиль имеет класс Двигатель)

Ассоциация – это связь между объектами. Композиция и агрегация – это частные случаи ассоциации.

Композиция – более строгий вид связи (автомобиль и двигатель).

Агрегация – менее строгий вид связи (автомобиль и парковочное место).