Три кита ООП.

1. Кратко об ООП
   1. Вопрос – что такое объектно-ориентированное программирование? Своими словами.
   2. Причины, по которым родилось ООП:
      1. Мир привычнее видеть с помощью объектов.
      2. Раньше программирование было функциональным, затем функции стали очень большими и стало неудобно работать.
      3. Плюсом ООП относительно функционального программирования, стало заключение функций в отдельные сущности, которые были близки в своем выполнении по бизнес-логике.
   3. Основные концепции ООП:
      1. Инкапсуляция
      2. Наследование
      3. Абстракция
      4. Полиморфизм
   4. Показать слайд и спросить – что НЕ является принципом ООП?
   5. (Шуточная иллюстрация основных принципов)
2. Инкапсуляция
   1. Что такое инкапсуляция?
      1. Спросить – что такое инкапсуляция?
      2. Ответ: Инкапсуляция заключается в объединении данных и методов, которые с ними работают, внутри одного объекта. При этом доступ к данным объекта ограничивается (через модификаторы доступа), чтобы скрыть внутреннюю реализацию и предоставить только необходимый интерфейс.
   2. Содержимое класса
      1. Показать пример **EncapsulationSimpleExample**
      2. Показать код класса
      3. Показать клиентский код
   3. Права доступа
      1. Спросить – как вы думаете: зачем нужны права доступа?
      2. Ответ:
         1. «Защита от дурака». Избежание ошибок: Например, если поле доступно только через метод, вы можете контролировать его изменения.
         2. Нет доступа в те области объекта, изменение которых может привести к неблагоприятным последствиям.
         3. Есть доступ только к тому, что необходимо видеть и не думать о неиспользуемых частях.
      3. Пролистать слайды с бытовыми примерами инкапсуляции
      4. Показать пример **EncapsulationExample**
   4. Решить задачи
3. Наследование
   1. Спросить - что такое наследование своими словами?
   2. Ответ: наследование - это механизм, который позволяет использовать возможности других классов.
   3. Показать синтаксис наследования на слайде.
   4. Показать пример синтаксиса наследования на слайде.
   5. Что наследуется:
      1. Рассказать, что наследуется на слайде.
      2. Показать синтаксис наследования в проекте **WhatIsInheritance**
   6. Что не наследуется:
      1. Рассказать, что наследуется с нюансами на слайде.
      2. Показать синтаксис наследования в проекте **WhatIsInheritance**
         1. Sealed
            1. Раскомментировать sealed у класса **NotInheritance**
            2. Раскомментировать sealed у метода **SomeClass. SealedMethodTest – ошибка компиляции.**
         2. Конструкторы
            1. Показать ключевое слово base. Явный вызов. Оно требуется только для передачи параметров
            2. Даже если не указан явный вызов конструктора, код конструктора родительского класса все равно выполнится.
            3. Показать в коде **Program.cs**
         3. Статичные данные не выдадут ошибку компиляции, если попытаться их использовать, но обращаться они будут все равно к родительскому методу.
         4. При попытке обращения к приватному полю/методу, компилятор будет ругаться и скомпилировать код не получится. Однако получить доступ к приватному члену класса все равно можно, через методы в классе предка. Главное, чтобы прав доступа хватало.
   7. Что такое множественное наследование и почему оно запрещено в C#?
      1. Показать пример что такое множественное наследование.
      2. Показать UML c множественным наследованием и прочитать вопрос со слайда.
   8. Транзитивное наследование. Просто прочитать со слайда.
   9. Ключевое слово base
4. Полиморфизм
5. Абстракция
   1. Спросить что такое абстракция?
   2. Дать корректный ответ.
   3. Показать пример и сказать зачем его использовать вместо интерфейса.(Контр страйк)
   4. Упомянуть интерфейсы.

<https://skillbox.ru/media/code/oop_chast_1_chto_takoe_klassy_i_obekty/>

<https://blog.skillfactory.ru/glossary/polimorfizm/>