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. Раскомментироватьsealed у класса**NotInheritance**
            2. Раскомментироватьsealedу 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. Статичныеданные 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 UMLcмножественным наследованием и прочитать вопрос со слайда.
   8. Транзитивное наследование. Просто прочитать со слайда.
   9. Ключевое слово base
      1. Зачитать слайд.
      2. Показать наследование метода. Проект **Base**
      3. Показать наследование конструктора. Проект **Base**
   10. Ключевое слово virtual
       1. Зачитать слайд virtual.
       2. Зачитать слайд override.
       3. Показать наследование метода. Проект **virtual**
       4. Показать наследование конструктора. Проект **virtual**
   11. Вопросы, на которые отвечает наследование
   12. Решение задач
4. Абстракция
   1. Спросить что такое абстракция своими словами?
      1. Зачитать текст со слайда: **Что такое абстракция?**
      2. Показать слайд: **Суть абстракции.** Привести также в пример «небесное тело». Кто-то представляет звезду, кто-то планету, кто-то метеорит, кто-то комету. И все варианты подходят.
      3. Наиболее яркий бытовой пример абстракции – армейский. Командир говорит солдату: сделай что-то и мне неважно как ты это сделаешь.
   2. Реализация абстракции в C# состоит в использовании абстрактных классов и интерфейсов. Проект **SimpleExample**
      1. Ключевое слово abstract – проект **SimpleExample**.Запустить проект и показать:
         1. Невозможность создать объекты интерфейса и абстрактного класса через ключевое слово new.
         2. Создать объект сапера и вызвать метод ToDo интерфейса.
         3. Вызвать метод SubmitRapport, реализованный в интерфейсе.
         4. Вызвать метод SubmitRapport, который переопределен в классе сапера.
         5. Вызвать метод JustGo, определенный в классе Human.
5. Полиморфизм
   1. Вопрос студентам: Что такое полиморфизм? Своими словами
      1. Полиморфизм – это свойство системы использовать объекты без информации о реализации.
      2. Примеры:
         1. Транспорт. Начальник нам сказал добраться из точки А в точку Б. С точки зрения бизнеса – все равно как устроены велосипед, автобус, такси или личный автомобиль. Необходимо просто знать, что это транспорт. Слайд с транспортом.
         2. Допустим, мы - руководитель зоопарка. У нас есть животные. Все животные кушают и имеют какой-то рацион питания. Все остальное – детали реализации.
         3. Показать пример: проект **Polymorphism**.
      3. Параметрический полиморфизм. По сути это работа с дженериками. Мы имеем какой-то универсальный тип, к которому мы обращаемся не зная деталей реализации. Проект **ParametricPolyMorphism**
      4. Под аd hoc-полиморфизмом обычно понимается перегрузка методов. Проект **AdHocPolymorphism**
   2. **Прорешать задачи.**
6. UML

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

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