**Тема: «Отношения между классами. Механизм наследования. Интерфейсы»**

**Задание 1: Работа с наследованием**

В каждой задаче:

* **Абстрактный базовый класс**.
* **Три наследника**.
* **Массив элементов базового класса**.
* **Бизнес-логика с массивом**.

**1. Создание системы сотрудников**

Разработать абстрактный класс Employee с абстрактным методом CalculateSalary(). Создать три производных класса: Manager, Developer, Intern. Создать массив Employee[], заполнить разными объектами, вывести список сотрудников с зарплатами.

**2. Животные и их звуки**

Создать абстрактный класс Animal с методом MakeSound(). Создать три наследника: Dog, Cat, Cow. Создать массив животных и вывести звуки каждого.

**3. Геометрические фигуры**

Разработать абстрактный класс Shape с методом GetArea(). Создать три класса: Circle, Rectangle, Triangle. Заполнить массив Shape[], вычислить и вывести площади всех фигур.

**4. Оплата товаров в магазине**

Создать абстрактный класс PaymentMethod с методом Pay(amount). Создать три метода оплаты: CreditCard, PayPal, Cash. Заполнить массив разными способами оплаты и провести платежи.

**5. Система транспортных средств**

Создать абстрактный класс Transport с методом Move(). Создать три транспорта: Car, Bike, Airplane. Создать массив транспортных средств и запустить движение каждого.

**6. Устройства вывода информации**

Создать абстрактный класс OutputDevice с методом DisplayInfo(). Создать три устройства: Monitor, Printer, Projector. Создать массив устройств и вывести информацию с каждого.

**7. Виды обучения**

Создать абстрактный класс LearningMode с методом GetLearningType(). Создать три типа обучения: Online, Offline, Hybrid. Заполнить массив видами обучения и вывести их особенности.

**8. Музыкальные инструменты**

Создать абстрактный класс MusicalInstrument с методом PlaySound(). Создать три инструмента: Guitar, Piano, Drum. Создать массив инструментов и сыграть на каждом.

**9. Работники завода**

Создать абстрактный класс Worker с методом DoWork(). Создать три рабочих: Welder, Assembler, Electrician. Создать массив рабочих и выполнить их работу.

**10. Способы передвижения**

Создать абстрактный класс Movement с методом Move(). Создать три класса: Walking, Running, Cycling. Создать массив передвижений и запустить каждое.

**11. Типы билетов в кино**

Создать абстрактный класс Ticket с методом GetPrice(). Создать три типа билетов: Standard, VIP, Student. Заполнить массив билетами и вывести их стоимость.

**12. Объекты солнечной системы**

Создать абстрактный класс CelestialBody с методом GetType(). Создать три класса: Planet, Star, Asteroid. Заполнить массив небесными телами и вывести их информацию.

**13. Виды пользователей в системе**

Создать абстрактный класс User с методом GetPermissions(). Создать три типа пользователей: Admin, Moderator, Guest. Заполнить массив пользователями и вывести их права доступа.

**14. Способы доставки товаров**

Создать абстрактный класс DeliveryMethod с методом Deliver(). Создать три метода доставки: Courier, Pickup, Post. Заполнить массив способами доставки и выполнить их работу.

**15. Типы спортивных состязаний**

Создать абстрактный класс SportEvent с методом StartEvent(). Создать три вида спорта: Football, Swimming, Tennis. Заполнить массив соревнованиями и запустить каждое.

**Задание 2: Работа с ассоциацией, агрегацией и композицией.**

В каждой задаче:

* **Должны быть классы, связанные между собой отношениями ассоциации, агрегации и композиции**.
* **Бизнес-логика, связанная с массивами объектов**.

**1. Автомобиль и его компоненты**

Создать класс Car, который имеет:

* **Агрегация:** массив Wheel[] (колёса могут устанавливаться на разные автомобили).
* **Композиция:** объект Engine (двигатель создаётся внутри автомобиля).
* **Ассоциация:** связь с Driver (водитель может водить разные автомобили).  
  Создать массив автомобилей и реализовать метод Drive().

**2. Книга и её содержание**

Создать класс Book, который имеет:

* **Агрегация:** массив Author[] (у книги может быть несколько авторов).
* **Композиция:** объект TableOfContents (содержание создаётся внутри книги).
* **Ассоциация:** связь с Library (книги принадлежат библиотеке).  
  Создать массив книг и реализовать метод DisplayInfo().

**3. Университет и студенты**

Создать класс University, который имеет:

* **Агрегация:** массив Student[] (студенты могут переходить между университетами).
* **Композиция:** объект Department (факультеты создаются внутри университета).
* **Ассоциация:** связь с Professor (преподаватель работает в нескольких университетах).  
  Создать массив университетов и реализовать метод ShowStudents().

**4. Компания и её сотрудники**

Создать класс Company, который имеет:

* **Агрегация:** массив Employee[] (работники могут менять компанию).
* **Композиция:** объект Department (отделы существуют внутри компании).
* **Ассоциация:** связь с Client (клиенты могут взаимодействовать с компанией).  
  Создать массив компаний и реализовать метод CalculateTotalSalary().

**5. Телефон и его модули**

Создать класс Smartphone, который имеет:

* **Агрегация:** массив App[] (приложения можно установить на разные телефоны).
* **Композиция:** объект Battery (батарея встроена в телефон).
* **Ассоциация:** связь с User (пользователь может владеть несколькими телефонами).  
  Создать массив телефонов и реализовать метод ShowInstalledApps().

**6. Город и его инфраструктура**

Создать класс City, который имеет:

* **Агрегация:** массив Citizen[] (жители могут переезжать).
* **Композиция:** объект Infrastructure (инфраструктура создаётся внутри города).
* **Ассоциация:** связь с Country (страна состоит из нескольких городов).  
  Создать массив городов и реализовать метод ShowPopulation().

**7. Спортсмен и его экипировка**

Создать класс Athlete, который имеет:

* **Агрегация:** массив Coach[] (спортсмен может тренироваться у нескольких тренеров).
* **Композиция:** объект Equipment (экипировка создаётся внутри спортсмена).
* **Ассоциация:** связь с Team (спортсмен может быть частью команды).  
  Создать массив спортсменов и реализовать метод Train().

**8. Ресторан и его меню**

Создать класс Restaurant, который имеет:

* **Агрегация:** массив Chef[] (повара могут работать в разных ресторанах).
* **Композиция:** объект Menu (меню создаётся внутри ресторана).
* **Ассоциация:** связь с Supplier (ресторан закупает продукты у поставщиков).  
  Создать массив ресторанов и реализовать метод PrepareDishes().

**9. Кинотеатр и его сеансы**

Создать класс Cinema, который имеет:

* **Агрегация:** массив Movie[] (фильмы могут показываться в разных кинотеатрах).
* **Композиция:** объект Schedule (расписание создаётся внутри кинотеатра).
* **Ассоциация:** связь с Distributor (кинотеатр получает фильмы от дистрибьюторов).  
  Создать массив кинотеатров и реализовать метод ShowMovies().

**10. Магазин и его товары**

Создать класс Store, который имеет:

* **Агрегация:** массив Supplier[] (поставщики могут работать с разными магазинами).
* **Композиция:** объект Inventory (товарный запас создаётся внутри магазина).
* **Ассоциация:** связь с Customer (покупатели могут делать покупки в магазине).  
  Создать массив магазинов и реализовать метод SellProduct().

**11. Больница и её пациенты**

Создать класс Hospital, который имеет:

* **Агрегация:** массив Doctor[] (врачи могут работать в разных больницах).
* **Композиция:** объект MedicalRecord (история болезни создаётся внутри больницы).
* **Ассоциация:** связь с Pharmacy (больница сотрудничает с аптекой).  
  Создать массив больниц и реализовать метод TreatPatients().

**12. Театр и его постановки**

Создать класс Theater, который имеет:

* **Агрегация:** массив Actor[] (актёры могут участвовать в разных постановках).
* **Композиция:** объект Stage (сцена создаётся внутри театра).
* **Ассоциация:** связь с Audience (зрители могут посещать театр).  
  Создать массив театров и реализовать метод PerformPlay().

**13. Туристическое агентство и его путешествия**

Создать класс TravelAgency, который имеет:

* **Агрегация:** массив Guide[] (гиды могут работать в разных агентствах).
* **Композиция:** объект TourPackage (турпакет создаётся внутри агентства).
* **Ассоциация:** связь с Customer (туристы могут заказывать туры).  
  Создать массив агентств и реализовать метод BookTour().

**14. Видеоигра и её персонажи**

Создать класс VideoGame, который имеет:

* **Агрегация:** массив Player[] (игроки могут менять игры).
* **Композиция:** объект GameWorld (мир игры создаётся внутри неё).
* **Ассоциация:** связь с Developer (разработчик создаёт игру).  
  Создать массив игр и реализовать метод StartGame().

**15. Фитнес-клуб и его клиенты**

Создать класс Gym, который имеет:

* **Агрегация:** массив Trainer[] (тренеры могут работать в разных клубах).
* **Композиция:** объект GymEquipment (тренажёры принадлежат клубу).
* **Ассоциация:** связь с Membership (клиенты подписываются на абонемент).  
  Создать массив фитнес-клубов и реализовать метод TrainClients().

**Задание 3: Работа с базовым классом, интерфейсами и наследованием**

**В каждой задаче должно быть:**

* **Один базовый класс.**
* **Два интерфейса.**
* **Два типа наследников, реализующих каждый свой интерфейс.**
* **Один массив элементов базового класса.**
* **Бизнес-логика по поиску и фильтрации элементов по типам интерфейсов.**

**1. Животные в зоопарке**

Создать базовый класс Animal.

* Интерфейс ICanFly (для птиц).
* Интерфейс ICanSwim (для рыб).
* Классы Bird и Fish, реализующие соответствующие интерфейсы.  
  Создать массив животных и отфильтровать только тех, кто умеет летать.

**2. Транспортные средства**

Создать базовый класс Vehicle.

* Интерфейс IElectric (для электромобилей).
* Интерфейс IDiesel (для дизельных машин).
* Классы ElectricCar и Truck, реализующие соответствующие интерфейсы.  
  Создать массив транспортных средств и найти все дизельные машины.

**3. Музыкальные инструменты**

Создать базовый класс Instrument.

* Интерфейс IStringInstrument (для струнных инструментов).
* Интерфейс IPercussionInstrument (для ударных).
* Классы Guitar и Drum, реализующие соответствующие интерфейсы.  
  Создать массив инструментов и найти только струнные.

**4. Работники компании**

Создать базовый класс Employee.

* Интерфейс IManager (для управленцев).
* Интерфейс IWorker (для обычных работников).
* Классы Director и Engineer, реализующие соответствующие интерфейсы.  
  Создать массив сотрудников и найти всех управленцев.

**5. Программисты и дизайнеры**

Создать базовый класс ITSpecialist.

* Интерфейс IProgrammer (для кодеров).
* Интерфейс IDesigner (для дизайнеров).
* Классы BackendDeveloper и UXDesigner, реализующие соответствующие интерфейсы.  
  Создать массив специалистов и отфильтровать программистов.

**6. Учебные заведения**

Создать базовый класс EducationInstitution.

* Интерфейс IHasOnlineCourses (для онлайн-курсов).
* Интерфейс IHasCampus (для классических университетов).
* Классы OnlineSchool и University, реализующие соответствующие интерфейсы.  
  Создать массив заведений и найти только онлайн-школы.

**7. Банковские счета**

Создать базовый класс BankAccount.

* Интерфейс IDebitAccount (для дебетовых счетов).
* Интерфейс ICreditAccount (для кредитных счетов).
* Классы SavingsAccount и LoanAccount, реализующие соответствующие интерфейсы.  
  Создать массив счетов и найти все кредитные.

**8. Умные устройства**

Создать базовый класс SmartDevice.

* Интерфейс ICanPlayMusic (для устройств с музыкой).
* Интерфейс ICanMakeCalls (для телефонов).
* Классы SmartSpeaker и Smartphone, реализующие соответствующие интерфейсы.  
  Создать массив устройств и найти те, которые могут звонить.

**9. Спортивные команды**

Создать базовый класс TeamMember.

* Интерфейс IBasketballPlayer (для баскетболистов).
* Интерфейс IFootballPlayer (для футболистов).
* Классы Basketballer и Footballer, реализующие соответствующие интерфейсы.  
  Создать массив игроков и найти баскетболистов.

**10. Автомобильные детали**

Создать базовый класс CarPart.

* Интерфейс IElectricalComponent (для электронных деталей).
* Интерфейс IMechanicalComponent (для механических деталей).
* Классы Battery и Gearbox, реализующие соответствующие интерфейсы.  
  Создать массив деталей и найти все механические.

**11. Художественные произведения**

Создать базовый класс ArtPiece.

* Интерфейс IPainting (для картин).
* Интерфейс ISculpture (для скульптур).
* Классы Portrait и Statue, реализующие соответствующие интерфейсы.  
  Создать массив произведений и найти все скульптуры.

**12. Лекарства и препараты**

Создать базовый класс Medicine.

* Интерфейс IPill (для таблеток).
* Интерфейс ILiquidMedicine (для сиропов).
* Классы Antibiotic и CoughSyrup, реализующие соответствующие интерфейсы.  
  Создать массив лекарств и найти все сиропы.

**13. Игровые персонажи**

Создать базовый класс GameCharacter.

* Интерфейс IMeleeFighter (для ближнего боя).
* Интерфейс IRangedFighter (для дальнего боя).
* Классы Knight и Archer, реализующие соответствующие интерфейсы.  
  Создать массив персонажей и найти всех лучников.

**14. Офисная техника**

Создать базовый класс OfficeEquipment.

* Интерфейс IPrinter (для принтеров).
* Интерфейс IScanner (для сканеров).
* Классы LaserPrinter и DocumentScanner, реализующие соответствующие интерфейсы.  
  Создать массив техники и найти все сканеры.

**15. Космические корабли**

Создать базовый класс Spacecraft.

* Интерфейс ICargoShip (для грузовых кораблей).
* Интерфейс IPassengerShip (для пассажирских).
* Классы Freighter и Shuttle, реализующие соответствующие интерфейсы.  
  Создать массив кораблей и найти пассажирские.

**Задание 4: Работа с интерфейсами и явной реализацией**

**В каждой задаче должно быть:**

* **Два интерфейса с одинаковыми методами.**
* **Один класс, реализующий оба интерфейса, используя явную реализацию.**
* **Объект класса, доступ к методам интерфейсов только через их ссылки.**
* **Демонстрация вызова методов через разные ссылки интерфейсов.**

**1. Управление звуком**

Создать интерфейсы ISpeaker и IMicrophone, оба содержат метод void AdjustVolume(int level);.  
Класс AudioDevice реализует оба интерфейса с явной реализацией.  
Продемонстрировать вызов метода AdjustVolume через объекты ISpeaker и IMicrophone.

**2. Системы уведомлений**

Создать интерфейсы IEmailNotifier и ISmsNotifier, оба содержат метод void SendNotification(string message);.  
Класс NotificationService реализует оба интерфейса с явной реализацией.  
Продемонстрировать вызов метода через объекты IEmailNotifier и ISmsNotifier.

**3. Графический редактор**

Создать интерфейсы IDraw и IPaint, оба содержат метод void ApplyColor(string color);.  
Класс GraphicEditor реализует оба интерфейса с явной реализацией.  
Вызвать метод ApplyColor через оба интерфейса.

**4. Финансовые операции**

Создать интерфейсы ICreditPayment и IDebitPayment, оба содержат метод void ProcessPayment(decimal amount);.  
Класс PaymentProcessor реализует оба интерфейса с явной реализацией.  
Показать вызов метода через объекты ICreditPayment и IDebitPayment.

**5. Работа с базами данных**

Создать интерфейсы ISqlDatabase и INoSqlDatabase, оба содержат метод void Connect();.  
Класс DatabaseConnector реализует оба интерфейса с явной реализацией.  
Показать вызов метода Connect через оба интерфейса.

**6. Контроль температуры**

Создать интерфейсы ICooler и IHeater, оба содержат метод void AdjustTemperature(int degrees);.  
Класс ClimateControl реализует оба интерфейса с явной реализацией.  
Вызвать метод AdjustTemperature через оба интерфейса.

**7. Электронные устройства**

Создать интерфейсы IPowerOn и IPowerOff, оба содержат метод void TogglePower();.  
Класс Device реализует оба интерфейса с явной реализацией.  
Вызвать метод TogglePower через оба интерфейса.

**8. Управление файлами**

Создать интерфейсы IReadFile и IWriteFile, оба содержат метод void AccessFile(string fileName);.  
Класс FileManager реализует оба интерфейса с явной реализацией.  
Вызвать метод через оба интерфейса.

**9. Логирование событий**

Создать интерфейсы IErrorLogger и IEventLogger, оба содержат метод void Log(string message);.  
Класс Logger реализует оба интерфейса с явной реализацией.  
Вызвать метод Log через оба интерфейса.

**10. Виртуальный помощник**

Создать интерфейсы ITextAssistant и IVoiceAssistant, оба содержат метод void Respond(string query);.  
Класс SmartAssistant реализует оба интерфейса с явной реализацией.  
Вызвать метод Respond через оба интерфейса.

**11. Хранение данных**

Создать интерфейсы ICloudStorage и ILocalStorage, оба содержат метод void SaveData(string data);.  
Класс StorageManager реализует оба интерфейса с явной реализацией.  
Продемонстрировать вызов метода через оба интерфейса.

**12. Сетевые соединения**

Создать интерфейсы IWiFiConnection и IEthernetConnection, оба содержат метод void Connect();.  
Класс NetworkAdapter реализует оба интерфейса с явной реализацией.  
Вызвать метод Connect через оба интерфейса.

**13. Видеопроигрыватель**

Создать интерфейсы IVideoPlayer и IAudioPlayer, оба содержат метод void Play(string fileName);.  
Класс MediaDevice реализует оба интерфейса с явной реализацией.  
Вызвать метод Play через оба интерфейса.

**14. Фильтрация данных**

Создать интерфейсы IImageFilter и IVideoFilter, оба содержат метод void ApplyFilter(string filterName);.  
Класс MediaProcessor реализует оба интерфейса с явной реализацией.  
Вызвать метод ApplyFilter через оба интерфейса.

**15. Управление задачами**

Создать интерфейсы ITaskScheduler и IAlarmScheduler, оба содержат метод void Schedule(string taskName);.  
Класс Scheduler реализует оба интерфейса с явной реализацией.  
Вызвать метод Schedule через оба интерфейса.