**Тема: «Делегаты, события»**

**Задание 1: Создание собственного делегата**

**Требования:**

* В каждой задаче необходимо создать пользовательский делегат.
* Должно быть два класса с методами, которые будут обработчиками делегата.
* Необходимо продемонстрировать работу с методами через делегат.
* Уровень сложности: **3**.

**1. Делегат для обработки сообщений**

Создайте делегат MessageHandler, который принимает строку.  
Создайте два класса ConsoleLogger и FileLogger, методы которых будут обрабатывать сообщения.  
Демонстрируйте работу с делегатом.

**2. Делегат для выполнения математических операций**

Создайте делегат MathOperation, который принимает два числа и возвращает результат.  
Создайте два класса Addition и Multiplication, реализующих методы сложения и умножения.  
Передавайте методы в делегат и вызывайте его.

**3. Делегат для отправки уведомлений**

Создайте делегат NotificationHandler, который обрабатывает строковые уведомления.  
Реализуйте два класса EmailNotifier и SmsNotifier, методы которых отправляют уведомления.  
Продемонстрируйте работу с делегатом.

**4. Делегат для работы с текстом**

Создайте делегат TextProcessor, который принимает и возвращает строку.  
Создайте два класса UpperCaseConverter и LowerCaseConverter, реализующих методы преобразования текста.  
Покажите работу через делегат.

**5. Делегат для расчета скидки**

Создайте делегат DiscountCalculator, который принимает цену и возвращает новую цену.  
Создайте классы StudentDiscount и SeniorDiscount, реализующие методы расчета скидок.  
Используйте делегат для вычисления стоимости товаров.

**6. Делегат для сортировки массива**

Создайте делегат SortMethod, который принимает массив и возвращает отсортированный массив.  
Создайте два класса BubbleSort и QuickSort, реализующих соответствующие алгоритмы.  
Передавайте методы в делегат и вызывайте их.

**7. Делегат для работы с временем**

Создайте делегат TimeHandler, который принимает объект DateTime и возвращает строку.  
Создайте классы TimeFormatter и DateFormatter, реализующие разные форматы вывода.  
Передавайте методы в делегат и демонстрируйте их работу.

**8. Делегат для преобразования чисел**

Создайте делегат NumberConverter, который принимает число и возвращает строку.  
Создайте классы BinaryConverter и HexConverter, реализующие перевод в двоичную и шестнадцатеричную системы.  
Продемонстрируйте работу делегата.

**9. Делегат для работы с файловой системой**

Создайте делегат FileHandler, который принимает путь к файлу и выполняет операцию.  
Создайте классы FileReader и FileWriter, реализующие чтение и запись в файл.  
Продемонстрируйте использование делегата.

**10. Делегат для выполнения операций над списком**

Создайте делегат ListProcessor, который принимает список чисел и возвращает список.  
Создайте классы FilterEvenNumbers и MultiplyByTwo, реализующие фильтрацию четных чисел и умножение каждого элемента на 2.  
Используйте делегат для обработки списка.

**11. Делегат для управления светом**

Создайте делегат LightControl, который включает и выключает свет.  
Создайте классы RoomLight и StreetLight, которые управляют освещением в комнате и на улице.  
Продемонстрируйте работу через делегат.

**12. Делегат для управления громкостью звука**

Создайте делегат VolumeControl, который принимает громкость в процентах.  
Создайте классы Speaker и Headphones, методы которых управляют громкостью.  
Демонстрируйте работу через делегат.

**13. Делегат для перевода валют**

Создайте делегат CurrencyConverter, который принимает сумму и возвращает новую сумму в другой валюте.  
Создайте классы DollarToEuro и EuroToYen, которые выполняют конвертацию.  
Используйте делегат для перевода валют.

**14. Делегат для работы с заказами в ресторане**

Создайте делегат OrderHandler, который принимает заказ и выполняет операцию.  
Создайте классы CookOrder и DeliverOrder, реализующие приготовление и доставку.  
Продемонстрируйте работу с делегатом.

**15. Делегат для управления доступом в систему**

Создайте делегат AccessControl, который проверяет права пользователя.  
Создайте классы AdminAccess и UserAccess, реализующие разные уровни доступа.  
Продемонстрируйте использование делегата.

**Задание 2: Делегаты как параметры методов**

**Требования:**

* В каждой задаче организовать передачу делегата в качестве параметра метода.
* Использовать два метода в качестве **callback** (методы обратного вызова).
* Уровень сложности: **3**.

**1. Делегат для обработки строк**

Создайте делегат StringProcessor, который принимает строку.  
Метод ProcessString должен принимать строку и делегат, который обрабатывает её.  
Создайте два метода: ToUpperCase и ToLowerCase. Передавайте их в ProcessString.

**2. Делегат для выполнения математических операций**

Создайте делегат MathOperation, который принимает два числа и возвращает результат.  
Метод Calculate должен принимать два числа и делегат, который выполняет операцию.  
Создайте два метода Add и Multiply, передавайте их в Calculate.

**3. Делегат для проверки чисел**

Создайте делегат NumberCheck, который принимает число и возвращает bool.  
Метод FilterNumbers должен принимать массив чисел и делегат, который проверяет каждое число.  
Создайте методы IsEven и IsOdd, передавайте их в FilterNumbers.

**4. Делегат для логирования сообщений**

Создайте делегат LogHandler, который принимает строку.  
Метод LogMessage должен принимать сообщение и делегат, который выполняет логирование.  
Создайте методы ConsoleLog и FileLog, передавайте их в LogMessage.

**5. Делегат для обработки заказов**

Создайте делегат OrderProcessor, который принимает идентификатор заказа.  
Метод ProcessOrder должен принимать ID заказа и делегат, который его обрабатывает.  
Создайте методы ApproveOrder и CancelOrder, передавайте их в ProcessOrder.

**6. Делегат для работы с датами**

Создайте делегат DateFormatter, который принимает DateTime и возвращает строку.  
Метод FormatDate должен принимать дату и делегат, который форматирует её.  
Создайте методы ShortFormat и LongFormat, передавайте их в FormatDate.

**7. Делегат для обработки массивов**

Создайте делегат ArrayProcessor, который принимает массив и возвращает обработанный массив.  
Метод ProcessArray должен принимать массив и делегат, который его изменяет.  
Создайте методы SortAscending и SortDescending, передавайте их в ProcessArray.

**8. Делегат для работы с пользователями**

Создайте делегат UserAction, который принимает ID пользователя.  
Метод PerformUserAction должен принимать ID и делегат, который выполняет действие.  
Создайте методы BlockUser и UnblockUser, передавайте их в PerformUserAction.

**9. Делегат для работы с изображениями**

Создайте делегат ImageProcessor, который принимает путь к изображению.  
Метод ProcessImage должен принимать путь и делегат, который выполняет обработку.  
Создайте методы ResizeImage и ConvertToGrayscale, передавайте их в ProcessImage.

**10. Делегат для фильтрации текстов**

Создайте делегат TextFilter, который принимает строку и возвращает bool.  
Метод FilterText должен принимать массив строк и делегат, который проверяет каждую строку.  
Создайте методы ContainsDigit и ContainsSpecialCharacters, передавайте их в FilterText.

**11. Делегат для обработки заказов в магазине**

Создайте делегат OrderHandler, который принимает сумму заказа.  
Метод HandleOrder должен принимать сумму и делегат, который выполняет обработку.  
Создайте методы ApplyDiscount и CalculateTax, передавайте их в HandleOrder.

**12. Делегат для настройки конфигурации**

Создайте делегат ConfigSetter, который принимает строковый параметр.  
Метод SetConfiguration должен принимать параметр и делегат, который применяет настройку.  
Создайте методы SetDatabaseConfig и SetCacheConfig, передавайте их в SetConfiguration.

**13. Делегат для обработки событий клавиатуры**

Создайте делегат KeyEventHandler, который принимает код клавиши.  
Метод HandleKeyPress должен принимать код клавиши и делегат, который выполняет действие.  
Создайте методы OnKeyEnter и OnKeyEscape, передавайте их в HandleKeyPress.

**14. Делегат для анализа данных**

Создайте делегат DataAnalyzer, который принимает массив данных.  
Метод AnalyzeData должен принимать массив и делегат, который выполняет анализ.  
Создайте методы CalculateAverage и FindMaximum, передавайте их в AnalyzeData.

**15. Делегат для работы с учетными записями**

Создайте делегат AccountManager, который принимает идентификатор аккаунта.  
Метод ManageAccount должен принимать ID и делегат, который выполняет действие.  
Создайте методы DeleteAccount и RestoreAccount, передавайте их в ManageAccount.

**Задание 3: Реализация паттерна "Издатель-Подписчик" (пользовательский делегат)**

**Требования:**

* Реализовать паттерн **"Издатель-Подписчик"** с пользовательским делегатом.
* В каждой задаче должен быть **класс-издатель** с событием, основанным на **пользовательском делегате**.
* Два класса-подписчика, которые обрабатывают событие.
* Уровень сложности: **3**.

**1. Оповещение о новом заказе**

Создайте класс OrderManager, который генерирует событие OrderPlaced.  
Два подписчика: EmailNotifier (отправляет email) и SmsNotifier (отправляет SMS).

**2. Изменение статуса заказа**

Создайте класс OrderStatusManager, который генерирует событие StatusChanged.  
Два подписчика: CustomerNotifier (оповещает клиента) и AdminLogger (записывает в лог).

**3. Контроль температуры в помещении**

Создайте класс TemperatureSensor, который генерирует событие TemperatureChanged.  
Два подписчика: CoolingSystem (включает кондиционер) и WarningSystem (предупреждает о перегреве).

**4. Система мониторинга акций**

Создайте класс StockMarket, который генерирует событие StockPriceChanged.  
Два подписчика: Investor (анализирует изменения) и NewsAgency (публикует новости).

**5. Изменение уровня громкости**

Создайте класс VolumeControl, который генерирует событие VolumeChanged.  
Два подписчика: Display (отображает уровень) и SpeakerSystem (изменяет громкость).

**6. Отслеживание загрузки файлов**

Создайте класс FileDownloader, который генерирует событие DownloadProgress.  
Два подписчика: ProgressBar (обновляет индикатор) и Logger (записывает ход загрузки).

**7. Уведомление о входе пользователя**

Создайте класс UserLoginManager, который генерирует событие UserLoggedIn.  
Два подписчика: SecuritySystem (проверяет доступ) и NotificationService (отправляет уведомление).

**8. Оповещение о завершении работы сервера**

Создайте класс ServerShutdownManager, который генерирует событие ServerShuttingDown.  
Два подписчика: BackupService (создаёт резервную копию) и AlertSystem (оповещает админа).

**9. Контроль уровня заряда батареи**

Создайте класс BatteryMonitor, который генерирует событие BatteryLow.  
Два подписчика: PowerSaver (включает режим энергосбережения) и UserNotifier (предупреждает пользователя).

**10. Отслеживание изменений в файле**

Создайте класс FileWatcher, который генерирует событие FileChanged.  
Два подписчика: BackupService (делает резервную копию) и Logger (записывает изменения).

**11. Уведомление о новых сообщениях**

Создайте класс ChatApplication, который генерирует событие MessageReceived.  
Два подписчика: DesktopNotifier (показывает всплывающее окно) и SoundAlert (издаёт звуковой сигнал).

**12. Оповещение об изменении погодных условий**

Создайте класс WeatherStation, который генерирует событие WeatherChanged.  
Два подписчика: DisplayPanel (обновляет данные) и WarningSystem (предупреждает о шторме).

**13. Контроль над уровнем воды в резервуаре**

Создайте класс WaterTankSensor, который генерирует событие WaterLevelChanged.  
Два подписчика: PumpController (включает насос) и WarningSystem (предупреждает о переполнении).

**14. Оповещение о завершении обработки данных**

Создайте класс DataProcessor, который генерирует событие ProcessingCompleted.  
Два подписчика: ReportGenerator (создаёт отчёт) и Notifier (оповещает пользователя).

**15. Контроль движения на складе**

Создайте класс WarehouseMonitor, который генерирует событие ItemMoved.  
Два подписчика: InventorySystem (обновляет данные) и SecuritySystem (проверяет разрешения).

**Задание 4: Реализация паттерна "Издатель-Подписчик" (EventHandler)**

**Требования:**

* В каждой задаче должен быть класс-издатель с **событиями, основанными на EventHandler**.
* Должен быть **промежуточный класс**, который подписывает подписчиков на события издателя.
* Два подписчика, которые обрабатывают событие.
* Уровень сложности: **4**.

**1. Система уведомлений о новых заказах**

Создайте класс OrderManager с событием OrderPlaced, основанным на EventHandler<OrderEventArgs>.  
Промежуточный класс OrderNotifier подписывает EmailService (отправляет email) и SmsService (отправляет SMS).

**2. Контроль температуры в помещении**

Создайте класс TemperatureSensor с событием TemperatureChanged.  
Промежуточный класс TemperatureMonitor подписывает CoolingSystem (включает кондиционер) и AlarmSystem (оповещает о перегреве).

**3. Система мониторинга биржевых котировок**

Создайте класс StockMarket с событием StockPriceUpdated.  
Промежуточный класс MarketObserver подписывает Investor (анализирует данные) и NewsPublisher (публикует новости).

**4. Контроль заряда батареи**

Создайте класс BatteryMonitor с событием BatteryLow.  
Промежуточный класс PowerManager подписывает PowerSaver (включает режим энергосбережения) и UserNotifier (предупреждает пользователя).

**5. Отслеживание изменений в файле**

Создайте класс FileWatcher с событием FileChanged.  
Промежуточный класс FileMonitor подписывает BackupService (делает резервную копию) и Logger (записывает изменения).

**6. Уведомление о новых сообщениях**

Создайте класс ChatApplication с событием MessageReceived.  
Промежуточный класс MessageNotifier подписывает DesktopNotification (показывает всплывающее окно) и SoundAlert (издаёт звуковой сигнал).

**7. Оповещение об изменении погодных условий**

Создайте класс WeatherStation с событием WeatherChanged.  
Промежуточный класс WeatherMonitor подписывает DisplayPanel (обновляет данные) и WarningSystem (предупреждает о шторме).

**8. Контроль движения на складе**

Создайте класс WarehouseMonitor с событием ItemMoved.  
Промежуточный класс InventoryTracker подписывает InventorySystem (обновляет данные) и SecuritySystem (проверяет разрешения).

**9. Уведомление о входе пользователя**

Создайте класс UserLoginManager с событием UserLoggedIn.  
Промежуточный класс LoginObserver подписывает SecuritySystem (проверяет доступ) и NotificationService (отправляет уведомление).

**10. Оповещение о завершении обработки данных**

Создайте класс DataProcessor с событием ProcessingCompleted.  
Промежуточный класс ProcessingMonitor подписывает ReportGenerator (создаёт отчёт) и Notifier (оповещает пользователя).

**11. Контроль уровня воды в резервуаре**

Создайте класс WaterTankSensor с событием WaterLevelChanged.  
Промежуточный класс WaterMonitor подписывает PumpController (включает насос) и WarningSystem (предупреждает о переполнении).

**12. Изменение громкости системы**

Создайте класс VolumeControl с событием VolumeChanged.  
Промежуточный класс VolumeManager подписывает Display (отображает уровень) и SpeakerSystem (изменяет громкость).

**13. Завершение работы сервера**

Создайте класс ServerShutdownManager с событием ServerShuttingDown.  
Промежуточный класс ShutdownMonitor подписывает BackupService (создаёт резервную копию) и AlertSystem (оповещает администратора).

**14. Система отслеживания загрузки файлов**

Создайте класс FileDownloader с событием DownloadProgressChanged.  
Промежуточный класс DownloadMonitor подписывает ProgressBar (обновляет индикатор) и Logger (фиксирует ход загрузки).

**15. Изменение статуса заказа**

Создайте класс OrderStatusManager с событием StatusChanged.  
Промежуточный класс StatusObserver подписывает CustomerNotifier (оповещает клиента) и AdminLogger (записывает в