#9 Делегаты, события и лямбда-выражения.

Задание:

1. Используя класс-коллекцию из предыдущей работы (#8), познакомьтесь с делегатами и событиями:
   https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/
   https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/events/
   a. Разработайте свой класс **EventArgs**, содержащий какую-то полезную нагрузку;
   b. Добавьте в свою иерархию 2 различных события с вашим кастомным классом **EventArgs**. Осуществите переопределение метода-триггера события, если необходимо;
   c. Добавьте в класс-коллекцию методы, реагирующий на события иерархии;
   d. Осуществите отписку от событий в деструкторе класса-коллекции.
   e. Продемонстрируйте работу событий и методов-подписчиков.
2. Изучите стандартные типы делегатов **Action**, **Func**, **Predicate**:
   https://docs.microsoft.com/en-us/dotnet/api/system.action?view=netframework-4.8
   https://docs.microsoft.com/en-us/dotnet/api/system.func-1?view=netframework-4.8
   https://docs.microsoft.com/en-us/dotnet/api/system.predicate-1?view=netframework-4.8
3. Добавьте в класс-коллекцию методы, принимающие параметрами каждый из стандартных типов делегатов, например:
   a. **Action** – выполнение полезной нагрузки с каждым объектом коллекции;
   b. **Func** – функция-маппер, преобразующая объект коллекции к новому типу;
   c. **Predicate** – поиск объектов, удовлетворяющих условию.
4. Используйте **lambda**-синтаксис везде, где это возможно:
   https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions
5. Используйте **try-catch** для обработки потенциально опасных мест кода.

Повышенный уровень:

1. Выделите ваш логгер из л.р. #7 в отдельный проект типа *class library*.
2. Создайте статический класс **LoggerFactory**, который будет содержать 2 перегрузки метода Log:
    a. Первая перегрузка принимает:
        i. Сообщение, которое нужно залоггировать;
        ii. Тип события (*warning*, *information*, *error*): используйте перечисление, созданное ранее;
        iii. Тип логгера, который нужно использовать: создайте перечисление с двумя полями: *FileLogger* и *ConsoleLogger*, соответственно, и используйте его значение.
    b. Вторая перегрузка принимает:
        i. Объект **Exception**.
        ii. Тип логгера (по аналогии с методом выше)
3. Постарайтесь не допустить дублирования кода.
4. Реорганизуйте иерархию классов таким образом, чтобы из сборки был доступен только статический класс и его методы.
5. Соберите проект и полученную сборку **.dll* подключите к проекту текущей работы.
6. Реализуйте обработку ошибок с помощью вашего класса **LoggerFactory** (тип логгера определите сами).

Вопросы:

1. Что такое делегаты? Каково их предназначение? Каким типом данных является делегат: ссылочным или значимым?
2. Как создать делегат?
3. Как можно присвоить делегату адрес метода?
4. Какими способами можно вызвать делегат? Возможно ли присвоить делегату сразу несколько разных методов?
5. Что такое событие? Какой синтаксис объявления события?
6. Как события связаны с делегатами? Что такое “***object*** *sender*”?
7. Что такое ковариантность и контравариантность делегатов? В чем их преимущества?
8. В чем разница между **Func** и **Action**?