В классе Main (пакет 'com.epam.jwd'):
1. Создать массив из 4 объектов Point
2. Создать массив из 2 объектов Line (объект Line состоит из 2 объектов Point)
3. Создать массив из 2 объектов Triangle (объект Triangle состоит из 3 объектов Point)
4. Создать массив из 1 объекта Square (объект Square состоит из 4 объектов Point)

5. Используя do/while вывести информацию о элементах (toString) массива Point (используя INFO log level)
6. Используя for вывести информаци о элементах (toString) массивов (используя INFO log level):
    6.1. Line
    6.2. Triangle (в случае, если треугольник не существует, НЕ выводить его информацию, но
                вывести информацию, формата 'Треугольник (toString) не может существовать'
                используя ERROR log level)
    6.3. Square (в случае, если объект – не квадрат, НЕ выводить его информацию, но
                вывести информацию, формата Объект (toString) не является квадратом'
                используя ERROR log level)
    6.4. В случае, если объект Triangle, Line, Square содержать одинаковые точки, вывести информацию, формата
                        Объект (toString) не является фигурой (figure name)'
                используя ERROR log level) вместо пунктов 6.1, 6.2, 6.3

---

7. Pattern Strategy
    7.1. Создать поле figurePropertiesStrategy в классе Figure.
    7.2. Создать пакет 'com.epam.jwd.strategy' в котором реализовать соответствующий паттерн (подсчёт площади и периметра, если это возможно).
8. Pattern Singleton
    8.1. Для реализованных стратегий, применить паттерн Singleton (1 Lazy. 1 ENUM. Остальные самые простые)
9. Pattern Factory
    9.1. Создать пакет 'com.epam.jwd.model' в котором реализовать паттерн Factory, который будет создавать единственный экземпляр фигуры с заданными параметрами.
    9.2. Запретить создание фигур из вне фабрики.
10. Создать класс 'model.MultiAngleFigure' который наследует Figure.
    10.1. Создать несколько экземпляров N-угольников (4 <= N <= 6)

---

11. Exceptions
    11.1. Создать пакет 'com.epam.jwd.exception' в котором создать семейство checked exception FigureException.
        11.1.1. Создать наследника FigureException - FigureNotExistException
        11.1.2. Добавить FigureNotExistException в сигнатуру метода фабрики.
    11.2. Создать пакет 'com.epam.jwd.service' в котором создать interface FigurePostProcessor с методом 'Figure process(Figure figure) throws FigureException'
        11.2.1. В подпакете 'com.epam.jwd.service.impl' реализовать интерфейс классом FigureExistencePostProcessor.
        11.2.2. Вызывать метод 'process' в конце работы фабрики.

---

12. Pattern Decorator
    12.1. Отрефакторить существующий код.
    12.2. Создать новый интерфейс 'FigureFactory' в пакете 'com.epam.jwd.factory'.
        12.2.1. Интерфейс должен содержать единственный метод 'Figure createFigure(FigureType type, Point[] figureConstituents) throws FigureException'
        12.2.2. Реализация 'SimpleFigureFactory' должна отвечать ТОЛЬКО за создание фигуры (Single Responsibility principle).
            12.2.2.1. Реализация все еще находится в пакете 'com.epam.jwd.model'
        12.2.3. Создать абстрактный класс FigureFigureDecorator implements FigureFactory – которая будет оборачивать метод createFigure в пакете 'com.epam.jwd.decorator'.
        12.2.4. Создать PreProcessing/PostProcessing|Factory extends FigureFactoryDecorator, которая будет выполнять pre/post-процессинг
            12.2.4.1. Данные реализации также будут находиться в пакете 'com.epam.jwd.decorator'.
        12.2.5. [необязательное]: препроцессоров/постпроцессоров может быть много. Реализовать добавление/убирание их.
        12.2.6. Создать класс ApplicationContext (рядом с фабрикой) для оборачивания вызовов декоратора
    12.3. Применить паттерн Singleton для мест, где он необходим

13. Collections
    13.1. Отрефакторить существующий код. Применить, где необходимо принцип Барбары Лисков
    13.2. Заменить существующие массивы на коллекции
    13.3. Использовать generics, где необходимо
    13.4. В пакете 'com.epam.jwd.service' создать интерфейс FigureCrud (в подпакете impl реализовать его классом FigureCrudImpl)
        (в пунктах 13.4.* подразумевается работа с Collection, StreamAPI, Optional в вашем Storage)
        13.4.1. Интерфейс должен содержать методы создания фигуры (Create), создания многих фигур за раз (MultiCreate), удаления (Delete), поиска (Find), обновления (Update),
            поиска по индексу (FindById), поиска по критерию (для построения критерии применить паттерн Builder)