**Примеры нарушения каждого принципа SOLID:**

**1. S – Single Responsibility Principle**

**Нарушение:** класс BirdHandler делает и создание объектов, и выполнение действий над ними. **Почему плохо:** изменение логики создания птиц или логики действий требует правки одного и того же класса.

**2. O – Open/Closed Principle**

**Нарушение:** метод ProduceBird() использует if-else, и чтобы добавить новую птицу, нужно менять этот метод. **Почему плохо:** класс не открыт для расширения, но открыт для модификации – при добавлении новых птиц придётся менять старый код.

**3. L – Liskov Substitution Principle**

**Нарушение:** Pinguin реализует метод Fly(), но он выбрасывает исключение. **Почему плохо:** если мы заменим Sparrow на Pinguin в клиентском коде, получим ошибку – значит, подстановка работает некорректно.

**4. I – Interface Segregation Principle**

**Нарушение:** интерфейс IBird требует реализации Fly() даже от тех, кто не умеет летать. **Почему плохо:** Pinguin вынужден реализовать метод, который ему не нужен.

**5. D – Dependency Inversion Principle**

**Нарушение:** BirdHandler зависит от конкретного класса BirdProducer. **Почему плохо:** изменение BirdProducer требует пересборки BirdHandler, а тестировать его невозможно без настоящего продюсера.