پروژه نهایی بوت‌کمپ C#: طراحی یک Message Broker

۱. شرح پروژه

هدف این پروژه، طراحی و پیاده‌سازی یک **Message Broker** با استفاده از زبان C# است که وظیفه مدیریت انتقال داده‌ها از **تولیدکننده** (Producer) به **مصرف‌کننده** (Consumer) را بر عهده دارد. این سیستم باید تضمین کند که پیام‌ها دقیقاً به همان ترتیبی که ارسال شده‌اند، به مصرف‌کننده تحویل داده شوند.

برای **پایداری داده‌ها**، پیام‌ها باید در فایل ذخیره شوند تا در صورت خاموشی یا خرابی سرور، هنگام راه‌اندازی مجدد، حذف نشوند.

Message Broker باید کاملاً مستقل از تولیدکننده و مصرف‌کننده طراحی شود.

---

۲. نیازمندی‌های اصلی

۲.۱ طراحی Endpoint برای ارسال و دریافت پیام

Message Broker باید دارای **Endpoint**هایی باشد که:

- **تولیدکننده** از طریق آن‌ها پیام‌ها را ارسال کند.
- **مصرف‌کننده** از طریق آن‌ها پیام‌ها را دریافت کند.

۲.۲ مدیریت پردازش هم‌زمان (Threading)

- تولیدکننده‌ها و مصرف‌کننده‌ها باید بتوانند **تعداد Threadهای موردنظرشان** را برای ارسال و پردازش پیام‌ها مشخص کنند.
- تعداد Threadها در **DLLهای تولیدکننده و مصرف‌کننده** تعیین شده و Message Broker نباید از آن‌ها اطلاعی داشته باشد.
- امکان دارد تعداد Threadها در تولیدکننده و مصرف‌کننده متفاوت باشد.

۲.۳ حفظ ترتیب پیام‌ها

- پیام‌ها باید **دقیقاً به همان ترتیبی** که ارسال شده‌اند، به مصرف‌کننده تحویل داده شوند.

۲.۴ ذخیره‌سازی و بازیابی پیام‌ها از فایل

- پیام‌ها باید در **فایل** ذخیره شوند تا در صورت **خاموشی یا خرابی سرور**، داده‌ها از بین نروند.

- هنگام راه‌اندازی مجدد، Message Broker باید بتواند تمامی پیام‌های ذخیره‌شده را **بازیابی** کند.

۲.۵ استفاده از مکانیزم افزونه (Plugin System)

- Message Broker نباید وابستگی مستقیمی به تولیدکننده و مصرف‌کننده داشته باشد.
- یک **اینترفیس (Interface)** و مجموعه‌ای از **ویژگی‌ها (Attributes)** برای تعامل با سیستم تعریف می‌شود.
- پروژه‌های **Third-Party** (تولیدکننده و مصرف‌کننده) باید این اینترفیس‌ها را پیاده‌سازی کنند و از طریق **بارگذاری پویا (Dynamic Loading)** اجرا شوند.

۲.۶ مکانیزم ارسال مجدد پیام‌ها

- تولیدکننده و مصرف کننده باید dll واسط خود را برای اجرای متدهای پیاده سازی شده داشته باشند.
- واسط تولید کننده باید تعداد دفعات تلاش مجدد را از dll پیاده سازی تولیدکننده و از طریق **ویژگی‌ها (Attributes)** دریافت کند.

```
[AttributeUsage(AttributeTargets.Class)]
class ProducerImplementationData : Attribute
{
    public ProducerImplementationData(int retryNumber)
    {
        RetryNumber = retryNumber;
    }
    public int RetryNumber { get; }
}

[ProducerImplementationData(retryNumber: 3)]
class ProducerImplementation
{
    // Implementation
}
```

- اگر سرور در زمان ارسال پیام‌ها **غیرفعال باشد**، تولیدکننده باید:
  1. به تعداد دفعات مشخص شده در ویژگی‌ها **تلاش مجدد** برای ارسال پیام انجام دهد.
  2. در صورت عدم موفقیت، **مدتی صبر کند** و مجدداً ارسال را ادامه دهد.
- اگر سرور در زمان دریافت پیام ها غیر فعال باشد، مصرف کننده باید صبر کند تا سرور دوباره فعال شود.
- dll تولید کننده و مصرف کننده باید تعداد Thread های همزمانی که میتوانند برای تولید و مصرف ایجاد شوند را از طریق Attribute با عنوان RateLimit به برنامه تولید کننده و مصرف کننده اعلام کنند.

۲.۷ پیاده‌سازی سیستم لاگینگ (Logging)

- لاگ‌گیری باید برای **تمامی عملیات کلیدی** از جمله ارسال و دریافت پیام، ذخیره‌سازی، بازیابی، خطاها و تلاش‌های مجدد انجام شود.
- سیستم لاگ باید امکان **ذخیره در فایل و نمایش در کنسول** را فراهم کند.
- امکان **پیکربندی سطح لاگ** (Info، Warning، Error) باید وجود داشته باشد.

۲.۸ مستندات راه‌اندازی (Setup Documentation)

- باید مستندات کاملی برای نحوه **راه‌اندازی و پیکربندی** Message Broker ارائه شود.
- این مستندات شامل **نصب، اجرای نمونه، تنظیمات تولیدکننده و مصرف‌کننده، و مثال‌های عملی** خواهد بود.

---

۳. امتیازی: مدیریت چند مصرف‌کننده (Multiple Consumers)

چگونه می‌توان **دو مصرف‌کننده** داشت، بدون اینکه پیام‌های یکدیگر را پردازش کنند؟