# 🧠 Zadanie: Rozszerzenie walidacji kontrahenta

## 📋 Kontekst biznesowy

W aplikacji do zarządzania kontrahentami istnieje kontroler `CustomerController`, który podczas dodawania nowego kontrahenta sprawdza poprawność numeru NIP.

🆕 **Nowe wymagania biznesowe**:

Klient systemu zgłosił potrzebę rozszerzenia tej walidacji o dodatkowe elementy:

- numer REGON,
- adres e-mail kontrahenta.

Zależy mu na tym, aby rozwiązanie było elastyczne i pozwalało łatwo dodawać kolejne reguły walidacji w przyszłości — **bez modyfikacji istniejącego kodu**!

## 🔧 Obecny kod

```
[ApiController]
[Route("api/[controller]")]
public class CustomerController : ControllerBase
{
    private readonly IValidator _validator;

    public CustomerController(IValidator validator)
    {
        _validator = validator;
    }

    [HttpPost]
    public IActionResult Post([FromBody] Customer customer)
    {
        if (!_validator.Validate(customer))
        {
            return BadRequest("Validation failed.");
        }

        // Tu normalnie byłby kod dodający kontrahenta do bazy danych
        return Created();
    }
}
```

```
public interface IValidator
{
    bool Validate(Customer customer);
}

public class TaxNumberValidator : IValidator
{
    public bool Validate(Customer customer)
    {
        return !string.IsNullOrWhiteSpace(customer?.TaxNumber);
    }
}

public class Customer
{
    public string TaxNumber { get; set; }
    public string Regon { get; set; }
    public string Email { get; set; }
}
```

---

## 🎯 Zadanie

1. Zaimplementuj dodatkowe walidatory: `RegonValidator`, `EmailValidator`.
2. Przygotuj rozwiązanie zgodne z zasadą **otwarte/zamknięte** (Open/Closed Principle).

## 🧪 Testy jednostkowe

1. Dodaj testy jednostkowe, które:
   - potwierdzą poprawne działanie walidatora, gdy wszystkie dane są poprawne,
   - zgłoszą błąd walidacji, gdy którykolwiek z warunków nie zostanie spełniony.