Value Objects

Value Objects

- brak rozróżnienia – brak identyfikacji
  - VO są takie same gdy ich atrybuty są takie same
- zwykle są immutable – ponieważ nie mają tożsamości
  - zatem można ich reużywać – nie dbamy o konkretną instancję, brak efektów ubocznych
- wyrażają jakieś domenowe znaczenie – wchodzą do słownictwa
  - zwiększają siłę wyrazu koncepcji i czytelności kodu
  - więcej znaczenia niż zwykły **`string`**, **`int`** itp.
- zawierają użyteczne metody (zamiast **`Utils`**)
  - walidacja (np. w konstruktorze?)
- przykłady: kolor, punkt, address, numer telefonu, money

Value Objects - przykład

```
public class DocumentNumber : ValueObject
{
   public const int MaxLen = 128;
   private string _value;
   public DocumentNumber(string value)
   {
      _value = value;
   }
   public static implicit operator DocumentNumber(string number)
   {
      return new DocumentNumber(number);
   }
   public static implicit operator string(DocumentNumber number)
   {
      if (number != null) { return number._value; }
      return null;
   }
   public override string ToString()
   {
      return _value;
   }
}
```

Value Objects - zapis do bazy danych

```
public OrderMap()
{
   Lazy(false);
   Id(x => x.Id);
   //...
   Property("_created", m => m.Type(new ValueTypeAsStringType<Date>())); }
```

Value Objects - typowi kandydaci

- `string` z ograniczeniami formatowania
  - zip code
  - nazwa
- liczby z ograniczeniami
  - procent (jako ułamek? jako liczba całkowita?)
  - ilość w jednostkach
- złożone struktury
  - money (również waluta, data)
  - adres (również czas obowiązywania)
  - przedział czasu (operacje: przecięcia, czas trwania,...)
- obiekty zwracane przez metody innych klas

Value Objects - typowi kandydaci (kont.)

- Money
  - hermetyzuje implementację (BigDecimal, Integer z przesuniętym przecinkiem)
  - hermetyzuje reprezentację – bazowa waluta
  - zawiera wygodne metody: przeliczanie (?)
- PhoneNumber
  - hermetyzuje reprezentację
  - zawiera wygodne metody: ekstrakcja numeru kierunkowego, walidacja, etc

Value Objects - zastosowania

- wrappery typów "technicznych" nadające znaczenie biznesowe
- parametry – sposób na wymianę danych między złożonymi obiektami
  - hermetyzacja wewnętrznej struktury
- Clean Code, redukcja "code smell":
  - Primitive Obsession, Data Clumps, Long Parameter List