Dobre praktyki programistyczne

Systemy notacji ciągów tekstowych

- Do najpopularniejszych systemów notacji ciągów znakowych należą:
  - Pascal case
  - Camel case
  - Snake case

Pascal case

- Zakłada, że kolejne wyrazy należy pisać łącznie, każdy rozpoczynając wielką literą
- Popularny np. w języku C#
  - ExampleClass
  - ExampleService

Camel case

- Zakłada, że kolejne wyrazy należy pisać łącznie, każdy (poza pierwszym) rozpoczynając wielką literą
- Powszechnie stosowany w języku Java
  - peopleList
  - fileHelper
  - createListOfPeople

Snake case

- Zakłada, że kolejne wyrazy należy pisać, rozpoczynając małą literą oraz oddzielając je znakiem podkreślenia
- Powszechnie stosowany w języku Python
  - my_cars
  - read_list
  - find_longest_word

DRY, KISS, SOLID, DEMETER…

DRY

- Don’t Repeat Yourself
- Nadrzędna zasada w świecie obiektowym – nie należy powtarzać kodu
- Najczęstszym sposobem na stosowanie DRY w kontekście kodu, jest zastosowanie metod w klasach
- Najważniejsza zaleta DRY – oszczędność czasu

KISS

- Keep It Simple, Stupid! → reguła z inżynierii wojskowej, zaimplementowana w wielu dziedzinach (w tym wytwarzaniu oprogramowania)
- KISS w IT zakłada, że projekt powinien być tworzony od początku do końca w sposób maksymalnie zrozumiały, bez niepotrzebnego
- Zasada odnosi się zarówno do etapu projektowania, jak i implementacji
- KISS ma szczególne znaczenie w kontekście utrzymania projektu – zespół może zmieniać się z biegiem czasu, stosowanie KISS pozwoli na łatwiejsze wdrażanie nowych członków zespołu

SOLID

- Mnemonik zaproponowany przez Roberta Martina, jako skrót nazw pięciu podstawowych zasad programowania obiektowego
  - Single Responsibility Principle
  - Open / Closed Principle
  - Liskov Substitution Principle
  - Interface Segragation Principle
  - Dependency Inversion Principle

Zasada Jednej Odpowiedzialności

- Klasa, metoda → powinna mieć jedną odpowiedzialność
- Powinien istnieć tylko jeden powód do modyfikacji
- Np. klasa przechowująca dane użytkownika, nie powinna jednocześnie zawierać metod walidujących te dane

Zasada „Otwarty / Zamknięty”

- Klasa powinna być otwarta na rozszerzanie, zamiast modyfikowanie,
- Oznacza to, że w przypadku zmiany wymagań, powinno być możliwe dodanie nowego kodu, zmieniającego działanie klasy, zamiast zmieniania istniejącego

Zasada Podstawienia Liskovej

- Korzystanie z klasy pochodnej powinno być możliwe wszędzie tam, gdzie oczekiwana jest jej klasa bazowa
- Np. jeśli osoba jest klasą bazową, po której dziedziczy student, powinna być możliwość podstawienia klasy dziedziczącej wszędzie tam, gdzie oczekiwana jest klasa bazowa
- Klasa bazowa nie powinna mieć wiedzy o swoich klasach dziedziczących

Zasada Segregacji Interfejsów

- Wykorzystując interfejsy należy pamiętać, iż lepszym rozwiązaniem jest zastosowaniem wielu różnych interfejsów (wyspecjalizowanych) zamiast jednego, ogólnego

Zasada Odwrócenia Zależności

- Moduły wysokopoziomowe nie powinny zależeć od niskopoziomowych

Prawo Demeter

- Inaczej: zasada minimalnej wiedzy ▫
- Zakłada, że metoda danego obiektu może odwoływać się do: ▫
  - Danego obiektu
  - Parametru przekazanego do tej metody
  - Obiektu utworzonego przez tę metodę
  - Innego składnika klasy, do której należy ta metoda

Najważniejsze zasady

Znaczące nazwy - zasady

- Nazwy powinny przedstawiać intencje
- Unikanie dezinformacji
- Tworzenie wyraźnych różnic
- Nazwy klas – rzeczowniki lub wyrażenia rzeczownikowe
- Nazwy metod – czasowniki lub wyrażenia czasownikowe
- Unikanie śmiesznych nazw
- Unikanie określeń, znanych małej grupie osób

Funkcje i ich przeznaczenie

- Funkcje powinny wykonywać jedną czynność
- Należy przygotowywać małe funkcje
- W ramach klasy – zasada zstępująca
- Należy wykorzystywać nazwy znaczące

Komentarze

- Czytelny kod nie wymaga komentarzy ☺
- Kiedy stosować komentarze?
  - Wyjaśnienie nietypowych rozwiązań
  - Komentarze typu TODO
  - Podkreślenie konieczności wykonania pewnych operacji
- Unikajmy:
  - Komentarzy oczywistych
  - Zakomentowanego kodu
  - Komentarzy mylących