## Kornel Chromiński

1. Paradygmat programowania obiektowego (OOP) – jest paradygmatem programowania opartym na koncepcji „obiektów”, które mogą zawierać dane i kod. Dane często w postaci pól (atrybuty lub właściwości) oraz kod w postaci funkcji (zwanych jako metody).

Wspólną cechą obiektów jest to, że są do nich dołączane metody, które mogą uzyskiwać dostęp do pól danych obiektu i modyfikować je (Gettery i Settery).

W OOP programy komputerowe projektuje się tworząc je z obiektów, które wchodzą ze sobą w interakcje. Często języki OOP są oparte na klasach, co oznacza, że obiekty są instancjami klas, które określają również ich typy.

1. Normalizacja baz danych – jest to proces strukturyzacji relacyjnej bazy danych zgodnie z serią tzw. form normalnych w celu zmniejszenia redundancji danych i poprawy ich integralności. Bez normalizacji bazy danych dane często się powtarzają, co wpływa nie tylko na zajętość pamięci, ale też na szybkość wyszukiwania w bazie danych, jak i modyfikacji w niej danych.

Występują trzy podstawowe postaci normalne

* Pierwsza postać normalna (1NF)
* Druga Postać Normalna (2NF)
* Trzecia Postać Normalna (3NF)

Pierwsza Postać Normalna (1NF) – podzielenie kolumn na niepodzielne kolumny. Dane muszą być atomowe. Przykładowo kolumna ADRES (Wrocław ul. Jana Pawła 2) powinna zostać podzielona na dwie kolumny MIASTO (Wrocław) i ULICĘ (Jana Pawła 2). Wciąż istnieje problem z nadmiarowymi danymi oraz z niepotrzebnymi nadmiarowymi operacjami aktualizacji bazy danych.

Druga Postać Normalna (2NF) – Baza danych jest w drugiej postaci normalnej, gdy spełnia pierwszą postać normalną oraz wszystkie kolumny w tabeli zależą tylko od klucza. W 2NF dzieli się tabelę na więcej tabel.

Trzecia Postać Normalna (3NF) – Baza jest w 3NF wtedy, gdy spełnia drugą postać normalną, oraz żadna z kolumn nie jest zależna od innej kolumny, która nie jest kluczem.

Jest jeszcze 4NF i 5NF, jednak dalsze dzielenie tabeli na podtabele nie jest często stosowane

1. Czym są i w jakim celu używa się interfejsów i klas abstrakcyjnych

Interfejsy – są strukturą, która definiuje „co potrafi zrobić” dany obiekt, a nie „jak to robi”. Interfejsy posiadają jedynie zdefiniowane operacje, nie zawiera natomiast ich implementacji i danych (metody są czysto wirtualne). Z tego powodu Klasy mogą implementować wiele interfejsów, bez problemów wynikających z wielokrotnego dziedziczenia. Interfejs pozwala na hermetyzację obiektów.

Klasy abstrakcyjne – klasy, dla których nie ma możliwości stworzenia instancji obiektu, a są one jedynie uogólnieniem dla klas **potomnych**. Po klasach abstrakcyjnych **dziedziczą** klasy, w których szczegółowo można opisać działanie funkcji zapoczątkowanych w klasie abstrakcyjnej. Klasy potomne dziedziczą metody i zmienne z klas abstrakcyjnych, mogą je modyfikować, nadpisywać, bądź być kolejną warstwą abstrakcji dla klas **potomnych**.

1. Na czym polega metodyka wytwarzania oprogramowania TDD

TDD (Test-Driven Development) – technika tworzenia oprogramowania, w której główną ideą jest w pierwszej kolejności pisanie testów do nieistniejącej funkcjonalności, a dopiero potem pisanie kodu implementującego tę funkcjonalność.

Zaletą tworzenia oprogramowania metodą TDD jest to, że większość błędów zostanie wykrytych podczas testów przed wprowadzeniem kodu w środowisko testowe lub produkcyjne. Służy minimalizacji zakresu występowania błędów.

CYKLE:

1. Napisanie przypadku testowego
2. Uruchomienie napisanego testu
3. Napisanie minimalnego kodu do przejścia testu
4. Refraktoryzacja kodu
5. Powtórzenie cyklu, dopóki nie zostaną napisane wszystkie testy pokrywające logikę biznesową.
6. Czym jest i w jakim celu wykonuje się testowanie statyczne

Testowanie statyczne jest testowaniem programu bez wykonywania tego oprogramowania. Techniki statyczne polegają na sprawdzeniu ręcznym (przeglądy kodu) lub analizie automatycznej kodu bez uruchamiania tego kodu. Testowanie statyczne sprawdza podstawową poprawność kodu i pozwala ocenić, czy program jest gotowy na bardziej szczegółowe testowanie.

W odróżnieniu od testowania dynamicznego celem testowania statycznego jest tańsze wyszukanie i naprawa defektów w projekcie. Najczęściej są przeprowadzone zanim dynamicznie uruchomi się produkt. Błędy wykryte dzięki przeglądowi we wczesnej fazie cyklu tworzenia oprogramowania są zazwyczaj znacznie tańsze do usunięcia niż te wykryte podczas wykonywania testów (np. defekty znalezione w wymaganiach).

1. Omówić mechanizm dziedziczenia

Dziedziczenie – Jeden z najważniejszych paradygmatów programowania obiektowego. Jest to rodzaj relacji pomiędzy dwoma klasami, która pozwala jednej z nich dziedziczyć kod od drugiej klasy. Dzięki temu mechanizmowi, można budować hierarchię między klasami. Przydatny mechanizm wtedy, gdy jest konieczność stosowania wielu obiektów, które w większości posiadają cechy wspólne.

Dziedziczyć można tylko po jednej klasie.

## Urszula Boryczka

alg. Boole'a, ONP, MT, DAS, NAS, j. Chomsky'ego.

1. Algebra Boola – jest działem algebry, w którym wartościami zmiennych są wartości prawdy true i false, oznaczane zwykle 1 i 0. Głównymi operacjami algebry boolowskiej są:
   1. Koniunkcja (oraz)
   2. Dysjunkcja (lub)
   3. Negacja (nie)

Z opisanych trzech wcześniej operacji Boole’a można złożyć inne operacje poprzez kompozycje. Operacje złożone z operacji podstawowych obejmują następujące przykłady:

1. Implikacja (->)
2. XOR – wykluczenie
3. Równoważność -

Jest to więc formalizm, służący do opisu operacji logicznych. Algebra Boola miała fundamentalne znaczenie dla rozwoju elektroniki cyfrowej i jest obecna we wszystkich nowoczesnych językach programowania. Stosuje się ją również w teorii zbiorów i statystyce.

PRAWA

1. 0 + a = a
2. 1 + a = 1
3. 0 \* a = 0
4. 1 \* a = a
5. a + a = a
6. a + Na = 1
7. a \* a = a
8. a \* Na = 0
9. NNa = a

PRAWO DE MORGANA

1. ONP (Odwrotna notacja polska) – sposób zapisu wyrażeń arytmetycznych, w którym znak operacji umieszczony jest po operandach (zapis postfiksowi), a nie pomiędzy nimi jak w konwencjonalnym zapisie algebraicznym (zapis infiksowy) lub przed operandami jak w zwykłej notacji polskiej (zapis prefiksowy). Zapis ten pozwala na całkowitą rezygnację z użycia nawiasów w wyrażeniach, jako że jednoznacznie określa kolejność wykonywanych działań.

ONP ułatwia wykonywanie obliczeń na komputerze z nawiasami i zachowaniem kolejności działań. W celu zapisu wyrażeń jest wykorzystywana struktura stosu.

Przykładowy zapis ONP:

1. MT - Maszyna Turinga

Maszyna Turinga to matematyczny model opisujący abstrakcyjną maszynę, która manipuluje symbolami na nieskończonej taśmie zgodnie z tabelą reguł. Nieskończona taśma pamięci jest podzielona na kolejne komórki, z których każda może pomieścić pojedynczy symbol ze skończonego zbioru zwanego alfabetem maszyny. Maszyna posiada głowicę, która w dowolnym momencie działania jest umieszczona nad jedną z tych komórek, oraz „stan” wybrany ze skończonego zbioru stanów. Maszynę Turinga definiuje się poprzez:

Gdzie:

Q – skończony zbiór stanów,

– stan początkowy

– zbiór stanów końcowych

– skończony zbiór dopuszczalnych symboli

- symbol pusty

– zbiór symboli wejściowych

– funkcja. Funkcja pobierająca aktualny stan maszyny oraz symbol wejściowy, a dająca w wyniku symbol, jaki ma się pojawić na taśmie, kolejny stan maszyny, oraz przesunięcie głowicy maszyny (L, P albo brak przesunięcia - ) .

Maszynę Turinga można rozpatrywać za pomocą tabeli przejść albo za pomocą diagramu stanów.

1. AUTOMATY

Celem automatu jest rozstrzygnięcie w skończonej ilości kroków, czy dowolne słowo należy bądź nie należy do języka badanego przez dany automat.

**DAS** jest abstrakcyjną maszyną o skończonej liczbie stanów, która zaczynając w stanie początkowym czyta kolejne symbole pewnego słowa, a następnie po przeczytaniu każdego zmienia swój stan na stan będący wartością funkcji jednego przeczytanego symbolu oraz stanu aktualnego. Jeśli po przeczytaniu całego słowa maszyna znajduje się w którymś ze stanów oznaczonych jako stan końcowy, słowo to należy do języka regularnego, do rozpoznawania, którego jest zbudowana.

DAS, podobnie jak inne automaty skończone (NAS) może być reprezentowane za pomocą tabeli przejść, albo za pomocą DIAGRAMU STANÓW (coś w stylu GRAF).

**NAS –** jest automatem skończonym. Próbuje w skończonej liczbie kroków powiedzieć, czy dane słowo należy do badanego alfabetu. Od deterministycznego automatu skończonego różni się tym, że po przeczytaniu danego symbolu może przejść do jednego z kilku różnych stanów.

1. Język Chomsky’ego