Abstrakcja – sprawozdanie Alicja Janeczko

Na czym polega paradygmat abstrakcji w programowaniu obiektowym?

Paradygmat abstrakcji w programowaniu obiektowym to ukrycie szczegółów implementacji wewnątrz czegoś. Kiedy wywołujemy funkcję, nie musimy dokładnie rozumieć, co ona robi. Jeśli używamy interfejsu, jeśli nie mamy dostępu bezpośrednio do wnętrza, wtedy abstrakcja może zagwarantować, że program będzie poprawny. Nie musimy rozumieć implementacji, żeby używać abstrakcji. Musimy tylko zrozumieć interfejs, który jest w większości przypadków, dużo, dużo prostszy. Nie musimy rozumieć, jak zbudować samochód, aby prowadzić samochód. I możemy podzielić program na wiele niezależnych abstrakcji, co znacznie upraszcza jego użytkowanie.

Dzięki abstrakcji rozwój dużych programów staje się możliwy. Programów, które są zbyt duże, by zmieścić się w głowie jednej osoby. Ponieważ każda abstrakcja ma jedną osobę, która jest odpowiedzialna za rozwój tej abstrakcji. Tylko ta osoba musi znać tę abstrakcję i interfejsy, z którymi się łączy.

Na czym polega abstrakcja realizowana na poziome klas w C++?

Klasa, która jest szablonem, określającym metodę komunikacji z obiektami tej klasy, jak i klas pochodnych - klasa taka jest interfejsem, określającym metodę komunikacji z obiektami tworzącymi hierarchię klas. Taka klasa bazowa może być skonstruowana w C++ w specyficzny sposób - może być klasą abstrakcyjną.

Funkcje w takiej klasie są oczywiście funkcjami wirtualnymi. Ponieważ jest to abstrakcja implementowana na poziomie klasy, trudno jest wskazać, jakie powinny być ciała tych funkcji. Właściwa implementacja funkcji powinna być umieszczona w klasach pochodnych. Dokładniej rzecz ujmując, każda klasa pochodna powinna przedefiniować funkcję składową, wprowadzając jej implementację adekwatną do typu klasy pochodnej.

Rola klasy abstrakcyjnej jest specyficzna - nie deklaruje się jawnie obiektów tej klasy, pełni ona rolę klasy bazowej, stając się korzeniem, często bardzo rozbudowanej hierarchii klas. Konkretne definicje funkcji abstrakcyjnych powinny być umieszczane w klasach pochodnych.

Na czym polega abstrakcja realizowana na poziome klas i interfejsów w C# i Javie?

Obiekt jest w rzeczywistości mieszanką czystego obiektu i czystego abstrakcyjnego typu danych. Interfejs jest typem referencyjnym podobnym do klasy, który zawiera tylko stałe, deklaracje metod i zagnieżdżone typy danych. Interfejsy nie mogą być używane do tworzenia obiektów - mogą być jedynie implementowane przez klasy lub rozszerzane przez inne interfejsy. Modyfikator właściwości *abstract* definiuje klasę abstrakcyjną. Nie można tworzyć żadnych obiektów tej klasy, jest ona przydatna tylko do dziedziczenia. Klasa abstrakcyjna powinna zawierać co najmniej jedną metodę abstrakcyjną zadeklarowaną z modyfikatorem *abstract* i nie zaimplementowaną. Jeśli klasa zawiera co najmniej jedną metodę abstrakcyjną, to musi być zadeklarowana jako abstrakcyjna.

Na czym polega abstrakcja danych?

Abstrakcja danych jest główną zasadą organizacji do budowania złożonych systemów oprogramowania. Bez abstrakcji danych, technologia obliczeniowa zatrzymałaby się w miejscu. Nie mielibyśmy nic w porównaniu z tym, co mamy dzisiaj.

Istnieją w rzeczywistości dwa główne rodzaje abstrakcji danych, które są nazywane obiektami i abstrakcyjnymi typami danych. I tak naprawdę są one zupełnie różne, chociaż każdy rodzaj operacji może być opakowany jako obiekt lub abstrakcyjny typ danych, ale sposób, w jaki się z nimi współpracuje jest zupełnie inny. Abstrakcyjny typ danych przechowuje wartości i operacje oddzielnie. Abstrakcja danych posiada również ochronę pomiędzy wnętrzem a zewnętrzem. Ta ochrona istnieje zarówno dla obiektów, jak i dla abstrakcyjnych typów danych.

Czy abstrakcja w C++ może być realizowana za pomocą zbiorów nagłówkowych?

Tak, abstrakcja w C++ może być realizowana za pomocą zbiorów nagłówkowych. Zadeklarowanie pliku w zbiorze nagłówkowym pozwala ukryć jego zawartość przed programem, w którym został zadeklarowany, jednocześnie umożliwiając korzystanie z jego np. Funkcji.

Źródła:

Kurs LouvainX Louv1.2x Paradigms of Computer Programming – Abstraction and Concurrency

B. Eckel. Thinking in Java. Helion, 2006.

E. Koffman and P. Wolfgang. Struktury danych i techniki obiektowe na przykładzie Javy 5.0. Willey/Helion, 2006.

Prata S. “Szkoła programowania. Język C++”:. Helion, Gliwice, Polska, 2006. H. Schildt. Java. Kompendium programisty. Helion, 2005

W. Wieczorek, M. Boryczka, R. Simiński Programowanie obiektowe – aspekty implementacyjne

<https://www.tutorialspoint.com/java/java_abstraction.htm>

<https://www.holbertonschool.com/coding-concept-what-is-object-oriented-programming>

<https://www.freecodecamp.org/news/four-pillars-of-object-oriented-programming/>

<https://youtu.be/E4bjO275McM>

<https://youtu.be/f84bELS28vU>

<https://youtu.be/kXL4LvfQ5bA>

<https://youtu.be/4OqCPNzTH7U>

<https://youtu.be/V8SjYAHN4oA>

<https://youtu.be/UTCQDPbXxpo>

<https://youtu.be/ALPv0YhTurc>

<https://youtu.be/U9JmXdGk7EI>

<https://pl.wikipedia.org/wiki/Abstrakcja_(programowanie)>