Profesjonalne projektowanie oprogramowania: architektura kodu, wzorce, zasady, refaktoryzacja (kod: WZORCE PROJEKTOWE)

Zapytaj o szczegóły
tel. 22 63 64 164
akademia@alx.pl

Opis i cel szkolenia

Szkolenie wprowadza uczestników w świat wzorców projektowych, zasad tworzenia czytelnego, elastycznego i łatwego w utrzymaniu kodu, a także technik jego refaktoryzacji. W trakcie zajęć omawiane są najważniejsze wzorce kreacyjne, strukturalne i czynnościowe oraz sposoby ich praktycznego zastosowania. Uczestnicy poznają również typowe błędy projektowe (antywzorce) i dowiedzą się, jak skutecznie poprawiać istniejący kod poprzez refaktoryzację z wykorzystaniem wzorców. Szkolenie łączy teorię z praktyką, umożliwiając natychmiastowe przećwiczenie zdobytej wiedzy w zadaniach programistycznych.

Czas trwania

3 dni

Program

1. Wprowadzenie do projektowania z wykorzystaniem wzorców
   - Omówienie zasady pojedynczej odpowiedzialności
   - Zasada otwarte-zamknięte – tworzenie kodu łatwego do rozbudowy, ale odpornego na modyfikacje.
   - Podstawienie podtypów – jak zapewnić zgodność typów przy dziedziczeniu
   - Odwrócenie zależności – wprowadzenie do iniekcji zależności i luźnego powiązania komponentów.
   - Segregacja interfejsów – projektowanie interfejsów dostosowanych do konkretnych potrzeb klas.
2. Wzorce tworzenia obiektów (kreacyjne)
   - Builder (budowniczy) – tworzenie złożonych obiektów krok po kroku.
   - Abstract Factory (fabryka abstrakcyjna) – rodziny powiązanych obiektów bez ujawniania klas.
   - Factory Method (metoda wytwórcza) – delegowanie tworzenia obiektów do podklas.
   - Prototype (prototyp) – kopiowanie istniejących instancji zamiast tworzenia od zera.
   - Singleton (singleton) – zapewnienie istnienia tylko jednej instancji klasy w aplikacji.
3. Wzorce organizacji struktury kodu (strukturalne)
   - Adapter – przekształcanie interfejsu klasy do oczekiwanej postaci.
   - Dekorator – dynamiczne rozszerzanie funkcjonalności obiektów.
   - Fasada – uproszczenie dostępu do złożonego systemu klas.
   - Kompozyt – traktowanie grupy obiektów tak samo jak pojedynczych elementów.
   - Most – oddzielenie abstrakcji od implementacji.
   - Pełnomocnik (Proxy) – zastępowanie rzeczywistych obiektów ich reprezentacją.
   - Pyłek (Flyweight) – współdzielenie danych pomiędzy wieloma obiektami dla oszczędności zasobów.
4. Wzorce zachowań (czynnościowe)

- — Interpreter – analiza i przetwarzanie języka zgodnie z regułami gramatycznymi.
- — Iterator – sekwencyjny dostęp do elementów kolekcji bez ujawniania jej struktury.
- — Łańcuch zobowiązań – przekazywanie żądania przez łańcuch obiektów aż zostanie obsłużone.
- — Mediator – centralizacja komunikacji między komponentami.
- — Metoda szablonowa – definiowanie szkieletu algorytmu w klasie bazowej.
- — Obserwator – powiadamianie zainteresowanych obiektów o zmianach stanu.
- — Odwiedzający (Visitor) – separacja operacji od struktury danych.
- — Pamiątka (Memento) – zapisywanie i przywracanie stanu obiektów.
- — Polecenie (Command) – enkapsulacja żądań jako obiektów.
- — Stan (State) – zmiana zachowania obiektu w zależności od jego stanu wewnętrznego.
- — Strategia – wybieranie algorytmu w czasie działania programu.
- — RAII (Resource Acquisition Is Initialization) – zarządzanie zasobami poprzez inicjalizację obiektów.

5. Błędy projektowe – antywzorce
   - — Przegląd typowych problemów projektowych i niezalecanych praktyk w tworzeniu oprogramowania.
6. Refaktoryzacja z wykorzystaniem wzorców projektowych
   - — Wprowadzenie do procesu poprawy struktury kodu bez zmiany jego funkcjonalności.
   - — Praktyczne przykłady refaktoryzacji:
   - — Zmiana nazw zmiennych, metod i klas dla poprawy czytelności,
   - — Wydzielanie metod i zmiennych dla lepszej organizacji,
   - — Lepsze odzwierciedlenie logiki algorytmu w kodzie,
   - — Planowanie i wdrażanie wzorców projektowych w już istniejącym kodzie.

Dla grup na zamówienie, szkolenie możemy realizować w odniesieniu do różnych języków programowania, np. Java, C#, C++, Python.

Przeznaczenie i wymagania

Zajęcia skierowane są do programistów na poziomie mid / senior, developerów, którzy chcą podnieść jakość swojego kodu i lepiej projektować systemy. Zapraszamy również osoby przygotowujące się do roli architekta oprogramowania lub zainteresowane pogłębieniem wiedzy o dobrych praktykach programistycznych.

Mile widziana podstawowa znajomość programowania obiektowego (OOP); umiejętność pracy w jednym z popularnych języków programowania (np. Java, C#, C++, Python); znajomość takich pojęć jak klasy, dziedziczenie, polimorfizm.

Certyfikaty

Uczestnicy szkolenia otrzymują imienne certyfikaty sygnowane przez ALX.

Lokalizacje

- — Warszawa – ul. Jasna 14/16A
- — Zdalnie – zajęcia realizowane poprzez platformę Zoom
- — Kraków – ul. św. Filipa 23
- — Warsaw (English) – Jasna 14/16A
- — Online (English) – your home, office or wherever you want

Zapytaj o szczegóły
tel. 22 63 64 164
akademia@alx.pl

— na życzenie dowolne miejsce w Polsce, lub UE (zajęcia prowadzone w języku angielskim)

Cena szkolenia

2490 PLN netto (VAT 23%)

W cenę szkoleń organizowanych w naszej siedzibie wliczone są:

— autorskie materiały szkoleniowe,
— indywidualne stanowisko komputerowe do pracy podczas zajęć,
— certyfikaty ukończenia szkolenia,
— drobny poczęstunek oraz ciepłe i zimne napoje,
— możliwość jednorazowego kontaktu z instruktorem (instruktorami) po szkoleniu i zadawania pytań dotyczących materiału szkolenia.

Cena szkolenia nie zawiera obiadów. Można je dokupić w cenie 35 zł netto za obiad.

Zapytaj o szczegóły
tel. 22 63 64 164
akademia@alx.pl