WYŻSZA SZKOŁA HANDLOWA W RADOMIU

RADOM ACADEMY OF ECONOMICS

Wyższa Szkoła Handlowa w Radomiu

Zaawansowane metody programowania obiektowego
Laboratorium 1

Radom 2021/2022

1. Cel zadania

Niemal każdy tworzony obecnie projekt informatyczny tworzony jest wedle jednego z wzorców projektowych oraz z wykorzystaniem odpowiednich narzędzi ułatwiających współpracę grupową. Niniejsze laboratorium ma zapoznać z tymi technikami.

2. Potrzebne narzędzia.

- Windows 10/11, najlepiej wersja Pro (aczkolwiek może być także Home)
- Microsoft Visual Studio minimum w wersji Community 2019
- Zalecane konto na github.com lub gitlab.com (aczkolwiek można je założyć w trakcie laboratorium)

3. Informacje wstępne

Obecnie większość aplikacji użytkowych tworzonych jest w językach obiektowych – a przynajmniej w językach, które mają właściwości języków obiektowych. Programowanie obiektowe ma swoje zalety i wady.

Do zalet zalicza się przeważnie czytelność kodu, możliwość wielokrotnego wykorzystania tej samej funkcjonalności w danym projekcie (bądź w różnych projektach), możliwość rozwijania poszczególnych funkcjonalności przez różne osoby (specjalizujące się w danej dziedzinie) czy wreszcie łatwiejsze dostosowywanie kodu do pożądanej funkcjonalności (ujednolicanie, przeciążanie itp.).

Do wad zaś można zaliczyć znaczną ilość nadmiarowego kodu (przy tworzeniu uniwersalnych elementów), znaczną komplikację struktury projektu (w przypadku dużych projektów może to być nawet kilkadziesiąt-kilkaset plików), możliwa duplikacja działań niektórych funkcjonalności (w przypadku dużych projektów taka możliwość znacznie wzrasta).

Celem eliminacji ewentualnych mankamentów programowania obiektowego stosuje się odpowiednie metody i technologie utrzymania projektu w jak najprostszej i czytelnej formie – nawet w przypadku jego realnego skomplikowania.

Pierwszym z etapów na poprawę czytelności projektu jest wykorzystanie tzw. Paradygmatów programowania (wzorce programistyczne). Dzięki utrzymywaniu kodu w określonej notacji (funkcyjnej, obiektowej, modułowej itp.) odnalezienie się w kodzie i jego ewentualna modyfikacja stanie się łatwiejsza zarówno dla twórcy rozwiązania, jak i osób z nim współpracujących.

Kolejnym etapem jest wykorzystanie wzorców architektonicznych. W przypadku Visual Studio i tworzeniu programów obiektowo najlepiej spisuje się wzorzec MVC, aczkolwiek można wykorzystać jego pochodne i/lub inne wzorce projektowe. Dlaczego MVC najlepiej spisuje się w roli wzorca dla chociażby języka C#? Ponieważ pozwalana na skonkretyzowanie modelu (tzw. back-end – logika aplikacji), oddzielonego od interfejsu (View) użytkownika (graficznego lub konsolowego, tzw. front-end – panel użytkownika). Oba te elementy łączone są przez kontroler (Controller) - klasę/klasy przekazujące informacje od funkcji użytkowej do logicznej. Dzięki takiemu podejściu kod może być pisany przez różne niezależne zespoły, a i tak ostatecznie będzie możliwe jego złączenie w całość bez ingerencji w jedną bądź drugą część projektu.

Ostatnią niedogodność programowania obiektowego można wyeliminować wykorzystując szereg odpowiednich narzędzi. Pierwsze z nich to odpowiedni edytor kodu/środowisko programistyczne, które pokaże nam w czytelny i intuicyjny sposób kod, pozwoli na wyświetlanie kodu w spersonalizowany sposób (np. same nagłówki funkcji/metod, powiązania zmiennych z miejscami ich użycia) oraz pozwoli na prześledzenie wykonywania się kodu i ukaże stan programu w dowolnym momencie.

Kolejnym narzędziem będzie aplikacja do tworzenia repozytorium projektu. Klasycznie nasz projekt tworzony jest w odpowiednim katalogu, który dostępny jest na naszym magazynie danych.

Jeżeli chcemy go przenieść/dać komuś kopię jesteśmy zmuszeni kopiować cały katalog projektu, a co najmniej pliki z kodem źródłowym. Ponadto przy dużych projektach zmiany dokonane w kodzie mogą dopiero po jakimś czasie wykazać nieprawidłowości w funkcjonowaniu – wszystko zależy od czasu dokonywania testów aplikacji. W tym wypadku pomóc może skorzystanie z repozytorium kodu – SVN lub git.

Obecnie najlepszym wyborem jest git. Narzędzie to stosowane jest niemal w każdej firmie zajmującej się projektami – niekoniecznie programistycznymi. Pozwala bowiem na śledzenie niemal każdej zmiany w edytowanych plikach – zapisuje każdą nową wersję pliku jako nowy, osobny plik lub jeżeli mamy do czynienia z plikiem kodu źródłowego to zapis następuje tylko dla zmienianych linii kodu (nadpis różnicowy). Dzięki temu, jeżeli wcześniejsza wersja kodu spełniała nasze oczekiwania (a aktualna zawiera błędy) bez problemu będziemy mogli cofnąć się do właściwej wersji kodu by ponownie rozpocząć modyfikację (lub będziemy w stanie skuteczniej ustalić źródło ewentualnego błędu).

Git jest też narzędziem do pracy grupowej. Każda naniesiona zmiana przypisywana jest osobie, która ją naniosła. Bez problemu można więc ustalić autora określonych zmian w kodzie. To zaś pozwala na ewentualne szybsze przydzielanie poprawek w kodzie określonym ludziom w zespole.

4. Przebieg.

Zadaniem będzie utworzenie dowolnego projektu oprogramowania z wykorzystaniem wszystkich opisanych wyżej rozwiązań i narzędzi. Projektem może być dowolna aplikacja napisana w języku C#, która będzie spełniania następujące kryteria:

- Będzie posiadała co najmniej dwie pojedyncze funkcjonalności wielokrotnego użytku
- Będzie posiadała co najmniej jeden rodzaj interfejsu dla użytkownika
- Będzie pozwalała na zmianę interfejsu użytkownika na dowolny inny (bądź na interfejs mieszany/wiele interfejsów użytkownika)

Można wykorzystać dowolny, własny pomysł na projekt. Przykładowe projekty spełniające powyższe wymagania:

- Program liczenia obwodów i pól figur geometrycznych oraz pozwalający na wyliczenia dodatkowych informacji - np. długości przeciwprostokątnych lub promień kątów występujących w figurze.
- Program do zbierania danych na temat szkoleń przeprowadzanych zdalnie; minimum podstawowe dane uczestnika (imię, nazwisko, adres zamieszkania) oraz dane dotyczące kursu (nazwa, data rozpoczęcia, czas trwania w dniach, opis kursu)
- Program do katalogowania posiadanych gier/książek/filmów; należy uwzględnić takie dane jak informacje o przechowywanym zasobie (nazwa, data wydania, nośnik przechowujący, opis) oraz o sposobie przechowywania (numer ewidencyjny, miejsce składowania, ewentualne pożyczenie, czas zwrotu)

Wymagania spełni także dowolna aplikacja klient-serwer (np. prosty komunikator sieciowy czy serwer plików) jednak zadanie to może być nieco trudniejsze (dodatkowa komunikacja, obsługa błędów przesyłania itp.).

Po wybraniu typu aplikacji w następnej kolejności należy utworzyć projekt aplikacji wykorzystując język C#. Sugerowane jest by wybrać aplikację konsolową (z menu tekstowym), aplikację WPF lub aplikację Web. Nie jest rekomendowane wybieranie aplikacji WinAPI (aplikacje te są rozpatrywane jako przestarzałe). Aplikacja ma spełaniać swoją funkcjonalność i działać offline. Istotne jest, by aplikacja miała architekturę MVC (Model-View-Controller), gdyż dzięki temu rozwiązaniu znacznie łatwiej będzie zamienić i modyfikować poszczególne elementy projektu (niezbędne do laboratorium drugiego).

Na koniec projekt, wraz z odpowiednimi komentarzami, należy ulokować w repozytorium git (na stronie github, gitlab, gitbucket lub równoważnym). Nie trzeba wykonywać sprawozdania! Wystarczy odpowiedni plik readme + komentarze w kodzie.

5. Zakończenie

Odnośnik do repozytorium proszę przesłać na adres piotr_dobosz@int.pl, w temacie wiadomości zawierając [WSH_ZMPO].
Jeżeli ktoś będzie miał problem z kodowaniem - również należy udostępnić prowadzącemu link do repozytorium (celem wglądu w projekt). Wszelkie problemy z pracą można omówić w dowolnym czasie na Teams: piotr.dobosz@wsh.pl