Grzegorz Bielski - mail: gbielski@wsb.gda.pl

Zaawansowane programowanie obiektowe

Manage Employees Project

Zasady zaliczenia

1. Obecność na zajęciach jest obowiązkowa i wpływa na ocenę końcową.
2. Każdą nieobecność należy usprawiedliwić i nadrobić (wykonać zadania z danego tematu).
3. Jeśli student nie zdążył skończyć zadań z danego tematu na zajęciach, ma obowiązek ukończyć je w domu.
4. Zajęcia zaczynają się omówieniem kodu napisanego na poprzednich zajęciach.
5. Na początku zajęć prowadzący wybiera osobę, która zaprezentuje swój kod. Omówienie kodu nie jest na ocenę i student nie jest oceniany. Na tym etapie wolno popełniać błędy - po to są zajęcia, aby omawiać błędy i się uczyć. Omówione błędy należy poprawić i przepisać swój kod.
6. Do zaliczenia przedmiotu wymagana jest znajomość zasad SOLID. Po omówieniu kodu, prowadzący pyta dowolną osobę o zasady SOLID. Jeśli student nie umie odpowiedzieć, oznacza to obniżenie oceny końcowej o 1. Student może cofnąć obniżenie oceny, zgłaszając się na następnych zajęciach do omówienia zasad SOLID.
7. Brak kodu z poprzednich zajęć oznacza obniżenie oceny końcowej o 1.
8. Ocena końcowa będzie wystawiana na podstawie końcowego kodu prezentowanego przez studenta w połączeniu z obecnościami i ewentualnymi brakami z pkt. 6.

SOLID

S: **Zasada pojedynczej odpowiedzialności** (ang. single responsibility principle) – każda klasa powinna być odpowiedzialna za jedną konkretną rzecz.

O: **Zasada otwarty/zamknięty** (ang. open/close principle) – każda klasa powinna być otwarta na rozbudowę ale zamknięta na modyfikacje.

L: **Zasada podstawienia Liskov** (ang. liskov substitution principle) – w miejscu klasy bazowej można użyć dowolnej klasy pochodnej (zgodność wszystkich metod).

I: **Zasada segregacji interfejsów** (ang. interface segregation principle) interfejsy powinny być małe i konkretne, aby klasy nie implementowały metod, których nie potrzebują.

D: **Zasada odwrócenia odpowiedzialności** (ang. dependency inversion principle) – wszystkie zależności powinny w jak największym stopniu zależeć od abstrakcji a nie od konkretnego typu.

Zobacz więcej: https://www.p-programowanie.pl/paradygmaty-programowania/zasady-solid/

Temat 1: Przegląd podstawowej składni programowania obiektowego

[blok 1 i 2]

Zadania

Zadanie 1. Zastanów się, z jakich klas powinna składać się aplikacja służąca zarządzaniu danymi o pracownikach i pozwalająca na monitorowanie operacji finansowych, jak wypłata za nadgodziny czy świadczenia socjalne. Zadeklaruj stosowne klasy.

Zadanie 2. Do klas z wcześniejszego zadania Person, Employee oraz Operation dodaj jako prywatne pola służące przechowywaniu danych ich reprezentujących. Dodaj pole HolidayBonus jako statyczne (czyt. określone dla klasy a nie obiektu) z zainicjalizowaną wartością jako 1000. Powiąż klasę Employee z Operation dla przechowywania listy operacji.

Zadanie 3. Dla klasy Employee dodaj strukturę opisującą jego wynagrodzenie (płaca zasadnicza, dodatek stażowy, inne) czyli Wage (basic, bonus, other). Połącz strukturę z klasą Employee poprzez odpowiednie pole.

Zadanie 4. Dla klasy Employee dodaj pole o nazwie Umowa (Contract). Pole powinno pozwolić wyłącznie na wartości: FullTime, PartTime, Contract.

Zadanie 5. Dodaj metody dla klas:

- Person – pozwalające na wypisanie informacji o osobie, modyfikację danych;
- Employee – umożliwiające pobranie i modyfikację danych o pracowniku, zmianę wysokości dodatku wakacyjnego, modyfikację informacji o składnikach pensji, wykonanie operacji (np. wypłata wynagrodzenia);
- Operation – zwrócenie informacji o operacjach finansowych (np. wypłata zaliczki).

Zadanie 6. Zdefiniowane klasy umieść w przestrzeni nazw Employees, znajdującej się w przestrzeni o nazwie Finances.

Zadanie 7. Przygotuj klasy jako bibliotekę dll o:

- nazwie Company.dll;
- domyślnej przestrzeni nazw Finances;

Zadanie 8. Zaimportuj bibliotekę do projektu Manage Employees tak, aby możliwe było jej zastosowanie. Przetestuj działanie biblioteki.

Zadanie 9. W związku z tym, że pisanie kodu tworzenia obiektów jest zbyt długie i może powodować nieodpowiednią ich inicjalizację, niezbędne jest zastosowanie odpowiednich metod zarządzania tworzeniem obiektów - konstruktorów. Dlatego dla wszystkich klas biblioteki napisz stosowane konstruktory, biorące pod uwagę różne możliwości tworzenia

obiektów klas Person, Employee i Operation.

Zadanie 10. Zdefiniuj klasę o nazwie Client na podstawie Person przechowującą informację o poszczególnych klientach. Następnie zdefiniuj klasę Manager na podstawie Employee odpowiedzialną za dostarczenie funkcjonalności zarządzania informacjami o poszczególnych menadżerach.

Pytania kontrolne

1. Wymień podstawowe składowe budowy klas.
2. W jaki sposób deklarujemy obiekty klas?
3. Jaka jest procedura publikowania i zaimportowania obiektowej biblioteki?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016
2. Włodarczyk M., ITA-105 Programowanie obiektowe
3. Programowanie zorientowane obiektowo (C# i Visual Basic), https://msdn.microsoft.com/pl-pl/library/dd460654(v=vs.110).aspx

Temat 2: Właściwości

[blok 3]

Zadania

Zadanie 1. Dla klas Person, Employee i Operation zgodnie z uznaniem przypisz właściwości pozwalające na pobieranie i modyfikację wartości pól (np. Name, Surname). Dla pola Wage klasy Employee za pomocą właściwości zaimplementuj mechanizm autoryzacji – konieczność podania loginu i hasła.

Pytania kontrolne

1. Jakie są korzyści stosowania właściwości?
2. Czym różnią się właściwości od metod akcesoriów?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016
2. Programowanie zorientowane obiektowo (C# i Visual Basic), https://msdn.microsoft.com/pl-pl/library/dd460654(v=vs.110).aspx

Temat 3: Kolekcje generyczne

[blok 3]

Zadania

Zadanie 1. Przekształć dla klasy Employee pole Operations zdefiniowane jako tablica klasy Operation na kolekcję, najpierw w postaci kolekcji zwykłej (List), a następnie generycznej typu Operation. Zdefiniuj klasę Employees, pozwalającą na przechowywanie poprzez kolekcje listy pracowników.

Pytania kontrolne

1. Jakie są korzyści stosowania kolekcji generycznych względem standardowych?
2. Wymień min. 3 klasy dla tworzenia kolekcji generycznych.

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016

Temat 4: Indeksatory

[blok 4]

Zadania

Zadanie 1. Zdefiniuj dla klasy Employees indeks, pozwalający na pobranie obiektu pracownika poprzez podanie imienia i nazwiska.

Pytania kontrolne

1. Jaka jest podstawowa korzyść stosowania indeksatorów w klasie?
2. W jaki sposób można zdefiniować wiele indeksów w klasie o tej samej nazwie?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016

Temat 5: Fabryki obiektów

[blok 4]

Zadania

Zadanie 1. Zaimplementuj dla klasy Employee mechanizm factory dla dodawania nowych obiektów klasy Employee – czyt. dodaj metodę statyczną pozwalającą na utworzenie obiektu klasy Employee.

Zadanie 2. Dodaj do klasy Employees metodę pozwalającą na stworzenie pracownika i dodanie go do listy.

Zadanie 3. Dodaj do klasy Employees metody pozwalające na dodawanie, usuwanie oraz pobieranie obiektów klasy Employee. Dla dodawania i usuwania dodaj walidację, że dany obiekt istnieje. Nie dodawaj już istniejących obiektów w liście.

Pytania kontrolne

1. Jaki jest cel tworzenia metod, których celem jest zwracanie obiektów?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016

Temat 6: Klasy abstrakcyjne

[blok 5]

Zadania

Zadanie 1. Zmień definicję klasy Person na abstrakcyjną (ang. abstract). Jaki jest tego efekt? Czy możesz utworzyć teraz obiekt klasy Person?

Zadanie 2. Dla klasy Person dodaj metodę o nazwie Interests (string Interest), gdzie chcesz ująć wyłącznie jej deklarację, a nie definicję. Czy jest to możliwe? Jaki jest wpływ zmian dla klasy Person na inne klasy?

Pytania kontrolne

1. Czy można tworzyć obiekty dla klasy abstrakcyjnej?
2. Jaka jest przyczyna tworzenia klas abstrakcyjnych?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016

Temat 7: Przeciążanie operatorów

[blok 5]

Zadania

Zadanie 1. Dla klasy Employee przeciąż domyślne działanie operatora == pozwalające na porównanie dwóch pracowników, jako porównanie ich imion i nazwisk.
Zadanie 2. Dla klasy Employee przeciąż domyślne działanie operatorów < i > pozwalające na porównanie dwóch pracowników, jako porównanie ich wynagrodzeń.
Zadanie 3. Zdefiniuj operator konwersji z typu Employee na double, powodujący zwrócenie jako wyniku wysokości pensji.
Zadanie 4. Dla klasy Employee przeciąż domyślne działanie operatora + jako pozwalające na dodanie do pracownika liczby. Jako wynik zwracana jest suma pensji pracownika i podanej liczby.

Pytania kontrolne

1. Czy można przeciążyć operator „>” bez przeciążania operatora „<” ?
2. Których operatorów nie można przeciążać?
3. Jaki modyfikator musi być zastosowany dla metod służących przeciążaniu operatorów?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016

Temat 8: Delegacje

[blok 6]

Zadania

Zadanie 1. Zdefiniuj w klasie Employee delegację pozwalającą podczepiać metody wywoływane, jeśli zmieni się imię lub nazwisko pracownika. Napisz kod wywołujący delegację, jeśli nastąpi zmiana imienia lub nazwiska.
Zadanie 2. Zdefiniuj metodę wypisującą, że nastąpiła zmiana nazwiska lub imienia. Przypisz ją do delegacji. Sprawdź czy delegacja działa, zmieniając dla pracownika najpierw imię o następnie nazwisko.
Zadanie 3. Zmodyfikuj kod, aby stosował delegacje oparte o klasy Action czy Func.

Pytania kontrolne

1. Jaki jest cel definiowania delegacji w klasach?
2. Czy delegacje są zwracane przez wartość czy przez referencję?
3. Czy typ delegacyjny może mieć określony zwracany typ?
4. Jakie zastosowanie mają klasy Action i Func?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016
2. Delegat Action<T>, https://msdn.microsoft.com/pl-pl/library/018hxwa8(v=vs.110).aspx
3. Delegat Func<T, TResult>, Delegat Func<T, TResult>, https://msdn.microsoft.com/pl-pl/library/bb549151(v=vs.110).aspx

Temat 9: Zdarzenia

[blok 6]

Zadania

Zadanie 1. Zdefiniuj w klasie Employee zdarzenie powodujące powiadamianie wszystkich subskrybentów o zmianie wysokości pensji, zwracające wartość oryginalną i nową pensji.
Zadanie 2. Zdefiniuj w klasie Employee zdarzenie, powodujące powiadamianie wszystkich subskrybentów o zmianie formy zatrudnienia.
Zadanie 3. Zmodyfikuj kod, aby zastosowana została klasa EventHandler.

Pytania kontrolne

1. Jaka jest różnica pomiędzy delegacjami a zdarzeniami?
2. Jak powinna wyglądać sygnatura delegacji dla zdarzenia?
3. Jakie są zalety zastosowania klasy EventHandler?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016
2. Delegat EventHandler, https://msdn.microsoft.com/pl-pl/library/system.eventhandler(v=vs.110).aspx

Temat 10: Wyrażenia Lambda

[blok 6]

Zadania

Zadanie 1. Zamień kod w aplikacji na stosujący wyrażenia lambda dla obsługi zdefiniowanych w temacie 9 zdarzeń.
Zadanie 2. Zamień kod w aplikacji stosujący wyrażenia lambda, aby były one wywoływane jako asynchroniczne lambda.

Pytania kontrolne

1. W jaki sposób dla obsługi zdarzenia można zastosować wyrażenia lambda?
2. Jakie korzyści daje zastosowanie asynchronicznych wyrażeń lambda?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016
2. Lambda Expressions, https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions

Temat 11: Przetwarzanie danych za pomocą Linq to Object

[blok 7]

Zadania

Za pomocą języka LINQ:
Zadanie 1. Posortuj alfabetycznie wg nazwisk i imion dane pracowników.
Zadanie 2. Wypisz wyłącznie imiona i nazwiska pracowników.
Zadanie 3. Wypisz nazwiska pracowników na literę P.
Zadanie 4. Wypisz wszystkich pracowników których pensja podstawowa jest większa niż 5000.
Zadanie 5. Wyświetl łączną kwotę wynagrodzeń wszystkich pracowników.
Zadanie 6. Wyświetl łączną kwotę wynagrodzeń poszczególnych pracowników, sortując po niej malejąco.

Pytania kontrolne

1. Za pomocą jakich metod w języku LINQ można sortować i filtrować dane?
2. Jakie typy danych mogą być przetwarzane za pomocą języka LINQ?

Literatura

1. Albahari J., Albahari B., C# 6.0 w pigułce. Wydanie VI, 2016
2. Introduction to LINQ Queries (C#), https://msdn.microsoft.com/en-us/library/bb397906.aspx
3. GetAsyncKeyState function, https://msdn.microsoft.com/en-us/library/bb397927.aspx