Państwowa Wyższa Szkoła Zawodowa w Nowym Sączu

Karta przedmiotu

obowiązuje studentów rozpoczynających studia w roku akademickim 2013/2014

Instytut Techniczny

Kierunek studiów: Informatyka — Profil: Ogólnoakademicki

Forma studiów: Niestacjonarne — Kod kierunku: 11.3

Stopień studiów: I

Specjalności: Informatyka stosowana

1 Przedmiot

Nazwa przedmiotu	Metodyka i techniki zaawansowanego programowania
Kod przedmiotu	IT 11.3 AIN C9 13/14
Kategoria przedmiotu	Przedmioty specjalnościowe
Liczba punktów ECTS	3
Semestry	6

2 Rodzaj zajęć, liczba godzin w planie studiów

Semestr	wykład	ćwiczenia	laboratorium	projekt	seminarium
6	8			15	

3 Cele przedmiotu

Cel 1 Przegląd zasad projektowania i programowania obiektowego, w oparciu o studiowanie wybranych wzorców projektowych. Realizacja abstrakcji, hermetyzacji i hierarchii danych na przykładach optymalnych rozwiązań danego problemu, w oparciu o notację UML, a następnie implementacja w językach C++ i Java.

Cel 2 Programowanie uogólnione - przegląd zasad programowania z użyciem szablonów. Kontenery i iteratory. Funktory oraz wyrażenia lambda jako efektywne narzędzia programowania w połączeniu z algorytmami w języku C++. Kolekcje i algorytmy w języku Java.

Cel 3 Programowanie wielowątkowe w nowym standardzie C++ (implementacja w oparciu o bibliotekę justthread). Programowanie wielowątkowe w Javie.

Cel 4 Metody pracy z kodem - narzędzia programistyczne. Szukanie błędów, testowanie kodu, kwestia wycieków pamięci a inteligentne obiekty (porównanie C++ i Java). Optymalizacja kodu i programu (profilowanie). Pisanie testów kodu. Symulacje rzeczywistych procesów.

4 Wymagania wstępne w zakresie wiedzy, umiejętności i innych kompetencji

a Student powinien posiadać ugruntowaną wiedzę i doświadczenie na temat programowania w językach C++ oraz Java, w zakresie obejmowanym przez kursy poprzedzające zajęcia z metodyki i technik zaawansowanego programowania.

5 Efekty kształcenia

EK1 Wiedza: Student dobiera formę programowania do problemu. Student zna zasady dobrego projektowania obiektowego. Potrafi przeprowadzić analizę problemu oraz wybrać optymalny model rozwiązania, w oparciu o wzorce projektowe. Objaśnia rozwiązanie przy użyciu diagramów UML. Następnie wdraża projekt w jednym z języków programowania (C++, Java).

EK2 Umiejętności: Potrafi napisać kod w postaci klas i funkcji szablonów. Użytkuje kontenery standardowe oraz iteratory. Stosuje algorytmy biblioteki standardowej C++, pisząc dla nich obiekty funkcyjne lub wyrażenia lambda. Podobnie w języku Java zna kolekcje oraz algorytmy.

EK3 Umiejętności: Student zna problematykę programowania wielowątkowego. Potrafi wykorzystywać wątki, operacje asynchroniczne oraz atomowe. Rozwiązuje problemy wyścigu danych. Realizuje to w pełni przenośnym kodzie według standardu języka C++11 oraz specyfikacji Java.

EK4 Umiejętności: Student potrafi napisać testy kodu (z użyciem zewnętrznych bibliotek np. CPPUnit, Boost). Obsługuje programy debuggery oraz profilowanie programu w celu optymalizacji jego działania. Potrafi przygotować symulację Monte Carlo modelującą określony problem inżynierski.

EK5 Kompetencje społeczne: Student potrafi przeprowadzić wywiad z klientem skutecznie pozyskując informacje na temat zamówionego projektu. Jest zdolny do takich zapytań, które sugerują optymalne rozwiązanie, ale uważnie i z dbałością podchodzi do stawianych wymagań. Postępuje zgodnie z regułami efektywnego projektowania i programowania.

6 Treści programowe

wykład

Lp	Tematyka zajęć Opis szczegółowy bloków tematycznych	Liczba godzin
W1	Repetytorium z zasad dobrego projektowania obiektowego. Tworzenie typów użytkownika, hermetyzacja danych i tworzenie hierarchii. Realizacja obiektowych relacji w językach C++ i Java. Wzorce projektowe - klasyfikacja (wzorce kreacyjne, strukturalne i czynnościowe), charakterystyka. Omówienie wybranych wzorców (singleton fabryka abstrakcyjna, budowniczy, adapter, fasada, dekorator, most, iterator, metoda szablonowa, obserwator, stan i strategia). Złożone wzorce projektowe (MVC - model, widok, kontroler). Zasady implementacji wzorców w językach C++ i Java.	2
W2	Kontenery i kolekcje. Ich charakterystyka, cele funkcjonalne, sposoby obsługi (C++, Java). Mechanika iteratorów. Wprowadzenie do algorytmów, przegląd możliwości oferowanych przez biblioteki języków. Szablony, typy generyczne. Klasy i metody szablonowe. Składnia w językach C++ i Java. Specjalizacje typów. Dedukcje typów. Tworzenie i używanie obiektów wywoływalnych. Obiekty funkcyjne, predykaty, wyrażenia lambda. Adaptowanie funkcji i metod składowych w charakterze funktorów. Sprzężenie z algorytmami.	3

WYKŁAD

Lp	Tematyka zajęć Opis szczegółowy bloków tematycznych	Liczba godzin
W3	Programowanie wielowątkowe: procesy atomowe, wątki, programowanie asynchroniczne. Rozwiązywanie kolizji. Mechanizmy dostępne w standardzie C++11 oraz języku Java.	2
W4	Pisanie testów kodu. Metody szukania błędów (debuger, kontrolowanie wycieków pamięci w C++). Profilowanie i optymalizacja programu. Podstawy symulacji Monte Carlo procesów spotykanych w zagadnieniach inżynierskich.	1
	Razem	**8**

PROJEKT

Lp	Tematyka zajęć Opis szczegółowy bloków tematycznych	Liczba godzin
P1	Przykład obiektowego rozwiązania projektu zarządzania osobami na wyższej uczelni. Zależności typu dziedziczenie kontra zawieranie. Klasy mieszane, role. Implementacja w językach C++ i Java.	2
P2	Wzorzec obserwator. Realizacja śledzenia stanu obiektu, powiadamiającego o zmianie stanu, poprzez dowolną i dynamicznie zmieniającą się listę obserwatorów. Implementacja w języku C++ lub Java.	2
P3	Fabryka abstrakcyjna. Produkcja specjalizowanych obiektów na życzenie, poprzez ogólny interfejs.	2
P4	Wzorce złożone - zastosowanie wzorca MVC (model - widok - kontroler) w działaniu interfejsów graficznych.	2
P5	Przykłady zastosowań kontenerów, ich własności, cechy, wydajność oraz sposoby obsługi. Kontenery w nowym standardzie C++11, kolekcje w Java.	2
P6	Zastosowania algorytmów w połączeniu z pisaniem obiektów funkcyjnych oraz wyrażeń lambda. Efektywne rozwiązania w kompaktowej postaci sprawdzonego kodu.	2
P7	Programowanie wielowątkowe - tworzenie i obsługa wątków. Muteksy i komunikacja (promise / future). Programowanie asynchroniczne. Operacje atomowe.	2
P8	Metodologia pisania testów kodu z użyciem asercji, bibliotek zewnętrznych testów. Szukanie błędów (debuger), lokalizowanie wycieków pamięci (C++). Optymalizacja kodu poprzez analizę (profilowania) wykonawczą. Zasady pisania symulacji Monte Carlo dla procesów inżynierskich.	1
	Razem	**15**

7 Metody dydaktyczne

M1 Wykłady

M2 Prezentacje multimedialne

M3 Ćwiczenia projektowe

M4 Studium przypadku

8 Obciążenie pracą studenta

Forma aktywności	Średnia liczba godzin na zrealizowanie aktywności
Godziny kontaktowe z nauczycielem akademickim, w tym:	
Godziny wynikające z planu studiów	23
Konsultacje przedmiotowe	1
Egzaminy i zaliczenia w sesji	0
Godziny bez udziału nauczyciela akademickiego wynikające z nakładu pracy studenta, w tym:	
Przygotowanie się do zajęć, w tym studiowanie zalecanej literatury	31
Opracowanie wyników	0
Przygotowanie raportu, projektu, prezentacji, dyskusji	20
Sumaryczna liczba godzin dla przedmiotu wynikająca z całego nakładu pracy studenta	**75**
Sumaryczna liczba punktów ECTS dla przedmiotu	3

9 Sposoby oceny

Przedmiot ma za zadanie wyrobienie zdolności w wyborze właściwych metod programowania w rozwiązywaniu różnych problemów spotykanych w pracy inżyniera (programowanie obiektowe, uogólnione, programowanie wielowątkowe, testowanie kodu i identyfikacja błędów).

Ocena formująca

F1 Ćwiczenie praktyczne

F2 Kolokwium

Kryteria oceny

Efekt kształcenia 1		Miejsce weryfikacji	Opis weryfikacji EK 1
Na ocenę 3	Student potrafi, za pomocą notacji UML, przedstawić zasady projektowania obiektowego. Zna implementację obiektowych relacji i mechanizmów w językach C++ i Java. Orientuje się we wzorcach projektowych na tyle, że potrafi po nie sięgnąć w celu rozwiązania konkretnego zagadnienia.	wykład, projekt	Projektowanie, zapis UML oraz dywersyfikacja rozwiązań dla konkretnych problemów obiektowych. Dyskusja nad optymalnymi rozwiązaniami (wzorcami) oraz ich programowana implementacja.
Na ocenę 4	Student pewnie orientuje się w zasadach obiektowego projektowania i programowania. Potrafi przedstawić najważniejsze wzorce wraz z rysunkiem oraz komentarzem, jakie zagadnienia można za ich pomocą rozwiązywać.		
Na ocenę 5	Student potrafi przeprowadzić dyskusję na temat alternatywnych rozwiązań w programowaniu obiektowym (relacje dziedziczenia, relacje zawierania, relacje mieszane) wraz z zaletami i wadami tych rozwiązań. Potrafi wybrać i omówić wzorzec projektowy jako optymalne rozwiązanie postawionego problemu.		

Efekt kształcenia 2		Miejsce weryfikacji	Opis weryfikacji EK 2
Na ocenę 3	Student potrafi, wychodząc z implementacji dla konkretnego typu, zapisać implementację generyczną. Potrafi korzystać z funktorów oraz samemu pisać proste obiekty funkcyjne i stosować je w połączeniu z algorytmami i kontenerami. Potrafi też używać prostych wyrażeń lambda.	projekt	Wdrażanie, na przykładzie prostych ćwiczeń, prostych, a potem bardziej złożonych algorytmów, z naciskiem na efektywne używanie bibliotek języków (w najnowszych standardach / wersjach). Monitorowanie postępów studenta podczas rozwiązywania takich zagadnień.
Na ocenę 4	Student pewnie orientuje się w algorytmach (języki C++ i Java) dostarczanych z bibliotekami. Potrafi je wykorzystywać w połączeniu z kontenerami oraz potrafi różnego rodzaju obiekty wywoływalne użyć w algorytmie, w tym wyrażenia lambda.		
Na ocenę 5	Student potrafi napisać kod uogólniony i efektywnie stosować algorytmy. Zna i stosuje kontenery oraz iteratory. Potrafi pisać funktory, obiekty funkcyjne, adaptować funkcje i metody jako obiekty funkcyjne. Potrafi w algorytmach również zastosować wyrażenia lambda.		
Efekt kształcenia 3		Miejsce weryfikacji	Opis weryfikacji EK 3
Na ocenę 3	Student potrafi zaplanować i zapisać podstawowe schematy wielowątkowe. Zna zasady tworzenia i kontrolowania wątków oraz problematykę współdzielenia danych przez różne wątki.	projekt	Zdefiniowanie problemu, który może być skutecznie rozwiązany poprzez rozbicie na wiele wątków. Dyskusja nad wydajnością konkretnego sprzętu a ilością używanych wątków. Implementacja przykładu, monitorowana przez prowadzącego.
Na ocenę 4	Student potrafi tworzyć i kontrolować wątki (różne rodzaje mutexów). Rozumie co to są operacje atomowe i jakie z nich wypływają korzyści.		
Na ocenę 5	Student potrafi rozwiązywać zagadnienia programowania wielowątkowego nie tylko poprzez tworzenie wątków i odpowiednie blokowanie dostępu do zasobów współdzielonych, ale również poprzez programowanie asynchroniczne i wymianę zasobów pomiędzy wątkami.		
Efekt kształcenia 4		Miejsce weryfikacji	Opis weryfikacji EK 4
Na ocenę 3	Student potrafi metodologicznie identyfikować błędy wykonawcze w kodzie. Rozumie, na czym polega testowanie kodu oraz jego optymalizacja. Zna zasady wykonywania symulacji i interpretacji wyników.	projekt	W oparciu o przykładowy, błędny kod, poznanie zasad monitorowania i wykrywania błędów. Rozwiązywanie i usuwanie z kodu błędów składniowych, logicznych, pamięciowych - w pracy indywidualnej każdego studenta.
Na ocenę 4	Student używa różnych narzędzi do śledzenia i szukania błędów. Potrafi pisać testy z użyciem zewnętrznych bibliotek. Potrafi wykonywać symulacje procesów i interpretować ich wyniki.		

Na ocenę 5	Student pewnie używa narzędzi do śledzenia i optymalizacji kodu (debugery, oprogramowanie śledzące alokacje pamięci, profilujące). Zna zasady pisania testów. Zna i implementuje procesy symulacyjne wraz z właściwą interpretacją wyników.		
Efekt kształcenia 5		Miejsce weryfikacji	Opis weryfikacji EK 5
Na ocenę 3	Student potrafi zaproponować projekt rozwiązujący dane zagadnienie, nie zawsze może przeprowadza wszechstronne studium, ale jego rozwiązanie jest akceptowalne. Potrafi je przedstawić w postaci diagramów UML i wyjaśnić, jak będzie wyglądała dalsza implementacja.	projekt	Studium przypadku. Praca w 2-osobowych grupach, w których symulowane będą warunki klient-programista, wypracowane rozwiązania następnie przedstawiane.
Na ocenę 4	Student rozważa różne techniki programistyczne i potrafi swoje pomysły przedstawić w czytelnych diagramach UML. Jego projekt jest otwarty na rozszerzenia, ale zarazem potrafi zaprojektować stabilny i niezmienny interfejs. Projekt jest zrozumiały dla zleceniodawcy, a czas jego realizacji oraz kolejne etapy wdrażania są przekonując zaplanowane.		
Na ocenę 5	Student potrafi projektować z użyciem synergicznych technik obiektowych jak i z wykorzystaniem metod generycznych. Tam gdzie jest to ekonomicznie i efektywnie uzasadnione, potrafi zaproponować rozwiązania wielowątkowe, w pełni wykorzystujące możliwości współczesnego sprzętu. Projekt dokumentuje i przedstawia za pomocą diagramów UML, również kolejne etapy jego implementacji oraz przewidywany czas i koszty realizacji projektu.		

Ocena do indeksu (ocena podsumowująca)

Ocena jest wypadkową monitorowanych kolokwiami postępów oraz aktywnego udziału w zajęciach warsztatowych.

Warunki zaliczenia przedmiotu

a Zaliczenie przedmiotu na podstawie pozytywnych wyników praktycznych kolokwiów oraz ogólnej aktywności podczas zajęć.

10 Macierz realizacji przedmiotu

Efekty kształcenia dla przedmiotu	Odniesienie do efektów kierunkowych	Cele przedmiotu	Treści programowe	Metody dydaktyczne
EK1	INF_K06, INF_W07, INF_UB07	Cel1	W1, P1, P2, P3, P4	M1, M2, M3, M4
EK2	INF_UB02, INF_K06, INF_W07, INF_UB07	Cel2	W2, P5, P6	M1, M2, M3, M4
EK3	INF_UB02, INF_K06, INF_W07, INF_UB07	Cel3	W3, P7	M1, M2, M3, M4
EK4	INF_UB02, INF_K06, INF_W07, INF_UB07	Cel4	W4, P8	M1, M2, M3, M4
EK5	INF_K06, INF_W07	Cel1, Cel4	W1, W2, W3, W4, P1, P6, P7, P8	M1, M2, M3, M4

11 Wykaz literatury

Literatura podstawowa:

[1] **Nicholas A. Solter, Scott J. Kleper** — *C++ Zaawansowane programowanie*, Warszawa, 2005, Helion

[2] **Bruce Eckel, Chuck Allison** — *Thinking in C++*, Warszawa, 2004, Helion

[3] **Jerzy Grebosz** — *Symfonia C++ Standard*, Kraków, 2010, Editions 2000

[4] **David Vandevoorde, Nicolai M. Josuttis** — *C++. Szablony. Vademecum profesjonalisty*, Warszawa, 2003, Helion

[5] **Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides** — *Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego uzytku*, Warszawa, 2010, Helion

[6] **Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra** — *Wzorce projektowe. Rusz głowa!*, Warszawa, 2010, Helion

[7] **Anthony Williams** — *C++ Concurrency in Action*, Great Britain, 2012, Manning Publications Co.

[8] **Herbert Schildt** — *Java. Kompendium programisty. Wydanie VIII*, Warszawa, 2012, Helion

[9] **Cay S. Horstmann, Gary Cornell** — *Java. Techniki zaawansowane. Wydanie VIII*, Warszawa, 2009, Helion

Literatura uzupełniająca:

[1] **Andrei Alexandrescu** — *Nowoczesne projektowanie w C++. Uogólnione implementacje wzorców projektowych*, Warszawa, 2011, Helion

[2] **Scott Meyers** — *STL w praktyce. 50 sposobów efektywnego wykorzystania*, Warszawa, 2004, Helion

[3] **Marcin Lis** — *Java. Cwiczenia praktyczne. Wydanie III*, Warszawa, 2011, Helion

[4] **Kathy Sierra, Bert Bates** — *Java. Rusz głowa! Wydanie II*, Warszawa, 2010, Helion

12 Informacje o nauczycielach akademickich

Osoba odpowiedzialna za kartę

dr Witold Przygoda (kontakt: witold.przygoda@gmail.com)

Osoby prowadzące przedmiot

dr Witold Przygoda (kontakt: witold.przygoda@gmail.com)

13 Zatwierdzenie karty przedmiotu do realizacji

(miejscowość, data) (odpowiedzialny za przedmiot) (kierownik zakładu) (dyrektor instytutu)

Przyjmuję do realizacji (data i podpisy osób prowadzących przedmiot)

. . . . . .

