KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Programowanie obiektowe [S1Bioinf1>POBKT]

Przedmiot

Kierunek studiów
Bioinformatyka

Rok/Semestr
1/2

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
pierwszego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
30

Laboratorium
30

Inne (np. online)
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

5,00

Koordynatorzy

dr hab. inż. Piotr Łukasiak prof. PP
piotr.lukasiak@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student rozpoczynający ten moduł powinien posiadać podstawową wiedzę z zakresu algorytmów i języków programowania. W zakresie umiejętności wymagana jest biegłość przy rozwiązywaniu podstawowych problemów powiązanych ze specyfikacją algorytmów, samodzielnym pisaniem, modyfikacją i testowaniem programów komputerowych wraz z umiejętnością pozyskiwania informacji ze wskazanych źródeł.

Cel przedmiotu

Celem przedmiotu jest nauczenie zasad tworzenia uniwersalnych modułów programowych zdatnych do wielokrotnego wykorzystania w różnych projektach programistycznych i łatwych w rozwoju i pielęgnacji, poprzez zastosowanie unikatowych rozwiązań algorytmicznych i programistycznych dostępnych w językach obiektowych na przykładzie C++. Dodatkowo celem jest nauczenie studentów tworzenia własnych bogatych semantycznie i uniwersalnych abstrakcyjnych typów danych, jak również rozwinięcie u studentów umiejętności projektowania i tworzenia systemów informatycznych o poprawnej architekturze charakteryzującej się spoistością składowych modułów programowych i luźnych związków między tymi modułami. Istotnym celem przedmiotu jest wykształcenie u studentów umiejętności komunikacji podczas niezależnego tworzenia modułów programów komputerowych oraz wyszukiwanie optymalnych komponentów możliwych do wykorzystania we własnych złożonych programach komputerowych

Przedmiotowe efekty uczenia się

Wiedza:
Student zna i rozumie zasady programowania strukturalnego i obiektowego
Student zna i rozumie podstawowe metody, techniki i narzędzia wykorzystywane w procesie rozwiązywania zadań bioinformatycznych, głównie o charakterze inżynierskim
Student zna i rozumie cykl życia systemów informatycznych

Umiejętności:
Student potrafi pozyskiwać informacje z literatury, baz danych oraz innych właściwie dobranych źródeł, także w języku angielskim
Student potrafi integrować i interpretować uzyskane informacje, a także wyciągać wnioski oraz formułować i uzasadniać swoje opinie
Student potrafi projektować i tworzyć oprogramowanie komputerowe zgodnie z zadaną specyfikacją, używając właściwych metod, technik i narzędzi
Student potrafi dokonać analizy funkcjonalności i analizy wymagań systemów informatycznych
Student potrafi samodzielnie zdobywać wiedzę i podnosić swoje kwalifikacje

Kompetencje społeczne:
Student jest gotów do uczenia się przez całe życie i podnoszenia swoich kompetencji
Student jest gotów do współdziałania i pracy w grupie, przyjmując w niej różne role
Student jest gotów do określania priorytetów służących realizacji zadania zdefiniowanego przez siebie lub innych
Student jest gotów do wzięcia odpowiedzialności za podejmowane decyzje

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Efekty kształcenia przedstawione wyżej weryfikowane są w następujący sposób:
Ocena formująca:
a) w zakresie wykładów:
• aktywność w trakcie wykładów
b) w zakresie ćwiczeń:
• na podstawie oceny bieżącego postępu realizacji zadań oraz projektu zaliczeniowego
Ocena podsumowująca:
a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez ocenę wiedzy i umiejętności wykazanych na zaliczeniu pisemnym i omówienie wyników. Zaliczenie składa się ze zbioru pytań otwartych i zamkniętych dotyczących umiejętności i rozumienia elementów programowania obiektowego na praktycznych instancjach
b) w zakresie laboratoriów weryfikowanie założonych efektów kształcenia realizowane jest przez ocenianie ciągłe, na każdych zajęciach (odpowiedzi ustne), premiowanie przyrostu umiejętności posługiwania się poznanymi zasadami i metodami oraz narzędziami programowymi oraz ocenę z realizacji projektów zaliczeniowych

Treści programowe

Program wykładu obejmuje
• przesłanki programowania obiektowego,
• ideę nowego paradygmatu programowania, który wspiera tworzenie programów o wysokiej jakości.
• poszukiwanie optymalnego języka programowania i metodyk właściwych dla budowy uniwersalnych modułów programowych do wielokrotnego użytku
• związki paradygmatu obiektowego z inżynierią oprogramowania
• metryki jakości architektury programów komputerowych: kohezja i niezależność modułów programowych
• implementację pojęcia abstrakcyjnych typów danych
• poznanie podstawowych konstruktorów modelu obiektowego: klasy, obiektu, zmiennych i operacji klas, relacji generalizacji, związków między klasami
• przykłady prostych modeli fragmentów rzeczywistości
• definicje podstawowych pojęć obiektowych: obiekt, atrybuty (zmienne) obiektu, metody obiektu,
• przesyłanie komunikatów wyzwalających wywołania metod obiektów, interfejsy klas, obiekty

jako wystąpienia klas
• przykłady definiowania klas obejmujące: definicje konstruktorów i destruktorów klas, operatorów przeciążonych, zmiennych i metod klasowych
• hermetyczność implementacji klas jako mechanizm ograniczania związków między modułami programowymi
• relację przyjaźni między klasami
• porównanie rozwiązań prostych problemów w sposób funkcjonalny i obiektowy
• implementację obiektów złożonych i związków między obiektami
• dziedziczenie klas i relację podtypu między klasami
• definicję nowych cech klas pochodnych, przesłanianie metod i zmiennych, implementacja klas abstrakcyjnych
• dziedziczenie wirtualne w języku C++
• dziedziczenie konstruktorów i destruktorów klas
• definiowanie zmiennych polimorficznych i podstawienia polimorficzne
• implementację i przykłady zastosowania mechanizmu późnego wiązania
• dynamiczne rzutowanie typów danych
• zwiększenie stopnia uniwersalności klas przez definiowane klas generycznych
• granice stosowalności klas generycznych: generyczność ograniczona i nieograniczona
• typowe przykłady klas generycznych
• wzorce klas w języku C++
• tworzenie niezawodnych programów komputerowych
• poziomy bezpieczeństwa kodu
• podstawowe strategie tworzenia programów odpornych na występowanie błędów i wyjątków
• metodyki i techniki obsługi wyjątków w językach obiektowych
• definiowanie i zgłaszanie wyjątków, wyłapywanie wyjątków i ich obsługa
• przykłady zastosowań obsługi wyjątków.

Tematyka zajęć

brak

Metody dydaktyczne

Wykład:
prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, rozwiązywanie zadań, pokaz multimedialny
Ćwiczenia laboratoryjne:
rozwiązywanie zadań, ćwiczenia praktyczne, dyskusja, praca w zespole, pokaz multimedialny

Literatura

Podstawowa
1. Programowanie zorientowane obiektowo, Bertrand Meyer, Helion, Warszawa, 2005
2. Metody obiektowe w teorii i praktyce, Ian Graham, WNT, Warszawa, 2004
3. Język C++, Bjarne Stroustrup, WNT, Warszawa, 1994
4. Programowanie obiektowe, Peter Coad, Edward Yourdon, Read Me, 1994
5. Analiza obiektowa, Peter Coad, Edward Yourdon, Read Me, 1994
6. Nowoczesne projektowanie w C++, Andrei Alexandrescu, WNT, 2005
Uzupełniająca
1. B. Eckel, Thinking in C++, HELION, 2002
2. Nicolai M. Josuttis, C++ Biblioteka standardowa, Podrecznik programisty

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	125	5,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	60	2,50
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	65	2,50