KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Projektowanie i Programowanie Obiektowe

Przedmiot

Kierunek studiów

Bioinformatyka

Studia w zakresie (specjalność)

Poziom studiów

pierwszego stopnia

Forma studiów

stacjonarne

Rok/semestr

1/2

Profil studiów

ogólnoakademicki

Język oferowanego przedmiotu

polski

Wymagalność

obligatoryjny

Liczba godzin

Wykład	Laboratoria	Inne (np. online)
30	30	
Ćwiczenia	Projekty/seminaria	

Liczba punktów ECTS

4

Wykładowcy

Odpowiedzialny za przedmiot/wykładowca:

dr hab. inż. Piotr Łukasiak

e-mail: Piotr.Lukasiak@put.poznan.pl

tel.: 61 665 3033

wydział: Informatyki i Telekomunikacji

adres: ul. Piotrowo 2, 60-965 Poznań

Odpowiedzialny za przedmiot/wykładowca:

Wymagania wstępne

Student rozpoczynający ten moduł powinien posiadać podstawową wiedzę z algorytmów i złożoności, architektury systemów komputerowych, systemów operacyjnych, języków i paradygmatów programowania, grafiki i komunikacji człowiek-komputer, baz danych, ma podbudowaną teoretycznie szczegółową wiedzę związaną z wybranymi zagadnieniami z zakresu informatyki, ma wiedzę o trendach rozwojowych i najistotniejszych nowych osiągnięciach w informatyce, bioinformatyce i w wybranych pokrewnych dyscyplinach naukowych, zna podstawowe metody, techniki i narzędzia stosowane przy rozwiązywaniu zadań inżynierskich z obszaru informatyki Powinien posiadać umiejętność rozwiązywania podstawowych problemów z dziedziny informatyki, potrafi wykorzystać do formułowania i rozwiązywania zadań inżynierskich i prostych problemów badawczych metody analityczne, potrafi — przy formułowaniu i rozwiązywaniu zadań inżynierskich — integrować wiedzę z różnych obszarów

informatyki (a w razie potrzeby także wiedzę z innych dyscyplin naukowych). Powinien również rozumieć konieczność poszerzania swoich kompetencji / mieć gotowość do podjęcia współpracy w ramach zespołu.

Ponadto w zakresie kompetencji społecznych student musi prezentować takie postawy, jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.

Cel przedmiotu

1. Przekazanie studentom podstawowej wiedzy dotyczącej programowania obiektowego w zakresie podstawowym na przykładzie języków obiektowych C++ i JAVA, projektowania obiektowego (język UML) oraz grafiki komputerowej

2. Zapoznanie studentów z paradygmatem programowania obiektowego, sposobami projektowania systemów informatycznych oraz aspektami grafiki komputerowej

3. Rozwinięcie u studentów umiejętności rozwiązywania problemów inżynierskich, przekładania opisu potocznego na opis projektowy umożliwiający jego implementację oraz projektowania prostych rozwiązań wizualizacyjnych

4. Kształtowanie u studentów umiejętności analitycznego podejścia do rozwiązywania problemów informatycznych

Przedmiotowe efekty uczenia się

Wiedza

1. ma wiedzę z zakresu matematyki przydatną do formułowania i rozwiązywania prostych zadań bioinformatycznych,

2. zna zagadnienia z zakresu algorytmów i struktur danych oraz podstawy teorii złożoności obliczeniowej niezbędnej do implementacji optymalnych rozwiązań obliczeniowych

3. zna podstawowe zagadnienia z zakresu optymalizacji kombinatorycznej niezbędne do implementacji algorytmów

4. zna doskonale zasady programowania strukturalnego i obiektowego, potrafi zastosować odpowiednie rozwiązanie w zależności od zadanego problemu

5. zna podstawy grafiki komputerowej

6. ma uporządkowaną, wiedzę w zakresie modelowania problemów biologicznych na gruncie kombinatorycznym i ich zastosowania w implementacji odpowiednich rozwiązań

7. ma podstawową wiedzę o cyklu życia systemów informatycznych

Umiejętności

1. potrafi pozyskiwać informacje z literatury, baz danych oraz innych właściwie dobranych źródeł, także w języku angielskim

2. integruje i interpretuje uzyskane informacje, a także wyciąga wnioski oraz formułuje i uzasadnia swoje opinie

3. stosuje podstawowe techniki i narzędzia informatyczne z zakresu obiektowości do rozwiązywania problemów biologicznych i potrafi ocenić ich przydatność

4. pod kierunkiem opiekuna naukowego stosuje metody analityczne, symulacyjne oraz eksperymentalne do formułowania i rozwiązywania zadań badawczych

5. samodzielnie zdobywa wiedzę i podnosi swoje kwalifikacje

6. potrafi dokonać analizy funkcjonalności i analizy wymagań systemów informatycznych

7. projektuje i tworzy oprogramowanie komputerowe zgodnie z zadaną specyfikacją, używając właściwych metod, technik i narzędzi

Kompetencje społeczne

1. rozumie potrzebę uczenia się przez całe życie i podnoszenia swoich kompetencji

2. potrafi współdziałać i pracować w grupie, przyjmując w niej różne role

3. potrafi odpowiednio określić priorytety służące realizacji zadania określonego przez siebie lub innych

4. ma świadomość odpowiedzialności za podejmowane decyzje

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formująca

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach

b) w zakresie laboratoriów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- ocenę przygotowania studenta do poszczególnych sesji zajęć laboratoryjnych (sprawdzian „wejściowy") oraz ocenę umiejętności związanych z realizacją ćwiczeń laboratoryjnych
- ocenianie ciągłe, na każdych zajęciach (odpowiedzi ustne) – premiowanie przyrostu umiejętności posługiwania się poznanymi zasadami i metodami
- ocenę sprawozdań przygotowywanych po zakończeniu zajęć; ocena ta obejmuje także umiejętność pracy w zespole
- ocenę i „obronę” zrealizowanych przez studenta ćwiczeń laboratoryjnych

Ocena podsumowująca

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- ocenę wiedzy i umiejętności wykazanych na trzech kolokwiach pisemnych o charakterze problemowym w formie testu wielokrotnego wyboru składającego się każdorazowo z 10 zadań (po1 punkcie za zadanie). Na ocenę 3,0 należy zdobyć więcej niż 50% punktów z wszystkich punktów możliwych do zdobycia z 3 kolokwiów. Dodatkowe punkty do zaliczenia wykładu można zdobyć podczas problemowych zadań prezentowanych podczas wykładów i omawianych ze studentami
- omówienie wyników egzaminu.

b) w zakresie laboratoriów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- ocenę wiedzy i umiejętności związanych z treściami przekazywanymi na ćwiczeniach poprzez zadania realizowane na zajęciach (40% oceny końcowej) oraz dwa projekty zaliczeniowe (po 30% oceny końcowej) powiązane tematycznie z prezentowanymi zagadnieniami. Ocena końcowa jest sumą punktów zdobytych z w/w elementów, przy czym pozytywna ocena projektów zaliczeniowych jest warunkiem koniecznym do otrzymania pozytywnej oceny z zajęć.

Aktywność podczas zajęć premiowana jest dodatkowymi punktami, w szczególności za:

- omówienie dodatkowych aspektów zagadnienia,
- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,

uwagi prowadzące do udoskonalenia materiałów dydaktycznych lub procesu dydaktycznego.

Treści programowe

Program wykładu obejmuje następujące zagadnienia: podstawy programowania obiektowego, różnice pomiędzy metodami programowania (strukturalne, obiektowe, modularne, itp.), podstawowe pojęcia i charakterystyka języka C++, pojęcie klasy, metody, definicje obiektu, obiektowe postrzeganie rzeczywistości, dziedziczenie klas, przeciążanie klas i metod, zaprzyjaźnienie klas i metod, wzorce obiektowe, klasy abstrakcyjne, metody wirtualne przeciążanie operatorów, strumienie, wątki, wyjątki, Podstawowe pojęcia i charakterystyka języka Java, aplikacje, a aplety, programowanie w środowisku tekstowym i graficznym, biblioteki związane z programowaniem kontekstu graficznego, podstawy języka UML, zasady projektowania obiektowego na bazie języka UML, powiązanie projektowania z implementacją, podstawy grafiki komputerowej, wybrane metody implementacyjne związane z projektowaniem graficznych wizualizacji.

Ćwiczenia laboratoryjne prowadzone są w formie piętnastu dwugodzinnych zajęć odbywających się w laboratorium komputerowym. Pierwsze zajęcia przeznaczone są na zapoznanie studentów z zasadami

użytkowania laboratorium i zaliczania ćwiczeń. Ćwiczenia realizowane są przez jedno lub dwuosobowe zespoły studentów. Program zajęć laboratoryjnych obejmuje następujące zagadnienia: Definiowanie przedmiotów – klasy i obiekty, Deklaracja i definicja klasy, ciało klasy, Sekcja prywatna, zabezpieczona i publiczna, Słowa kluczowe klasy, a atrybuty dostępu do składowych, Dostęp do składowych klasy, Zmienna this, Deklaracje i definicje składowych klas, Składowe statyczne i uprzywilejowane, Konstruktory, destruktory (def. i zastosowanie), Dziedziczenie, Polimorfizm, Definicje zagnieżdżone, Kwalifikator zakresu, Konstruktory i destruktory w dziedziczeniu pojedynczym, Przeciążanie operatorów jednoargumentowych (przedrostkowych i przyrostkowych), Przeciążanie operatorów dwuargumentowych, Zaprzyjaźnienie, Wzorce w klasach (definicje, cele, zastosowanie, przykłady), Wirtualna maszyna Javy, Aplikacja, a aplet, Kompilacja apletu, Parametryzowanie apletu (osadzanie w kodzie HTML), Cykl życia apletu, Predefiniowane metody (init, start, stop, destroy), komponenty i kontenery, klasa Component, metody klasy Component, klasa Label, komponenty tekstowe, klasa TextField, konstruktory i metody klasy TextField, klasa Graphics, metoda paint, update, Obsługa zdarzeń od urządzeń zewnętrznych (mysz, klawiatura), Wczytywanie grafiki, Wyświetlanie obrazów, Podwójne buforowanie, Dźwięki w apletach, Wątki i procesy, Stany wątku, Klasa Thread, Synchronizacja i priorytety wątków, Obsługa wyjątków, Elementy języka Java: identyfikatory, słowa kluczowe, literały, operatory, typy, obiekty, zmienne, Typy prymitywne i klasowe, Enkapsulacja, Dziedziczenie, Interfejsy, Polimorfizm (metody wirtualne), Konstruktory w Javie, Klasy i metody parametryzowane typami, Definiowanie ograniczeń dla typów parametryzujących klasy i metody, Interfejs Collection i jego rozszerzenia, Pętla for-each i iteratory, Blok obsługi wyjątków: try…catch…finalny, Klasy Exception, RuntimeException i Error, Wyjątki kontrolowane i niekontrolowane, Nieobsłużone wyjątki, przekazywanie wyjątków, stos wyjątków, zgłaszanie wyjątków w bloku catch, Własne wyjątki, Wyjątki podczas tworzenia i inicjalizacji obiektów, Klasy: InputStream, OutputStream, Reader i Writer, Typy strumieni: pliku, tablicy bajtów/znaków, obiekcie String oraz łącza (ang. pipe), międzyprocesowego, Konwersje pomiędzy strumieniami (binarny/znakowy), Standardowe wejście/wyjście: przekierowywanie, kompresja, Nowe wejście/wyjście – java.nio.*, Klasy ObjectOutputStream, ObjectInputStream oraz interfejs Serializable, Serializacja obiektów i powiązań pomiędzy nimi, Składowe transient, Metody writeObject() i readObject(), Interfejs Externalizable, czynniki jakości oprogramowania, inżynieria oprogramowania, Notacja UML, model wymagań, pojęcie aktora, diagram przypadków użycia, diagram interakcji, cykl życia produktu informatycznego, diagram klas, diagram stanów, diagramy interakcji, diagramy komponentów, diagramy wdrożeniowe, tworzenie grafiki komputerowej, podstawowe środowiska programowania grafiki komputerowej, podstawowe aplikacje, zasady tworzenia elementów grafiki. Rastrowanie, Światło i barwa w grafice komputerowej

Metody dydaktyczne

1. 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. Bjarne Stroustrup, Jezyk C++, WNT, 2002
2. J. Grebosz, Symfonia C ++ standard
3. Grady Booch, James Rumbaugh, Ivar Jacobson, UML przewodnik użytkownika
4. B. Eckel, Thinking in Java, HELION, 2006
5. P. Shirley, Fundamentals of Computer Graphics, sec. ed. A K Peters, 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	120	4,0
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	65	2,0
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)[1]	55	2,0

[1] niepotrzebne skreślić lub dopisać inne czynności