KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Modelowanie i programowanie obiektowe	**1010515311010514676**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Informatyka	**ogólnoakademicki**	**1 / 1**

Ścieżka obieralności/specjalność	Przedmiot oferowany w języku:	Kurs (obligatoryjny/obieralny)
Sieci komputerowe	**polski**	**obligatoryjny**

Stopień studiów:	**Forma studiów** (stacjonarna/niestacjonarna)
II stopień	**niestacjonarna**

Godziny	Liczba punktów
Wykłady: **8** Ćwiczenia: **-** Laboratoria: **20** Projekty/seminaria: **-**	**3**

Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny)	(ogólnouczelniany, z innego kierunku)
kierunkowy	**z danego kierunku**

Obszar(y) kształcenia i dziedzina(y) nauki i sztuki	**Podział ECTS (liczba i %)**
nauki techniczne	**3 100%**
nauki techniczne	**3 100%**

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Cezary Sobaniec
email: cezary.sobaniec@cs.put.poznan.pl
tel. 61 665 2370
Informatyki
ul. Piotrowo 2, 60-965 Poznań

Odpowiedzialny za przedmiot / wykładowca:

mgr inż. Andrzej Stroiński
email: andrzej.stroinski@cs.put.poznan.pl
tel. 61 665 2371
Informatyki
ul. Piotrowo 2, 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z zakresu algorytmów i struktur danych.
2	**Umiejętności:**	Ponadto, student powinien posiadać umiejętność rozwiązywania podstawowych problemów oraz samodzielnego pisania, modyfikowania i testowania programów komputerowych. Niezbędna jest również umiejętność pozyskiwania informacji ze wskazanych źródeł oraz rozumienie konieczności pracy w grupie nad wspólnym projektem.
3	**Kompetencje społeczne**	Dodatkowo student musi prezentować takie postawy społeczne jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista i szacunek dla innych ludzi.

Cel przedmiotu:

1. Nauczenie studentów zasad modelowania oraz tworzenia programów zgodnie z paradygmatem obiektowym, z uwzględnieniem podziału takich aplikacji na re-używalne komponenty oraz tworzeniem nowych typów danych.

2. Rozwijanie u studentów umiejętności modelowania i tworzenia systemów informatycznych o odpowiedniej architekturze, która cechuje się spoistością składowych modułów programowych i luźnych związków między tymi modułami.

Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

Wiedza:

1. ma zaawansowaną i pogłębioną wiedzę z zakresu algorytmów, architektury systemów komputerowych, języków i paradygmatów programowania oraz inżynierii oprogramowania i środowisk programistycznych wykorzystywanych do ich implementacji - [K2st_W1]

2. ma uporządkowaną i podbudowaną teoretycznie wiedzę ogólną w zakresie algorytmów, architektury systemów komputerowych, języków i paradygmatów programowania oraz inżynierii oprogramowania - [K2st_W2]

3. ma zaawansowaną wiedzę szczegółową dotyczącą wybranych zagadnień z zakresu działania systemów informatycznych - [K2st_W3]

4. zna zaawansowane metody, techniki i narzędzia stosowane przy rozwiązywaniu złożonych zadań inżynierskich i prowadzeniu prac badawczych związanych z wytwarzaniem oprogramowania i opracowywaniem architektur systemów komputerowych - [K2st_W6]

Umiejętności:

1. potrafi ? przy formułowaniu i rozwiązywaniu zadań inżynierskich ? integrować wiedzę z różnych obszarów informatyki (a w razie potrzeby także wiedzę z innych dyscyplin naukowych, takich jak telekomunikacja, zarządzanie projektami, matematyka) oraz zastosować podejście systemowe, uwzględniające także aspekty pozatechniczne - [K2st_U5]

2. potrafi ocenić przydatność i możliwość wykorzystania nowych osiągnięć (metod i narzędzi) oraz nowych produktów informatycznych m.in. środowisk programistycznych, języków programowania, bibliotek - [K2st_U6]

3. potrafi dokonać krytycznej analizy istniejących rozwiązań technicznych (biblioteki programowe i narzędzia programistyczne) oraz zaproponować ich ulepszenia (usprawnienia) - [K2st_U8]

4. potrafi ocenić przydatność metod i narzędzi służących do rozwiązywania problemów zgodnie z podejściem obiektowym, polegających na budowie lub ocenie systemu informatycznego lub jego składowych, w tym dostrzec ograniczenia tych metod i narzędzi - [K2st_U9]

5. potrafi ? zgodnie z zadaną specyfikacją, uwzględniającą aspekty pozatechniczne ? zaprojektować złożone urządzenie, system informatyczny lub proces oraz zrealizować ten projekt ? co najmniej w części ? używając np. języka UML oraz języka programowania obiektowego, w tym przystosowując do tego celu istniejące lub opracowując nowe narzędzia - [K2st_U11]

6. potrafi współdziałać w zespole, przyjmując w nim różne role np. architekta systemowego, programisty itd. - [K2st_U15]

Kompetencje społeczne:

1. rozumie, że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe, szczególnie w zakresie języków programowania, powiązanych z nimi technologii oraz architektur systemowów informatycznych - [K2st_K1]

2. rozumie znaczenie wykorzystywania najnowszej wiedzy z zakresu informatyki w rozwiązywaniu problemów badawczych i praktycznych z wykorzystaniem podejścia obiektowego modelowania rzeczywistości - [K2st_K2]

Sposoby sprawdzenia efektów kształcenia

Efekty kształcenia przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formująca:

a) w zakresie wykładów:

- na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach,

b) w zakresie zajęć laboratoryjnych:

- na podstawie oceny bieżącego postępu realizacji zadań,

Ocena podsumowująca (wykład i laboratorium): ocenę wiedzy i umiejętności wykazanych na egzaminie pisemnym. Aby zaliczyć egzamin i uzyskać ocenę 3.0, student musi uzyskać co najmniej 50% maksymalnej liczby punktów. W trakcie egzaminu student nie może korzystać z materiałów dydaktycznych.

Uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za:

- omówienia dodatkowych aspektów zagadnienia,
- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,
- umiejętność współpracy w ramach zespołu praktycznie realizującego zadanie szczegółowe w laboratorium,
- uwagi związane z udoskonaleniem materiałów dydaktycznych,
- wskazywanie trudności percepcyjnych studentów umożliwiające bieżące doskonalenia procesu dydaktycznego.

Treści programowe

Program wykładów z przedmiotu obejmuje następujące zagadnienia:

1. Motywacje dla nowego paradygmatu oprogramowania - programowanie obiektowe. Idea nowego paradygmatu programowania. Różnice pomiędzy podejściem obiektowym, a proceduralnym. Historia języków obiektowych. Podstawowe pojęcia związane z paradygmatem obiektowym. Założenia paradygmatu obiektowego.

2. Problem konstrukcji złożonych systemów informatycznych. Analiza i projektowanie obiektowe. Obiektowe modelowanie dziedziny. Modelowanie z wykorzystaniem języka UML. Tworzenie diagramów przebiegu. Mapowanie języka UML na C# i odwrotnie. Karty CRC.

3. Podstawowe konstrukcje paradygmatu obiektowego oraz idee. Budowa klasy. Budowa programu. Zakresy widoczności. Tworzenie i niszczenie obiektów. Własności programów obiektowych i ich konsekwencje. Mechanizm dziedziczenia. Polimorfizm. Składowe klasy.

4. Wprowadzenie do platformy .Net. Historia platformy i jej rozwoju. Alternatywne platformy na przykładzie Mono. Przedstawienie podstawowych konstrukcji obiektowych na przykładzie języka C#.

W ramach zajęć laboratoryjnych studenci zapoznają się z obiektowym językiem programowania: C# oraz podstawowymi funkcjami środowiska programistycznego Visual Studio. Ćwiczenia polegają na samodzielnym tworzeniu programów zawierających podstawowe konstrukcje języków obiektowych przedstawianych na wykładach i laboratorium. Dodatkowo, realizowane są niewielkie zadania domowe poszerzające wiedzę i umiejętności zdobyte na zajęciach.

Cześć wymienionych wyżej treści programowych realizowana jest w ramach pracy własnej studenta.

Metody dydaktyczne:

1. wykład: pokaz multimedialny, prezentacja ilustrowana przykładami podawanymi na tablicy,
2. ćwiczenia laboratoryjne: ćwiczenia praktyczne, dyskusja, praca w zespole, pokaz multimedialny

Literatura podstawowa:

1. Programowanie zorientowane obiektowo : nowy sposób myślenia / Donald W. MacVittie, Lori A. MacVittie ; z ang. przeł. Grzegorz Niksiński. Wydawnictwo MIKOM. ZNI MIKOM, 1996.
2. Programowanie zorientowane obiektowo : nowy sposób myślenia / Donald W. MacVittie, Lori A. MacVittie ; z ang. przeł. Grzegorz Niksiński. Wydawnictwo MIKOM. ZNI MIKOM, 1996.
3. C# 3.0 : leksykon kieszonkowy / Joseph Albahari, Ben Albahari ; [tł. Przemysław Szeremiota]. Helion, 2008.
4. C#. Leksykon, Ben Albahari, Peter Drayton, Brad Merrill, Helion , Gliwice, 2001
5. Wstęp do programowania w języku C# / Adam Boduch. Wydawnictwo Helion, cop. 2006.
6. http://msdn.microsoft.com/en-us/library/vstudio/kx37x362.aspx

Literatura uzupełniająca:

1. Metody obiektowe w teorii i praktyce, Ian Graham, WNT, Warszawa, 2004
2. Microsoft Visual C# 2015 : krok po kroku / John Sharp ; przekład: Natalia Chounlamany, Janusz Machowski, Krzysztof Szkudlarek, Marek Włodarz. APN Promise, 2016.

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. udział w zajęciach laboratoryjnych:	20
2. przygotowanie do ćwiczeń laboratoryjnych:	20
3. realizacja zadań domowych:	5
4. udział w konsultacjach (mogą być realizowane drogą elektroniczną) związanych z realizacją procesu kształcenia, w szczególności ćwiczeń laboratoryjnych / zadań domowych	2
5. napisanie programu / programów, uruchomienie I weryfikacja (czas poza zajęciami laboratoryjnymi)	5
6. przygotowanie do egzaminu 8 godz. + egzamin 2godz.	10
7. udział w wykładach	8
8. zapoznanie się ze wskazaną literaturą / materiałami dydaktycznymi (10 stron tekstu naukowego = 1 godz.), 100 stron	10

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	80	3
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	32	1
Zajęcia o charakterze praktycznym	50	2