KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Języki programowania obiektowego [S1Teleinf1>JPO]

Przedmiot

Kierunek studiów
Teleinformatyka

Rok/Semestr
2/4

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
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

5,00

Koordynatorzy

dr inż. Paweł Sroka
pawel.sroka@put.poznan.pl

dr inż. Marek Michalski
marek.michalski@put.poznan.pl

dr hab. inż. Remigiusz Rajewski
remigiusz.rajewski@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student zna i rozumie – w zaawansowanym stopniu – wybrane fakty, obiekty i zjawiska oraz dotyczące ich metody i teorie wyjaśniające złożone zależności między nimi, stanowiące podstawową wiedzę ogólną z zakresu matematyki. Student potrafi pozyskiwać informacje z literatury, baz danych i innych źródeł; potrafi integrować uzyskane informacje, dokonywać ich interpretacji, a także wyciągać wnioski oraz formułować i uzasadniać opinie; potrafi formułować i rozwiązywać złożone i nietypowe problemy oraz wykonywać zadania w warunkach nie w pełni przewidywalnych, poprzez właściwy dobór źródeł oraz informacji z nich pochodzących, dokonywanie oceny, krytycznej analizy i syntezy tych informacji. Student jest gotów do krytycznej oceny posiadanej wiedzy, uznania znaczenia wiedzy w rozwiązywaniu problemów poznawczych i praktycznych

Cel przedmiotu

Szczegółowe zapoznanie studentów ze stylem programowania obiektowego. Opanowanie przez nich umiejętności posługiwania się konstrukcjami języków obiektowych, w tym: - projektowania i implementowania obszernych algorytmów w językach obiektowych C++ i Java, - wdrażania zasad solidnego programowania obiektowego (SOLID). Opanowanie zasad doboru stylu i języka programowania do charakteru zadania

Przedmiotowe efekty uczenia się

Wiedza:
1. Student potrafi opracować dokumentację dotyczącą realizacji zadania inżynierskiego i przygotować tekst zawierający omówienie wyników realizacji tego zadania; potrafi komunikować się z użyciem specjalistycznej terminologii.
2. Student potrafi konstruować algorytmy z wykorzystaniem podstawowych technik algorytmicznych, dokonać analizy ich złożoności i je ocenić.
3. Student potrafi posłużyć się środowiskami i platformami programistycznymi do pisania, wykonywania i testowania prostych programów kodowanych w językach programowania imperatywnego, obiektowego i deklaratywnego; potrafi przy identyfikacji i formułowaniu specyfikacji zadań inżynierskich oraz ich rozwiązywaniu:
− wykorzystać metody analityczne, symulacyjne i eksperymentalne.

Umiejętności:
1. Student ma uporządkowaną i podbudowaną teoretycznie wiedzę w zakresie podstawowych algorytmów i ich
analizy, technik projektowania algorytmów, abstrakcyjnych struktur danych i ich implementacji, problemów obliczeniowo trudnych.
2. Student zna i rozumie w zaawansowanym stopniu – wybrane fakty, obiekty i zjawiska oraz dotyczące ich metody i teorie wyjaśniające złożone zależności między nimi, stanowiące podstawową wiedzę ogólną z zakresu programowania.

Kompetencje społeczne:
1. Student ma świadomość ważności i rozumie pozatechniczne aspekty i skutki działalności inżyniera-teleinformatyka i związaną z tym odpowiedzialność za podejmowane decyzje, jest gotów do dbałości o dorobek i
tradycje zawodu.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:
Wykład: egzamin pisemny lub ustny.
Laboratorium: zaliczenie na podstawie kolokwiów, aktywności programistycznej na zajęciach, programów napisanych w ramach prac domowych lub (opcjonalnie) zrealizowania indywidualnego zadania projektowego (specyfikacja wymagań, projekt rozwiązania, implementacja w C++ lub Javie, dokumentacja).
Kryterium egzaminacyjne i zaliczeniowe: od 50,1%.

Treści programowe

Wykład (treści ogólne)
Klasyfikacja stylów programowania. Cele i zasady modelowania obiektowego. Język UML - najczęściej używane
diagramy struktur i zachowań.
Podstawowe paradygmaty programowania zorientowanego obiektowo (hermetyzacja, dziedziczenie, polimorfizm) i ich implementacja w języku C++. Obsługa błędów i obsługa wyjątków w języku obiektowym. Przeciążanie nazw funkcji i operatorów. Biblioteki wejścia-wyjścia w języku C++. Dynamiczne zarządzanie pamięcią w językach obiektowych. Programowanie uogólnione (wzorce), biblioteka STL. Programowanie wielowątkowe. Wyrażenia regularne i klasa boost::regex.
Elementy programowania w języku Java: kod bajtowy, implementacja klas i obiektów, implementacja wejścia-wyjścia, pakiety, interfejsy, kolekcje, wyjątki i ich obsługa, programowanie wielowątkowe,

historyczne aplety.
SOLID - zasady efektywnego programowania w języku obiektowym.
Laboratorium (treści ogólne)
Projektowanie algorytmów i ich implementacja w językach C++ i Java
Wykład (treści szczegółowe - jak wyżej, bez zmian)
Klasyfikacja stylów programowania. Cele i zasady modelowania obiektowego. Język UML - najczęściej używane diagramy struktur i zachowań.
Podstawowe paradygmaty programowania zorientowanego obiektowo (hermetyzacja, dziedziczenie, polimorfizm)
i ich implementacja w języku C++. Biblioteki wejścia-wyjścia w języku C++. Obsługa błędów i obsługa wyjątków w
języku obiektowym. Przeciążanie nazw funkcji i operatorów. Dynamiczne zarządzanie pamięcią w językach obiektowych. Programowanie uogólnione (wzorce), biblioteka STL. Programowanie wielowątkowe. Wyrażenia
regularne i klasa boost::regex.
4
Elementy programowania w języku Java: kod bajtowy, implementacja klas i obiektów, implementacja wejścia-
wyjścia, pakiety, interfejsy, kolekcje, wyjątki i ich obsługa, programowanie wielowątkowe, historyczne aplety.
SOLID - zasady efektywnego programowania w języku obiektowym.
Laboratorium (treści szczegółowe)
Projektowanie algorytmów i ich implementacja w języku C++. Praca w zintegrowanym środowisku programistycznym MS Visual Studio.
Projektowanie algorytmów i ich implementacja w języku Java. Praca w zintegrowanym środowisku programistycznym rozpowszechnianym z platformą Eclipse.

Tematyka zajęć

Wykład (treści ogólne)
Klasyfikacja stylów programowania. Cele i zasady modelowania obiektowego. Język UML - najczęściej używane
diagramy struktur i zachowań.
Podstawowe paradygmaty programowania zorientowanego obiektowo (hermetyzacja, dziedziczenie, polimorfizm) i ich implementacja w języku C++. Obsługa błędów i obsługa wyjątków w języku obiektowym. Przeciążanie nazw funkcji i operatorów. Biblioteki wejścia-wyjścia w języku C++. Dynamiczne zarządzanie pamięcią w językach obiektowych. Programowanie uogólnione (wzorce), biblioteka STL. Programowanie wielowątkowe. Wyrażenia regularne i klasa boost::regex.
Elementy programowania w języku Java: kod bajtowy, implementacja klas i obiektów, implementacja wejścia-wyjścia, pakiety, interfejsy, kolekcje, wyjątki i ich obsługa, programowanie wielowątkowe, historyczne aplety.
SOLID - zasady efektywnego programowania w języku obiektowym.
Laboratorium (treści ogólne)
Projektowanie algorytmów i ich implementacja w językach C++ i Java
Wykład (treści szczegółowe - jak wyżej, bez zmian)
Klasyfikacja stylów programowania. Cele i zasady modelowania obiektowego. Język UML - najczęściej używane diagramy struktur i zachowań.
Podstawowe paradygmaty programowania zorientowanego obiektowo (hermetyzacja, dziedziczenie, polimorfizm)
i ich implementacja w języku C++. Biblioteki wejścia-wyjścia w języku C++. Obsługa błędów i obsługa wyjątków w
języku obiektowym. Przeciążanie nazw funkcji i operatorów. Dynamiczne zarządzanie pamięcią w językach obiektowych. Programowanie uogólnione (wzorce), biblioteka STL. Programowanie wielowątkowe. Wyrażenia
regularne i klasa boost::regex.
4
Elementy programowania w języku Java: kod bajtowy, implementacja klas i obiektów, implementacja wejścia-
wyjścia, pakiety, interfejsy, kolekcje, wyjątki i ich obsługa, programowanie wielowątkowe, historyczne aplety.

SOLID - zasady efektywnego programowania w języku obiektowym.
Laboratorium (treści szczegółowe)
Projektowanie algorytmów i ich implementacja w języku C++. Praca w zintegrowanym środowisku programistycznym MS Visual Studio.
Projektowanie algorytmów i ich implementacja w języku Java. Praca w zintegrowanym środowisku programistycznym rozpowszechnianym z platformą Eclipse.

Metody dydaktyczne

Wykład:
- konwersatoryjny (z elementami dyskusji)
Ćwiczenia laboratoryjne:
- wprowadzenie do tematu (krótka prelekcja), następnie samodzielne rozwiązywanie zadań, bieżąca korekt i ocena rozwiązań
- prezentacja indywidualnego projektu w dwóch etapach (metodą seminaryjną):
- specyfikacja wymagań względem aplikacji, projekt aplikacji (UML)
- aplikacja w działaniu, szczegóły implementacji

Literatura

Podstawowa:
1. Stroustrup B., Język C++. Kompendium wiedzy. Wydanie IV, Helion, 2014.
2. Grębosz J., Opus magnum C++11. Programowanie w języku C++. Tom 1-3, Helion, 2020.
3. Prata S., Język C++. Szkoła programowania. Wydanie VI, Helion, 2012.
4. Schildt H., Java. Przewodnik dla początkujących. Wydanie VIII, Helion, 2022.

Uzupełniająca:
1. Prata S., Język C. Szkoła programowania. Wydanie VI, Helion, 2016.
2. Stroustrup B., Programowanie. Teoria i praktyka z wykorzystaniem C++, Wydanie III, Helion, 2020.
3. Schildt H., Java. Kompendium programisty. Wydanie XI, Helion, 2020.
4. Eckel B., Thinking in Java. Edycja polska. Wydanie VI, Helion, 2017.

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	120	5,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	64	3,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	56	2,00