



## KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Programowanie obiektowe [S1S1E>POB]

### Przedmiot

Kierunek studiów

Sztuczna inteligencja/Artificial Intelligence

Rok/Semestr

2/3

Studia w zakresie (specjalność)

–

Profil studiów

ogólnoakademicki

Poziom studiów

pierwszego stopnia

Język oferowanego przedmiotu

angielski

Forma studiów

stacjonarne

Wymagalność

obligatoryjny

### Liczba godzin

Wykład

15

Laboratorium

30

Inne (np. online)

0

Ćwiczenia

0

Projekty/seminaria

0

### Liczba punktów ECTS

4,00

### Koordynatorzy

dr hab. inż. Dariusz Brzeziński prof. PP  
dariusz.brzezinski@put.poznan.pl

### Wykładowcy

### Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę uzyskaną na przedmiotach: Wprowadzenie do informatyki oraz Algorytmy i Struktury Danych. Powinien posiadać umiejętność rozwiązywania podstawowych problemów ze specyfikacją algorytmów, samodzielnego pisania, modyfikowania i testowania programów komputerowych oraz umiejętność pozyskiwania informacji ze wskazanych źródeł. Powinien również rozumieć konieczność poszerzania swoich kompetencji i 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 i szacunek dla innych ludzi.

### Cel przedmiotu

Nauczenie studentów zasad tworzenia modułów programowych zdolnych do wielokrotnego wykorzystania w różnych projektach oraz łatwych w rozwoju i pielęgnacji, poprzez zastosowanie unikalnych rozwiązań dostępnych w językach obiektowych. Ponadto celem jest nauczenie studentów tworzenia własnych typów danych, rozwijanie umiejętności projektowania systemów informatycznych o poprawnej architekturze oraz kształtowanie umiejętności komunikacji podczas niezależnego tworzenia modułów programów komputerowych.

## Przedmiotowe efekty uczenia się

### Wiedza:

1. Student ma uporządkowaną, podbudowaną teoretycznie podstawową wiedzę dotyczącą języków i paradygmatów programowania oraz inżynierii oprogramowania.
2. Ma podstawową wiedzę o cyklu życia oraz procesach zachodzących w programowych systemach informatycznych.

### Umiejętności:

1. Student posiada podstawowe umiejętności informatyczne w zakresie analizy złożoności obliczeniowej algorytmów, programowania z użyciem popularnych języków oraz użytkowania systemów operacyjnych i szerokiego spektrum systemów informatycznych.
2. Potrafi zaprojektować - zgodnie z zadaną specyfikacją - oraz zrealizować system informatyczny, dobierając i stosując dostępne metody, techniki i narzędzia informatyczne, w szczególności oparte o metodykę programowania obiektowego.
3. Ma umiejętność prostej adaptacji istniejących oraz formułowania i implementacji nowych algorytmów z użyciem przynajmniej jednego z popularnych narzędzi programowania obiektowego.

### Kompetencje społeczne:

1. Rozumie, że języki, paradygmaty i inżynieria programowania są wciąż rozwijającymi się dyscyplinami nauki, dostrzegając przy tym potrzebę ciągłego dokształcania oraz podnoszenia własnych kompetencji.
2. Ma świadomość istotności wiedzy i badań naukowych związanych z paradygmatami programowania (w tym programowania obiektowego) w rozwiązywaniu praktycznych problemów o kluczowym znaczeniu dla funkcjonowania jednostek, firm, organizacji oraz całego społeczeństwa.

## Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Wiedza nabyta w ramach wykładu jest weryfikowana przez zaliczenie pisemne o różnej charakterystyce problemów do rozwiązania: pytania otwarte, proste zadania programistyczne, zadania z modelowania obiektowego. Zaliczenie pod warunkiem uzyskania ponad połowy punktów.

Umiejętności nabyte w ramach zajęć laboratoryjnych weryfikowane są poprzez ocenę dwóch projektów.

## Treści programowe

### Wykład:

1. Czym jest programowanie obiektowe i kiedy je stosować. Programowanie obiektowe a inne paradygmaty programowania. Podstawowe cechy języków zorientowanych obiektowo.
2. Analiza obiektowa. Modelowanie obiektowe za pomocą diagramów klas UML: dziedziczenie, implementacja, agregacja, kompozycja, asocjacja. Inne diagramy UML.
3. SOLID, czyli pięć podstawowych założeń programowania obiektowego: zasada jednej odpowiedzialności, zasada otwarte-zamknięte, zasada podstawienia Liskov, zasada segregacji interfejsów, zasada odwrócenie zależności.
4. Dziedziczenie, klasy abstrakcyjne, klasy generyczne: implementacje w różnych językach, przykłady użycia.
5. Wybrane wzorce projektowe z przykładami w różnych językach zorientowanych obiektowo.
6. Rozszerzenia języków programowania obiektowego: refleksja, typy anonimowe, wyrażenia lambda, kowariancja i kontrawariancja, programowanie asynchroniczne, typowanie automatyczne i dynamiczne, elementy programowania deklaratywnego.
7. Programowanie przez kontrakt.
8. Programowanie aspektowe i dalsze kierunki rozwoju.

### Laboratorium:

1. Laboratorium wprowadzające: zasady programowania obiektowego, proste diagramy klas UML.
2. C++ - wskaźniki, klasy, dziedziczenie, abstrakcja i polimorfizm.
3. C++ - operatory, klasa nadrzędna, przyjaciele.
4. C++ - szablony, obietnice i wyjątki.
5. Wprowadzenie do języka Java.
6. Dziedziczenie w Javie.
7. Kolekcje.
8. Projektowanie interfejsów użytkownika (Swing i JavaFX).
9. Wątki i mechanizmy kontroli współbieżności.

10. Strumienie wejścia/wyjścia i serializacja.
11. Wyjątki i debugowanie.
12. Typy generyczne w Javie.
13. Rozszerzenia języków obiektowych (Java, C++, C#).
14. Profiler i optymalizacja programów.
15. Zdawanie projektów końcowych.

### Metody dydaktyczne

Wykład: prezentacja multimedialna, burza mózgów, prezentacja ilustrowana przykładami podawanymi na tablicy.

Ćwiczenia laboratoryjne: ćwiczenia praktyczne, dyskusja, quiz, praca w zespole, prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy.

### Literatura

Podstawowa

Thinking in Java. Edycja polska, Bruce Eckel, Helion, Warszawa, 2017

Czysty kod. Podręcznik dobrego programisty, Robert C. Martin, Helion, Warszawa, 2014

Język C++. Kompendium wiedzy, Bjarne Stroustrup, Helion, Warszawa, 2014

Uzupełniająca

The Agile Samurai. How Agile Masters Deliver Great Software, Jonathan Rasmusson, Pragmatic Programmers, USA, 2017

### Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	100	4,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	45	2,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwium/egzaminu, wykonanie projektu)	55	2,00