SYLABUS

DOTYCZY CYKLU KSZTAŁCENIA *2025-2028*

(skrajne daty)

Rok akademicki 2025/2026

1. PODSTAWOWE INFORMACJE O PRZEDMIOCIE

Nazwa przedmiotu	Programowanie obiektowe
Kod przedmiotu*	
Nazwa jednostki prowadzącej kierunek	Wydział Nauk Ścisłych i Technicznych
Nazwa jednostki realizującej przedmiot	Wydział Nauk Ścisłych i Technicznych Instytut Matematyki
Kierunek studiów	Matematyka
Poziom studiów	studia I stopnia
Profil	ogólnoakademicki
Forma studiów	stacjonarne
Rok i semestr studiów	rok I, semestr 2
Rodzaj przedmiotu	kierunkowy
Język wykładowy	język polski
Koordynator	dr Paweł Pasteczka
Imię i nazwisko osoby prowadzącej / osób prowadzących	dr Paweł Pasteczka

***** *- zgodnie z ustaleniami w Jednostce*

1.1.Formy zajęć dydaktycznych, wymiar godzin i punktów ECTS

Semestr (nr)	Wykł.	Ćw.	Konw.	Lab.	Sem.	ZP	Prakt.	Inne (jakie?)	**Liczba pkt ECTS**
2	15			30					4

1.2. Sposób realizacji zajęć

☐ zajęcia w formie tradycyjnej

TAK zajęcia realizowane z wykorzystaniem metod i technik kształcenia na odległość

1.3 Forma zaliczenia przedmiotu (z toku) (egzamin, zaliczenie z oceną, zaliczenie bez oceny)
Wykład-Zaliczenie, ćwiczenia - zaliczenie z oceną

2.WYMAGANIA WSTĘPNE

Podstawy programowania

3. CELE, EFEKTY UCZENIA SIĘ , TREŚCI PROGRAMOWE I STOSOWANE METODY DYDAKTYCZNE

3.1 Cele przedmiotu

C1	Przygotowanie do rozwiązywania problemów (również matematycznych) przy użyciu środków informatyki oraz korzystania z komputerów na zajęciach matematycznych.
C2	Zapoznanie studentów z modelowania problemów oraz ich rozwiązania przy pomocy podejścia obiektowego
C3	Wprowadzenie w zagadnienia programowania obiektowego – modelowanie dziedziny i zapis w języku programowania.
C4	Zapoznanie studentów z podstawowymi metodami projektowania algorytmów, pisania, uruchamiania i testowania programów. Analiza wyników obliczeń.

3.2 Efekty uczenia się dla przedmiotu

EK (efekt uczenia się)	Treść efektu uczenia się zdefiniowanego dla przedmiotu	Odniesienie do efektów kierunkowych
EK_01	student zna podstawy technik obliczeniowych i programowania, które mogą być wykorzystane we wspomaganiu pracy matematyka oraz rozumie ograniczenia pojawiające się przy ich wykorzystaniu.	K_W05
EK_02	student potrafi rozpoznać i dokonać specyfikacji problemu, który można rozwiązać algorytmicznie, student dla danego problemu potrafi ułożyć i przeanalizować algorytm zgodny ze specyfikacją i zapisać go w odpowiednim języku programowania, a następnie testować napisany samodzielnie program komputerowy i dokonywać niezbędnych korekt.	K_U14
EK_03	student wyraża własne opinie na temat teoretycznych i praktycznych zagadnień z algorytmiki dotyczących danego problemu oraz formułuje pytania służące zrozumieniu badanego problemu, a w razie potrzeby uzupełnia swoje kompetencje.	K_K01
EK_04	student jest gotów do prezentowania krytycznej postawy wobec odbieranych treści, ma świadomość błędów, które mogą się pojawić przy układaniu algorytmów oraz zapisach w używanych językach programowania.	K_K02, K_K03

3.3 Treści programowe

A. Problematyka wykładu

Treści merytoryczne
Obiektowe modelowanie dziedziny: klasy pojęciowe, powiązania, atrybuty
Obiekty a programowanie: atrybuty, metody, hierarchia klas, widoczność, kapsułkowanie, UML
Java: identyfikatory, literały, proste instrukcje, instrukcje imperatywne w Java, zmienne, typy prymitywne i referencyjne, tablice, obiekty, kolekcje (wybrane klasy implementujące java.util.Collection), strumienie, funkcje (metody), parametry.

Kapsułkowanie, konstruktory i inicjalizacja składnia deklaracji, dziedziczenie, klasy wirtualne, modyfikatory dostępu, polimorfizm, typy uogólnione, klasy parametryzowane typami, interfejsy
Wyjątki Klasa *java.lang.Throwable* i jej podklasy. Wyjątki nadzorowane i nienadzorowane. Try/catch

B. Problematyka ćwiczeń laboratoryjnych

Treści merytoryczne
Obiektowe modelowanie dziedziny: odnajdywanie powiązań, atrybutów, hierarchia klas, UML
Wstęp do Java: Kompilacja, maszyna wirtualna Javy, pakiety, IDE, dokumentacja, narzędzia do pracy współbieżnej (CodeWithMe)
Java: Imperatywne instrukcje języka Java, proste klasy, zmienne, metody, wyjątki nadzorowane i nienadzorowane, typy uogólnione, kolekcje
Rozdzielanie odpowiedzialności między klasy: dziedziczenie, interfejsy, hierarchia klas, pakiety, modyfikatory dostępu
Rozwiązywanie problemów metodami programowania obiektowego

3.4 Metody dydaktyczne

Wykład z prezentacją multimedialną.

Laboratorium: rozwiązywanie zadań, dyskusja, projektowanie klas, tworzenie programów. Zajęcia prowadzone zdalnie. Do współpracy wykorzystywany będzie MSTeams oraz narzędzie CodeWithMe (by JetBrains).

4. METODY I KRYTERIA OCENY

4.1 Sposoby weryfikacji efektów uczenia się

Symbol efektu	Metody oceny efektów uczenia się (np.: kolokwium, egzamin ustny, egzamin pisemny, projekt, sprawozdanie, obserwacja w trakcie zajęć)	Forma zajęć dydaktycznych (w, ćw, ...)
EK_01	projekt zaliczeniowy	wykład, laboratorium
EK_ 02	projekt zaliczeniowy	wykład, laboratorium
EK_ 03	projekt zaliczeniowy	laboratorium
EK_ 04	projekt zaliczeniowy	laboratorium

4.2 Warunki zaliczenia przedmiotu (kryteria oceniania)

Wykład

Zaliczenie wykładu na podstawie testu

Laboratorium:

Warunkiem uzyskania zaliczenia z laboratorium jest uzyskanie co najmniej 51 procent punktów możliwych do uzyskania z projektów. W przypadku niespełnienia powyższych warunków student może wysłać projekty w terminie poprawkowym. Aktywność podczas zajęć premiowana jest dodatkowymi punktami. Oceny wystawiane są według następującej skali procentowej:

0 – 50	niedostateczny
51 – 60	dostateczny
61 – 70	plus dostateczny
71 – 80	dobry
81 – 90	plus dobry
91 – 100	bardzo dobry

5. CAŁKOWITY NAKŁAD PRACY STUDENTA POTRZEBNY DO OSIĄGNIĘCIA ZAŁOŻONYCH EFEKTÓW W GODZINACH ORAZ PUNKTACH ECTS

Forma aktywności	Średnia liczba godzin na zrealizowanie aktywności
Godziny z harmonogramu studiów	45
Inne z udziałem nauczyciela (udział w konsultacjach, egzaminie)	5
Godziny niekontaktowe – praca własna studenta (przygotowanie do zajęć, egzaminu, napisanie referatu itp.)	50
SUMA GODZIN	100
SUMARYCZNA LICZBA PUNKTÓW ECTS	4

** Należy uwzględnić, że 1 pkt ECTS odpowiada 25-30 godzin całkowitego nakładu pracy studenta.*

6. PRAKTYKI ZAWODOWE W RAMACH PRZEDMIOTU

wymiar godzinowy	nie dotyczy
zasady i formy odbywania praktyk	nie dotyczy

7. LITERATURA

Literatura podstawowa:

1. Bruce Eckel, Thinking in Java, Wydawnictwo Helion, Gliwice 2006.

2. Cay Horstmann, Gary Cornell„ Java 2. Podstawy", Wydawnictwo Helion, Gliwice 2003.

3. Matt Weisfeld, Myślenie obiektowe w programowaniu, Helion, Gliwice 2020.

Literatura uzupełniająca:

1. Grady Booch, James Rumbaugh, Ivar Jacobson,UML Przewodnik użytkownika, WNT, Warszawa 2012

2. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Wzorce projektowe, Elementy oprogramowania obiektowego wielokrotnego użytku, WNT, Warszawa 2008

3, Alan Shalloway, James R. Trott, Programowanie zorientowane obiektowo. Wzorce projektowe, Helion, Gliwice 2005

Akceptacja Kierownika Jednostki lub osoby upoważnionej