POLITECHNIKA GDAŃSKA

Karta przedmiotu

Nazwa i kod przedmiotu	Obiektowe języki programowania III, PG_00060228		
Kierunek studiów	Fizyka Techniczna		
Data rozpoczęcia studiów	październik 2023 r.	Rok akademicki realizacji przedmiotu	2025/2026
Poziom kształcenia	I stopnia - inżynierskie	Grupa zajęć	Grupa zajęć fakultatywnych Grupa zajęć powiązanych z prowadzonymi badaniami naukowymi w dziedzinie nauki związanej z kierunkiem - profil ogólnoakademicki
Forma studiów	stacjonarne	Sposób realizacji	na uczelni
Rok studiów	3	Język wykładowy	angielski
Semestr studiów	5	Liczba punktów ECTS	5.0
Profil kształcenia	ogólnoakademicki	Forma zaliczenia	zaliczenie
Jednostka prowadząca	Wydziały Politechniki Gdańskiej -> Wydział Fizyki Technicznej i Matematyki Stosowanej -> Katedra Fizyki Teoretycznej i Informatyki Kwant.		
Imię i nazwisko wykładowcy (wykładowców)	Odpowiedzialny za przedmiot	dr hab. Jan Franz	
	Prowadzący zajęcia z przedmiotu		

Formy zajęć i metody nauczania

Forma zajęć	Wykład	Ćwiczenia	Laboratorium	Projekt	Seminarium	RAZEM
Liczba godzin zajęć	15.0	0.0	60.0	0.0	0.0	75

W tym liczba godzin zajęć na odległość: 0.0

Adresy kursu na platformie eNauczanie:

Moodle ID: 1306 Obiektowe języki programowania III
https://enauczanie.pg.edu.pl/2025/course/view.php?id=1306

Aktywność studenta i liczba godzin pracy

Aktywność studenta	Udział w zajęciach dydaktycznych, objętych planem studiów	Udział w konsultacjach	Praca własna studenta	RAZEM
Liczba godzin pracy studenta	75	5.0	45.0	125

Cel przedmiotu

Celem kursu jest wprowadzenie studentów do programowania obiektowego (OOP) w języku Java ze szczególnym uwzględnieniem zastosowań w fizyce i informatyce stosowanej. Studenci nauczą się projektować, implementować i testować oprogramowanie naukowe z wykorzystaniem nowoczesnych narzędzi, bibliotek i wzorców projektowych Javy. Nacisk położony jest na pisanie niezawodnego, łatwego w utrzymaniu kodu oraz rozwijanie umiejętności potrzebnych w większych projektach badawczych i technologicznych.

Efekty uczenia się przedmiotu	Efekt kierunkowy	Efekt z przedmiotu	Sposób weryfikacji i oceny efektu
	[K6_U03] Posiada umiejętność programowania w wybranym języku oraz stosowania podstawowych pakietów oprogramowania.	potrafi pisać programy w języku obiektowym, korzystać z narzędzi do zarządzania projektami, stosować frameworki testowe oraz wykorzystywać wybrane biblioteki naukowe do rozwiązywania problemów z zakresu fizyki i technologii.	[SU1] Ocena realizacji zadania
	[K6_W01] Rozumie cywilizacyjne znaczenie fizyki i jej zastosowań.	potrafi modelować proste układy fizyczne z wykorzystaniem programowania obiektowego oraz dostrzegać, jak umiejętności obliczeniowe wspierają szersze zastosowania fizyki w nauce i technologii.	[SW2] Ocena wiedzy zawartej w prezentacji
	[K6_K01] Rozumie potrzebę uczenia się przez całe życie oraz potrzebę podnoszenia kompetencji zawodowych i osobistych. Potrafi inspirować i organizować proces uczenia się innych osób.	potrafi samodzielnie poszerzać swoją wiedzę z zakresu programowania obiektowego, krytycznie stosować narzędzia i wzorce projektowe programowania obiektowego do rozwiązywania problemów naukowych oraz współpracować w sposób wspierający i inspirujący proces uczenia się innych.	[SK5] Ocena umiejętności rozwiązywania problemów występujących w praktyce
	[K6_W05] Posiada wiedzę w zakresie metodyki i technik programowania oraz wykorzystywania wybranych narzędzi informatycznych w fizyce i technice.	potrafi stosować metody i techniki programowania obiektowego oraz efektywnie wykorzystywać wybrane narzędzia informatyczne do rozwiązywania problemów z zakresu fizyki i technologii.	[SW1] Ocena wiedzy faktograficznej

Treści przedmiotu	1. Ekosystem Javy i konfiguracja projektu Wykład: Java Virtual Machine (JVM), Java Development Kit (JDK), Zintegrowane Środowiska Programistyczne (IDE); zarządzanie projektem za pomocą Maven i Gradle. Laboratorium: Utworzenie pierwszego projektu Maven; uruchomienie prostego programu związanego z fizyką. 2. Klasy, obiekty i testowanie Wykład: klasy, pola, metody, konstruktory; wprowadzenie do testów jednostkowych w JUnit. Laboratorium: Implementacja klasy Particle oraz podstawowe testy jednostkowe. 3. Typy proste, klasy opakowujące, tablice i Efficient Java Matrix Library (EJML) Wykład: typy proste kontra obiekty; tablice; klasy opakowujące; pierwsze użycie EJML. Laboratorium: Operacje wektorowe przy użyciu tablic i EJML. 4. Dziedziczenie i interfejsy Wykład: dziedziczenie, przesłanianie metod, klasy abstrakcyjne, interfejsy. Laboratorium: Hierarchia klas dla różnych typów cząstek. 5. Wyjątki i niezawodny kod Wykład: wyjątki sprawdzane i niesprawdzane; strategie obsługi błędów. Laboratorium: Niezawodna obsługa wejścia/wyjścia (I/O) oraz obsługa błędów w prostej symulacji. 6. Framework kolekcji Wykład: List, Set, Map; iteratory; kiedy stosować kolekcje. Laboratorium: Przechowywanie i analiza trajektorii cząstek z użyciem kolekcji. 7. Wzorce projektowe I

Wykład: Factory, Singleton, Observer (z prostymi ilustracjami w Unified Modeling Language UML).

Laboratorium: Implementacja fabryki cząstek i obserwatora do logowania.

8. Klasy i metody generyczne oraz kolekcje w praktyce

Wykład: klasy i metody generyczne; implementacje kolekcji.

Laboratorium: Generyczne kontenery na wyniki; wykorzystanie posortowanych zbiorów/map.

9. Refaktoryzacja i praktyki testowania

Wykład: spójność, powiązania, zasady SOLID (Single responsibility, Openclosed, Liskov substitution, Interface segregation, Dependency inversion); programowanie sterowane testami (TDD).

Laboratorium: Refaktoryzacja wcześniejszego kodu i rozszerzenie testów jednostkowych.

10. Wyrażenia lambda (podstawy)

Wykład: interfejsy funkcyjne, składnia lambda.

Laboratorium: Zastosowanie lambd do prostych transformacji numerycznych.

11. Strumienie i zastosowania lambd

Wykład: Stream Application Programming Interface (API): map, filter, reduce; strumienie równoległe.

Laboratorium: Analiza wyników symulacji przy użyciu strumieni.

12. Biblioteki naukowe w Javie

Wykład: EJML w większych szczegółach; Apache Commons Math; JavaScript Object Notation (JSON) i Extensible Markup Language (XML).

Laboratorium: Rozwiązywanie układów równań liniowych i parsowanie danych z plików.

13. Wzorce projektowe II i organizacja projektu

	Wykład: Strategy, Composite; modularna struktura projektu w Maven/Gradle. Laboratorium: Zastosowanie wzorca Strategy do wyboru modeli symulacji. 14. Prezentacje projektów studenckich Wykład: Podsumowanie programowania obiektowego (OOP) w Javie i integracja narzędzi. Laboratorium: Prezentacje końcowe projektów wraz z krótkimi wystąpieniami. 15. Podsumowanie i perspektywy Wykład: Kierunki rozwoju w programowaniu (trendy w Javie, współbieżność, styl funkcyjny, programowanie wspomagane sztuczną inteligencją AI). Laboratorium: Dyskusja, jak umiejętności zdobyte na kursie można zastosować w projektach badawczych.
Wymagania wstępne i dodatkowe	Obiektowe języki programowania 1 i 2

Sposoby i kryteria oceniania osiąganych efektów uczenia się

Sposób oceniania (składowe)	Próg zaliczeniowy	Składowa oceny końcowej
zaliczenie laboratorium	50.0%	75.0%
zaliczenie wykładu	50.0%	25.0%

Zalecana lista lektur

Podstawowa lista lektur	1. Joshua Bloch, Java. Efektywne programowanie. Wydanie III, Helion, 2018 2. Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft, Nowoczesna Java w działaniu, Helion, 2019
Uzupełniająca lista lektur	1. Cay S. Horstmann, Java. Podstawy. Wydanie X, Helion, 2016 2. Cay S. Horstmann, Java. Techniki zaawansowane. Wydanie X, Helion 2016 3. Herbert Schildt, Java. Kompendium programisty. Wydanie X, Helion 2018
Adresy eZasobów	Podstawowe https://docs.oracle.com/en/java/ - Strona dokumentacji Oracle Java udostępnia kompleksowe przewodniki deweloperskie, przykłady kodu, dokumentację API i samouczki obejmujące Java SE, Java EE oraz pokrewne technologie, wspierając budowę solidnych aplikacji Java. Uzupełniające https://ejml.org - EJML (Efficient Java Matrix Library) to otwarta biblioteka Java umożliwiająca szybkie i elastyczne operacje macierzowe — dekompozycję, rozwiązywanie układów równań oraz obsługę macierzy rzadkich i gęstych.

Przykładowe zagadnienia/ przykładowe pytania/ realizowane zadania	1.) Otrzymujesz pojedynczą klasę Simulation, która bezpośrednio obsługuje wejście z pliku, przechowywanie danych, obliczenia i wypisywanie wyników. Wskaż co najmniej dwa problemy z takim projektem. Zaproponuj strategię refaktoryzacji przy użyciu oddzielnych klas lub pakietów. 2.) Zadanie programistyczne: Symulacja rozpadu promieniotwórczego Zaimplementuj klasę Particle z atrybutami (id, okres połowicznego rozpadu, stan). Przechowuj cząstki w kolekcji i symuluj rozpad krok po kroku, korzystając z liczb losowych. Obsłuż niepoprawne dane wejściowe za pomocą wyjątków. Dodaj co najmniej jeden test jednostkowy JUnit. Wypisz liczbę nierozpadłych cząstek po każdym kroku.
Praktyki zawodowe w ramach przedmiotu	Nie dotyczy