POLITECHNIKA GDAŃSKA

Karta przedmiotu

Nazwa i kod przedmiotu	Obiektowe języki programowania III, PG_00064060		
Kierunek studiów	Fizyka Techniczna		
Data rozpoczęcia studiów	październik 2026 r.	Rok akademicki realizacji przedmiotu	2028/2029
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							
	Forma zajęć	Wykład	Ćwiczenia	Laboratorium	Projekt	Seminarium	RAZEM
	Liczba godzin zajęć	15.0	0.0	45.0	0.0	0.0	60
	W tym liczba godzin zajęć na odległość: 0.0						

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	60	5.0	60.0	125

Cel przedmiotu	Celem kursu jest wprowadzenie studentów do programowania obiektowego 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. Nacisk zostanie położony na tworzenie niezawodnego, czytelnego i łatwego w utrzymaniu kodu oraz rozwijanie umiejętności przydatnych w większych projektach badawczych i technologicznych.

Efekty uczenia się przedmiotu	Efekt kierunkowy	Efekt z przedmiotu	Sposób weryfikacji i oceny efektu
	[K6_K01] jest gotów do nieustannego uzupełniania wiedzy z zakresu fizyki i nauk pokrewnych, w tym informatyki stosowanej lub energetyki, krytycznej oceny tej wiedzy oraz uznawania jej znaczenia w rozwiązywaniu problemów praktycznych i poznawczych.	jest gotów do samodzielnego i nieustannego poszerzania wiedzy z zakresu programowania obiektowego, krytycznej oceny aktualnych narzędzi i wzorców projektowych oraz uznawania ich znaczenia w rozwiązywaniu problemów naukowych i praktycznych.	[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.	posiada wiedzę z zakresu metod i technik programowania obiektowego oraz zna zasady wykorzystywania wybranych narzędzi informatycznych w rozwiązywaniu problemów z zakresu fizyki i technologii.	[SW1] Ocena wiedzy faktograficznej
	[K6_W01] rozumie cywilizacyjne znaczenie fizyki i jej zastosowań	rozumie znaczenie modelowania prostych układów fizycznych z wykorzystaniem programowania obiektowego oraz zna rolę metod obliczeniowych we wspieraniu zastosowań fizyki w nauce, technologii i rozwoju cywilizacyjnym.	[SW3] Ocena wiedzy zawartej w opracowaniu tekstowym i projektowym
	[K6_U03] posiada umiejętność programowania w wybranym języku oraz stosowania podstawowych pakietów oprogramowania	potrafi tworzyć programy w wybranym języku obiektowym, stosować narzędzia wspierające zarządzanie projektem programistycznym, wykorzystywać frameworki testowe oraz wybrane biblioteki naukowe w rozwiązywaniu problemów z zakresu fizyki i technologii.	[SU1] Ocena realizacji zadania

Treści przedmiotu	Treści przedmiotu - wykład **1. Ekosystem Javy i konfiguracja projektu** Java Virtual Machine (JVM), Java Development Kit (JDK), Zintegrowane Środowiska Programistyczne (IDE); zarządzanie projektem za pomocą Maven i Gradle. **2. Klasy, obiekty i testowanie** Klasy, pola, metody, konstruktory; wprowadzenie do testów jednostkowych w JUnit. **3. Typy proste, klasy opakowujące, tablice i Efficient Java Matrix Library (EJML)** Typy proste kontra obiekty; tablice; klasy opakowujące; pierwsze użycie EJML. **4. Dziedziczenie i interfejsy** Dziedziczenie, przesłanianie metod, klasy abstrakcyjne, interfejsy. **5. Wyjątki i niezawodny kod** Wyjątki sprawdzane i niesprawdzane; strategie obsługi błędów. **6. Framework kolekcji** List, Set, Map; iteratory; kiedy stosować kolekcje. **7. Wzorce projektowe I** Factory, Singleton, Observer (z prostymi ilustracjami w Unified Modeling Language UML). **8. Klasy i metody generyczne oraz kolekcje w praktyce** Klasy i metody generyczne; implementacje kolekcji. **9. Refaktoryzacja i praktyki testowania** spójność, powiązania, zasady SOLID (Single responsibility, Openclosed, Liskov substitution, Interface segregation, Dependency inversion); programowanie sterowane testami (TDD). **10. Wyrażenia lambda (podstawy)** interfejsy funkcyjne, składnia lambda. **11. Strumienie i zastosowania lambd** Stream Application Programming Interface (API): map, filter, reduce; strumienie równoległe. **12. Biblioteki naukowe w Javie** EJML w większych szczegółach; Apache Commons Math; JavaScript Object Notation (JSON) i Extensible Markup Language (XML).

13. Wzorce projektowe II i organizacja projektu.

Strategy, Composite, modularna struktura projektu w Maven/Gradle.

14. Prezentacje projektów studenckich

Podsumowanie programowania obiektowego (OOP) w Javie i integracja narzędzi.

15. Podsumowanie i perspektywy

Kierunki rozwoju w programowaniu (trendy w Javie, współbieżność, styl funkcyjny, programowanie wspomagane sztuczną inteligencją AI).

Treści przedmiotu - laboratoria

1. Ekosystem Javy i konfiguracja projektu

Utworzenie pierwszego projektu Maven; uruchomienie prostego programu związanego z fizyką.

2. Klasy, obiekty i testowanie

Implementacja klasy Particle oraz podstawowe testy jednostkowe.

3. Typy proste, klasy opakowujące, tablice i Efficient Java Matrix Library (EJML)

Operacje wektorowe przy użyciu tablic i EJML.

4. Dziedziczenie i interfejsy

Hierarchia klas dla różnych typów cząstek.

5. Wyjątki i niezawodny kod

Niezawodna obsługa wejścia/wyjścia (I/O) oraz obsługa błędów w prostej symulacji.

6. Framework kolekcji

Przechowywanie i analiza trajektorii cząstek z użyciem kolekcji.

7. Wzorce projektowe I

Implementacja fabryki cząstek i obserwatora do logowania.

8. Klasy i metody generyczne oraz kolekcje w praktyce

Generyczne kontenery na wyniki; wykorzystanie posortowanych zbiorów/map.

9. Refaktoryzacja i praktyki testowania

Refaktoryzacja wcześniejszego kodu i rozszerzenie testów jednostkowych.

---

10. Wyrażenia lambda (podstawy)

Zastosowanie lambd do prostych transformacji numerycznych.

11. Strumienie i zastosowania lambd

Analiza wyników symulacji przy użyciu strumieni.

12. Biblioteki naukowe w Javie

Rozwiązywanie układów równań liniowych i parsowanie danych z plików.

13. Wzorce projektowe II i organizacja projektu.

Zastosowanie wzorca Strategy do wyboru modeli symulacji.

14. Prezentacje projektów studenckich

Prezentacje końcowe projektów wraz z krótkimi wystąpieniami.

15. Podsumowanie i perspektywy

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	

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.

Zajęcia praktyczne w ramach przedmiotu

Nie dotyczy