

**SYLABUS**  
DOTYCZY CYKLU KSZTAŁCENIA 2022-2026  
Rok akademicki 2022-2023 i 2023-2024

**1. PODSTAWOWE INFORMACJE O PRZEDMIOCIE**

Nazwa przedmiotu	<i>programowanie obiektowe</i>
Kod przedmiotu*	
Nazwa jednostki prowadzącej kierunek	<i>Kolegium Nauk Przyrodniczych</i>
Nazwa jednostki realizującej przedmiot	<i>Kolegium Nauk Przyrodniczych</i>
Kierunek studiów	<i>informatyka</i>
Poziom studiów	<i>studia inżynierskie I-go stopnia</i>
Profil	<i>ogólnoakademicki</i>
Forma studiów	<i>stacjonarne</i>
Rok i semestr/y studiów	<i>rok I, II, semestry 2, 3</i>
Rodzaj przedmiotu	<i>przedmiot inżynierski kierunkowy</i>
Język wykładowy	<i>polski</i>
Koordynator	<i>dr inż. Wojciech Koziół</i>
Imię i nazwisko osoby prowadzącej / osób prowadzących	<i>dr inż. Wojciech Koziół, mgr inż. Ewa Żesławska, mgr inż. Wojciech Gałka</i>

\* -opcjonalnie, 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	30			30					5
3	30			45					5

**1.2. Sposób realizacji zajęć**

zajęcia w formie tradycyjnej

**1.3 Forma zaliczenia przedmiotu (z toku)**

zaliczenie z oceną po semestrze 2 i egzamin po semestrze 3

**2. WYMAGANIA WSTĘPNE**

Podstawy programowania
------------------------

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

#### 3.1 Cele przedmiotu

C <sub>1</sub>	Zapoznanie studentów z zagadnieniami dotyczącymi paradygmatu programowania zorientowanego obiektowo.
C <sub>2</sub>	Nauczenie studentów myślenia, projektowania i rozwiązywania problemów przy użyciu obiektów i relacji występujących pomiędzy obiektami. Nauczenie studentów tworzenia prostych programów w języku Java i C#.
C <sub>3</sub>	Zapoznanie studentów z językiem Java i środowiskiem NetBeans oraz językiem C# i środowiskiem Visual Studio.
C <sub>4</sub>	Zapoznanie studentów z zagadnieniami dotyczącymi tworzenia graficznych interfejsów użytkownika i dostępu do relacyjnej bazy danych w języku Java.
C <sub>5</sub>	Nabycie przez studenta umiejętności tworzenia aplikacji z graficznym interfejsem użytkownika w języku Java i C#.
C <sub>6</sub>	Nabycie umiejętności tworzenia aplikacji umożliwiających dostęp do relacyjnych baz danych w języku Java.

#### 3.2 Efekty uczenia się dla przedmiotu

EK (efekt uczenia się)	Treść efektu uczenia się zdefiniowanego dla przedmiotu	Odniesienie do efektów kierunkowych <sup>1</sup>
EK_01	Zna podstawowe konstrukcje programistyczne i struktury danych występujące w językach Java i C#.	K_Wo4, K_Wo7
EK_02	Ma podstawową wiedzę dotyczącą obiektowego paradygmatu programowania i jego zastosowania; rozumie takie pojęcia jak: klasa, klasa abstrakcyjna, interfejs, obiekt, hermetyzacja, dziedziczenie, polimorfizm.	K_Wo4, K_Wo7
EK_03	Ma podstawową wiedzę na temat tworzenia graficznych interfejsów użytkownika oraz interfejsów do łączenia się z relacyjnymi bazami danych i użycia ich w języku Java.	K_Wo4, K_Wo7
EK_04	Potrafi precyzyjnie specyfikować problemy informatyczne i formułować ich rozwiązania w językach Java i C#, wykorzystując poznane techniki programowania zorientowanego obiektowo.	K_U11, K_U12
EK_05	Potrafi stosować podstawowe konstrukcje programistyczne i struktury danych występujące w językach Java i C#. Rozumie ich zalety i wady oraz potrafi odpowiednio je dobrać uwzględniając złożoność, efektywność i jakość utworzonego rozwiązania.	K_U10, K_U12
EK_06	Umie tworzyć proste aplikacje w językach Java i C#.	K_U14

<sup>1</sup> W przypadku ścieżki kształcenia prowadzącej do uzyskania kwalifikacji nauczycielskich uwzględnić również efekty uczenia się ze standardów kształcenia przygotowującego do wykonywania zawodu nauczyciela.

EK_07	Umie zastosować poznane standardowe biblioteki i interfejsy języków Java i C# do tworzenia oprogramowania w paradygmacie obiektowym, w tym w języku Java również biblioteki do tworzenia graficznych interfejsów użytkownika oraz do łączenia się z relacyjnymi bazami danych.	K_U12
-------	--	-------

### 3.3 Treści programowe

#### A. Problematyka wykładu

<b>SEMESTR II</b>
Geneza języka Java. Zasada działania technologii Java.
Ogólna postać programu w języku Java. Praca w środowisku NetBeans.
Identyfikatory, typy, zmienne, wyrażenia, operacje wejścia-wyjścia i komentarze.
Przepływ sterowania programem i sposoby jego modyfikacji. Iteracja i rekurencja w języku Java.
Zmienne tekstowe i operacje na łańcuchach w języku Java.
Obiekty, klasy, pola i metody, konstruktory w języku Java.
Hermetyzacja składowych (w oparciu o język Java).
Dziedziczenie (w oparciu o język Java).
Polimorfizm (w oparciu o język Java).
Klasy abstrakcyjne i interfejsy (w oparciu o język Java).
Składowe statyczne klasy: pola statyczne, metody statyczne, inicjalizatory statyczne (w oparciu o język Java).
Wyjątki w języku Java.
Typy surowe i generyczne w języku Java.
Kolekcje surowe i generyczne w języku Java.
Strumienie i pliki w języku Java.
<b>SEMESTR III</b>
Graficzny interfejs użytkownika – biblioteka SWING.
Graficzny interfejs użytkownika – biblioteka JavaFX.
Łączenie się z relacyjnymi bazami danych poprzez JDBC.
Mapowanie obiektowo-relacyjne w języku Java przy użyciu Hibernate.
Wprowadzenie do technologii JSP.
Wprowadzenie do platformy .NET i języka C#. Porównanie technologii .NET i Java. Obsługa środowiska Microsoft Visual Studio .NET.
Podstawowe typy danych, wyrażenia, instrukcje, tablice jedno i wielowymiarowe, wskaźniki w języku C#.
Programowanie obiektowe w C#: tworzenie klas i obiektów, pola i metody, hermetyzacja, dziedziczenie, abstrakcja, polimorfizm
Obsługa wyjątków w języku C#.
Operacje na strumieniach w C#.
Kolekcje w języku C#.
Tworzenie graficznego interfejsu użytkownika przy użyciu standardowych komponentów .NET.

#### B. Problematyka ćwiczeń laboratoryjnych

<b>SEMESTR II</b>
Wprowadzenie do środowiska NetBeans/IntelliJ IDEA.
Tworzenie i uruchamianie prostych programów w środowisku NetBeans/IntelliJ IDEA.
Przepływ sterowania programem w języku Java – pętle, rekurencje. Wyrażenia warunkowe.
Operacje na zmiennych łańcuchowych w Javie.
Tworzenie klas i obiektów w języku Java.
Kapsułkowanie w języku Java.

Dziedziczenie w języku Java.
Polimorfizm w języku Java.
Klasy i metody abstrakcyjne oraz interfejsy w języku Java.
Składowe statyczne klasy w języku Java.
Rzucanie i obsługa wyjątków w języku Java.
Typy uogólnione w języku Java.
Wykorzystanie kolekcji w języku Java.
Obsługa strumieni w języku Java.
<b>SEMESTR III</b>
Tworzenie GUI w języku Java przy użyciu biblioteki Swing.
Tworzenie GUI w języku Java przy użyciu biblioteki JavaFX.
Obsługa relacyjnych baz danych w języku Java przy użyciu interfejsu JDBC.
Obsługa relacyjnych baz danych przy użyciu frameworka Hibernate.
Tworzenie prostych aplikacji w technologii JSP.
Wprowadzenie do środowiska Visual Studio.
Typy danych, wyrażenia warunkowe i instrukcje sterujące, wskaźniki w języku C#.
Tworzenie klas, obiektów i struktur w języku C#.
Hermetyzacja, dziedziczenie i kompozycja w języku C#.
Polimorfizm i abstrakcja (klasy abstrakcyjne i interfejsy) w języku C#.
Obsługa wyjątków w języku C#.
Operacje na strumieniach w języku C#.
Obsługa kolekcji w języku C#.
Tworzenie GUI i obsługa zdarzeń w języku C#.

### 3.4 Metody dydaktyczne

Wykład: wykład z prezentacją multimedialną.

Laboratoria: tworzenie programów komputerowych, projekt praktyczny w postaci oprogramowania realizowanego głównie jako praca domowa w dużej mierze samodzielna, ale konsultowana.

## 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	egzamin pisemny	w
EK_02	egzamin pisemny	w
EK_03	egzamin pisemny	w
EK_04	kolokwium, projekt	lab
EK_05	kolokwium, projekt	lab
EK_06	kolokwium, projekt	lab
EK_07	projekt	lab

### 4.2 Warunki zaliczenia przedmiotu (kryteria oceniania)

<p><b>LABORATORIUM:</b></p> <p><b>Semestr II</b> Zaliczenie wszystkich efektów uczenia się na ocenę pozytywną. Weryfikowane są efekty EK_04, EK_05, EK_06. Ocena końcowa jest średnią ocen uzyskanych z kolokwiów w założeniu, że student</p>
---

zaliczył wszystkie efekty uczenia się na ocenę pozytywną.

### **Semestr III**

Zaliczenie wszystkich efektów uczenia się na ocenę pozytywną. Weryfikowane są efekty EK\_04, EK\_05, EK\_06, EK\_07. Ocena końcowa jest średnią ocen uzyskanych z projektu oraz kolokwium w założeniu, że student zaliczył wszystkie efekty uczenia się na ocenę pozytywną.

#### **Warunki zaliczenia dla efektów EK\_04, EK\_05, EK\_06:**

##### **Dostateczny:**

Student umie tworzyć proste programy w językach Java i C#. Tworząc programy używa typów prostych, referencyjnych i tablicowych, wyrażeń, instrukcje warunkowych, pętli. Potrafi przetwarzać łańcuchy tekstowe. Umie tworzyć klasy i definiować ich składowe. Potrafi tworzyć obiekty. Umie przeciążać metody i konstruktory. Umie zastosować poznane standardowe biblioteki i interfejsy do tworzenia oprogramowania w paradygmacie obiektowym. Zna pojęcia hermetyzacji, dziedziczenia, kompozycji i polimorfizmu i umie je zastosować w prostych programach. Umie tworzyć i używać klasy abstrakcyjne oraz przesłaniać metody. Zna pojęcie składowych statycznych klasy i potrafi ich użyć. Umie zastosować poznane standardowe biblioteki i interfejsy do tworzenia oprogramowania w paradygmacie obiektowym w tym biblioteki do obsługi kolekcji. Umie obsługiwać standardowe wyjątki.

##### **Dobry:**

Student umie tworzyć bardziej złożone programy w językach Java i C#. Potrafi debugować programy. Zna dobrze podstawy języka Java tj. typy proste, referencyjne i tablicowe, wyrażenia, instrukcje warunkowe, pętle – tworząc programy z łatwością ich używa. Tworzone przez niego programy są napisane starannie i czytelnie. Potrafi przetwarzać łańcuchy tekstowe. Umie tworzyć klasy, pola, metody, konstruktory i obiekty. Umie przeciążać metody i konstruktory. Zna pojęcia hermetyzacji, dziedziczenia, kompozycji, polimorfizmu i stosuje je w swoich programach. Z aplikacji, które tworzy wynika, że dobrze zrozumie istotę tych mechanizmów paradygmatu obiektowego. Umie tworzyć i używać klasy abstrakcyjne i interfejsy oraz przesłaniać metody. Zna pojęcie składowych statycznych klasy i potrafi ich użyć. Umie zastosować poznane standardowe biblioteki i interfejsy do tworzenia oprogramowania w paradygmacie obiektowym, w tym biblioteki do obsługi kolekcji surowych i generycznych. Potrafi obsługiwać standardowe wyjątki. Potrafi tworzyć nowe typy wyjątków i obsługiwać je. Potrafi zapisywać dane do pliku i odczytywać je z pliku w formacie tekstowym i binarnym.

##### **Bardzo dobry:**

Student umie tworzyć złożone programy w językach Java i C#. Wykazuje się w tym zakresie kreatywnością i inwencją twórczą. Potrafi debugować programy i w większości przypadków sam potrafi poprawić błędy zgłaszane przez kompilator. Tworzone przez niego programy są napisane starannie i czytelnie. Zna dobrze podstawy języka Java tj. typy proste, referencyjne i tablicowe, wyrażenia, instrukcje warunkowe, pętle, rekurencję – tworząc programy z łatwością się w nich porusza. Potrafi przetwarzać łańcuchy tekstowe. Umie tworzyć klasy, pola, metody, konstruktory i obiekty. Umie przeciążać metody i konstruktory. Zna pojęcia hermetyzacji, dziedziczenia, kompozycji i polimorfizmu i stosuje je w swoich programach. Wykazuje się głębokim zrozumieniem tych mechanizmów paradygmatu obiektowego. Umie tworzyć i używać klasy abstrakcyjne i interfejsy oraz przesłaniać metody. Zna pojęcie składowych statycznych klasy i potrafi ich użyć. Umie zastosować poznane standardowe biblioteki i interfejsy do tworzenia oprogramowania w paradygmacie obiektowym, w tym biblioteki do obsługi kolekcji surowych i generycznych. Potrafi właściwie dobrać typ kolekcji do rozwiązywanego problemu. Potrafi tworzyć własne typy sparametryzowane. Potrafi przewidzieć i obsługiwać standardowe wyjątki. Potrafi tworzyć nowe typy wyjątków i obsługiwać je. Potrafi zapisywać dane do pliku i odczytywać je z pliku w formacie tekstowym i binarnym.

#### **Warunki zaliczenia dla efektu EK\_07:**

**Dostateczny:**

Student potrafi tworzyć oprogramowanie z graficznym interfejsem użytkownika umożliwiające łączenie się z relacyjnymi bazami danych. Tworzone oprogramowanie jest realizowane w ramach większego projektu wykonywanego poza zajęciami. Student łączy się z bardzo prostą relacyjną bazą danych (jej schemat zawiera przynajmniej jedną tabelę), którą sam zaprojektował i utworzył. Do łączenia się z bazą danych wykorzystuje interfejs JDBC. Student w ramach projektu używa biblioteki SWING do wykonania graficznego interfejsu użytkownika, poprzez który może zarządzać bazą danych.

**Dobry:**

Student potrafi tworzyć oprogramowanie z graficznym interfejsem użytkownika umożliwiające łączenie się z relacyjnymi bazami danych. Tworzone oprogramowanie jest realizowane w ramach większego projektu wykonywanego poza zajęciami. Student łączy się z prostą relacyjną bazą danych (jej schemat zawiera przynajmniej dwie tabele połączone relacją), którą sam zaprojektował i utworzył. Do łączenia się z bazą danych wykorzystuje interfejs JDBC lub framework Hibernate. Student w ramach projektu używa biblioteki JavaFX do wykonania graficznego interfejsu użytkownika, poprzez który może zarządzać bazą danych.

**Bardzo dobry:**

Student potrafi tworzyć oprogramowanie z graficznym interfejsem użytkownika umożliwiające łączenie się z relacyjnymi bazami danych. Tworzone oprogramowanie jest realizowane w ramach większego projektu wykonywanego poza zajęciami. Student łączy się z bardziej złożoną relacyjną bazą danych (jej schemat zawiera przynajmniej trzy tabele połączone relacjami), którą sam zaprojektował i utworzył. Do łączenia się z bazą danych wykorzystuje framework Hibernate. Student w ramach projektu używa biblioteki JavaFX do wykonania graficznego interfejsu użytkownika, poprzez który może zarządzać bazą danych.

**WYKŁAD:****Semestr II**

Test zaliczeniowy z wykładu obejmujący weryfikację efektów EK\_01 i EK\_02. Aby zaliczyć test student musi odpowiedzieć poprawnie na co najmniej 50% pytań dotyczących EK\_01 i na co najmniej 50% pytań dotyczących EK\_02.

**Semestr III**

Pozytywny wynik egzaminu pisemnego z zakresu materiału prezentowanego na wykładzie przy czym:

- Student otrzymuje z egzaminu ocenę **dostateczną**, jeśli wykona co najmniej 50% zadań egzaminacyjnych dotyczących efektu EK\_01 i co najmniej 50% zadań egzaminacyjnych dotyczących efektu EK\_02 oraz co najmniej 50% zadań egzaminacyjnych dotyczących efektu EK\_03.
- Student otrzymuje z egzaminu ocenę **dobrą**, jeśli średnia wykonanych przez niego zadań egzaminacyjnych dotyczących efektów EK\_01, EK\_02, EK\_03 wynosi co najmniej 70%, przy czym każdy z efektów musi być zaliczony na co najmniej 50%.
- Student otrzymuje z egzaminu ocenę **bardzo dobrą**, jeśli średnia wykonanych przez niego zadań egzaminacyjnych dotyczących efektów EK\_01, EK\_02, EK\_03 wynosi co najmniej 90%, przy czym każdy z efektów musi być zaliczony na co najmniej 50%.

Student przystępujący do egzaminu poprawkowego jest zobowiązany do poprawy tylko tych efektów, których nie zaliczył w terminie podstawowym.

W procedurze potwierdzania efektów uczenia się uzyskanych w procesie uczenia się poza systemem studiów istnieje możliwość zaliczenia wykładu z pierwszej części przedmiotu na podstawie posiadanych certyfikatów.

Sposób uznawania osiągnięcia efektów na podstawie certyfikatów:

Java SE 8 Programmer I – zalicza efekty EK\_01 i EK\_02 z pierwszego semestru przedmiotu (w odniesieniu do języka Java) na ocenę dostateczną.

Java SE 8 Programmer I + Java SE 8 Programmer II – zalicza efekty EK\_01 i EK\_02 z pierwszego semestru przedmiotu (w odniesieniu do języka Java) na ocenę dobrą.

Java SE 11 Developer – zalicza efekty EK\_01 i EK\_02 z pierwszego semestru przedmiotu (w odniesieniu do języka Java) na ocenę bardzo dobrą.

Java SE 17 Developer – zalicza efekty EK\_01 i EK\_02 z pierwszego semestru przedmiotu (w odniesieniu do języka Java) na ocenę bardzo dobrą.

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

Forma aktywności	Średnia liczba godzin na zrealizowanie aktywności
Godziny kontaktowe wynikające z harmonogramu studiów	135
Inne z udziałem nauczyciela akademickiego (udział w konsultacjach, egzaminie)	3
Godziny niekontaktowe – praca własna studenta (przygotowanie do zajęć, egzaminu, napisanie referatu itp.)	112
SUMA GODZIN	250
<b>SUMARYCZNA LICZBA PUNKTÓW ECTS</b>	10

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

## 6. PRAKTYKI ZAWODOWE W RAMACH PRZEDMIOTU

wymiar godzinowy	-
zasady i formy odbywania praktyk	-

## 7. LITERATURA

Literatura podstawowa:

1. C. S. Horstmann: Core Java 2. [T. 1], Podstawy, Gliwice, Helion, 2003
2. C. S. Horstmann: Core Java 2. [T. 2], Techniki zaawansowane, Gliwice, Helion, 2003
3. B. Eckel: Thinking in Java : edycja polska, Gliwice, Helion, 2006
4. H. Schild: Java : kompendium programisty, Gliwice, Helion, 2012
5. M. Lis: Java. Ćwiczenia praktyczne. Wyd. 3, Gliwice, Helion, 2011
6. M. Lis, Java : praktyczny kurs, Gliwice, Helion, 2015
7. W. Rychlicki: Programowanie w języku Java : zbiór zadań z (p)odpowiedziami, Gliwice, Helion, 2012
8. M. Hall, L. Brown: Java Servlet i JavaServer Pages. T. 1, Gliwice, Helion, 2006
9. A. Hemrajani: Java : tworzenie aplikacji sieciowych za pomocą Springa, Hibernate i Eclipse [tł. z ang. K. Rymczak, E. Bucka], Gliwice, Helion, 2007
10. J. Matulewski: C# : lekcje programowania : praktyczna nauka programowania dla platform .NET i .NET Core, Gliwice, Helion 2021

11. A. Troelsen ; tł. [z ang.] Maciej Baranowski: Język C# 2010 i platforma .NET 4 , Warszawa, Wydawnictwo Naukowe PWN, 2011

12. M. Lis: C# : praktyczny kurs, Gliwice, Helion, 2021

Literatura uzupełniająca:

1. M. Lis: Java: ćwiczenia zaawansowane, Gliwice, Helion, 2012

2. B. J. Evans, D. Flanagan: Java w pigułce, Gliwice, 2015

3. J. Bloch; tłum. Rafał Jońca: Java : efektywne programowanie. Wyd. 3, Gliwice, Helion, 2018

4. I. Griffiths ; tł. Piotr Rajca: C# 5.0. Programowanie : tworzenie aplikacji Windows 8, internetowych oraz biurowych w .NET 4.5 Framework, Gliwice, Helion, 2013

5. K. Rychlicki-Kicior: C# : tworzenie aplikacji graficznych w.NET 3.0, Gliwice, Helion, 2007

6. S. Perkins : Hibernate search : skuteczne wyszukiwanie [tł. A. Bobak], Gliwice, Helion, 2014