Załącznik nr 1.5 do Zarządzenia Rektora UR nr 7/2023

SYLABUS

DOTYCZY CYKLU KSZTAŁCENIA *2025-2029*
(skrajne daty)
Rok akademicki 2025/2026 i 2026/2027

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 Informatyki
Kierunek studiów	Informatyka i ekonometria
Poziom studiów	studia pierwszego stopnia
Profil	praktyczny
Forma studiów	studia stacjonarne
Rok i semestr/y studiów	rok I i II, semestr 2, 3
Rodzaj przedmiotu	przedmiot kierunkowy
Język wykładowy	polski
Koordynator	dr inż. Wojciech Kozioł
Imię i nazwisko osoby prowadzącej / osób prowadzących	dr inż. Wojciech Kozioł, mgr inż. Marcin Mrukowicz, mgr inż. Wojciech Gałka

***** *-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	15			15					3

1.2. Sposób realizacji zajęć

☒ zajęcia w formie tradycyjnej
☐ zajęcia realizowane z wykorzystaniem metod i technik kształcenia na odległość

1.3 Forma zaliczenia przedmiotu (z toku) (egzamin, zaliczenie z oceną, zaliczenie bez oceny)

ZALICZENIE Z OCENĄ PO SEM. 2 ORAZ EGZAMIN PO SEM. 3

2.WYMAGANIA WSTĘPNE

Podstawy programowania

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

3.1 Cele przedmiotu

C1	Zapoznanie studentów z zagadnieniami dotyczącymi paradygmatu programowania zorientowanego obiektowo.
C2	Nauczenie studentów myślenia, projektowania i rozwiązywania problemów przy użyciu obiektów i relacji występujących pomiędzy obiektami. Nauczenie studentów tworzenia prostych programów w języku Java.
C3	Zapoznanie studentów z językiem Java i środowiskiem do tworzenia aplikacji w języku Java.
C4	Zapoznanie studentów z zagadnieniami dotyczącymi tworzenia graficznych interfejsów użytkownika i dostępu do relacyjnej bazy danych w języku Java.
C5	Nabycie przez studenta umiejętności tworzenia aplikacji z graficznym interfejsem użytkownika w języku Java.
C6	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 Student:	Odniesienie do efektów kierunkowych [1]
EK_01	Zna dobrze podstawowe konostrukcje programistyczne i struktury danych występujące w języku Java oraz środowisko NetBeans lub IntelliJ IDEA.	K_W03
EK_02	Ma wiedzę dotyczącą obiektowego paradygmatu programowania i jego zastosowania; rozumie takie pojęcia jak: klasa, klasa abstrakcyjna, interfejs, obiekt, hermetyzacja, dziedziczenie, polimorfizm.	K_W03
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_W03
EK_04	Umie właściwie zaprojektować i tworzyć aplikacje w języku Java zarówno proste jak i nieco bardziej złożone aplikacje w języku Java wykorzystując również poznane standardowe biblioteki dostępne w środowisku Java. Potrafi dokonać ich analizy i ocenić poprawność działania..	K_U01
EK_05	Umie zaprojektować i tworzyć w języku Java aplikacje z graficznym interfejsem użytkownika, który umożliwia połączenie się z relacyjną bazą danych i wykorzystanie jej w realizowanej aplikacji. Wykorzystuje w tym celu poznane biblioteki i narzędzia informatyczne. Potrafi utworzyć dokumentację dla utworzonej aplikacji.	K_U11

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

3.3 Treści programowe

A. Problematyka wykładu

SEM 2
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 i kompozycja (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.
SEM 3
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 JSP

B. Problematyka ćwiczeń, laboratoriów

SEM II
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 i kompozycja 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.
SEM III
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 przy użyciu JSP.

3.4 Metody dydaktyczne

Wykład: wykład z prezentacją multimedialną.
Laboratoria: tworzenie programów komputerowych w oparciu o treści zadań zawartych w konspektach laboratoryjnych, 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	projekt	lab

4.2 Warunki zaliczenia przedmiotu (kryteria oceniania)

- Zaliczenie efektu EK_04 na ocenę pozytywną – weryfikowanego poprzez kolokwium.
- Zaliczenie projektu na ocenę pozytywną. Weryfikowane są: EK_04, EK_05, przy czym do zaliczenia projektu wymagane jest aby obydwa efekty uczenia się były zaliczone na ocenę pozytywną.

Ocena końcowa jest średnią ocen uzyskanych z projektu oraz kolokwium, przy czym obydwie oceny muszą być pozytywne.

Warunki zaliczenia dla efektu EK_04:

Dostateczny:
Student umie tworzyć proste programy w języku Java. 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 języku Java. 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 obsłużyć 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 języku Java. 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 obsłużyć 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_05:

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:
1. 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 z harmonogramu studiów	90
Inne z udziałem nauczyciela akademickiego	5

(udział w konsultacjach, egzaminie)	
Godziny niekontaktowe – praca własna studenta (przygotowanie do zajęć, egzaminu, napisanie referatu itp.)	105
SUMA GODZIN	200
SUMARYCZNA LICZBA PUNKTÓW ECTS	8

** 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
Literatura uzupełniająca: 1. M. Lis: Java: ćwiczenia zaawansowane, Gliwice, Helion, 2012 2. B. J. Evans, D. Flanagan: Java w pigułce, Gliwice, Helion, 2015 3. J. Bloch; tłum. Rafał Jońca: Java : efektywne programowanie. Wyd. 3, Gliwice, Helion, 2018 4. S. Perkins : Hibernate search : skuteczne wyszukiwanie [tł. A. Bobak], Gliwice, Helion, 2014

Akceptacja Kierownika Jednostki lub osoby upoważnionej