Nazwa przedmiotu						
Paradygmaty programowania Programming paradigms						
Dyscyplina					Oznaczenie przedmiotu	
Informatyka					5K_INS1	
Rodzaj przedmiotu	Stopień studiów	Tryb studiów	Język zajęć		Rok	Semestr
obowiązkowy	1	niestacjonarne	polski		II	IV
Rodzaj zajęć		Wyk.	Ćw.	Lab.	Sem.	Proj.	Liczba punktów ECTS
Liczba godzin w semestrze		18	0	18	0	0	6
Koordynator	Dr inż. Jacek Łyp, jackrat@el.pcz.czest.pl						
Prowadzący	Dr inż. Jacek Łyp, jackrat@el.pcz.czest.pl Dr inż. Piotr Szeląg, szelag@el.pcz.czest.pl						

I. KARTA PRZEDMIOTU

Cel przedmiotu

C1. Zapoznanie z najistotniejszymi paradygmatami współczesnego programowania (imperatywnym, strukturalnym, proceduralnym, obiektowym, funkcyjnym i logicznym).
C2. Zrozumienie sposobu działania i istotnej różnorodności implementacji języków programowania.
C3. Opanowanie zasad programowania obiektowego.
C4. Nabycie praktycznej umiejętności projektowania i implementacji prostych aplikacji z graficznym interfejsem użytkownika w środowisku programistycznym zorientowanym obiektowo.

Wymagania wstępne w zakresie wiedzy, umiejętności i innych kompetencji

1. Umiejętność obsługi komputera.
2. Umiejętność korzystania ze źródeł literaturowych i zasobów internetowych.
3. Znajomość podstaw programowania w zakresie ogólnej wiedzy o arytmetyce komputerów, podstawowych typach danych i instrukcjach sterujących (instrukcje podstawienia, warunkowe, pętle).

Efekty kształcenia

EK1. Student potrafi ocenić przydatność wybranych paradygmatów i związanych z nimi środowisk programistycznych do rozwiązywania różnego typu problemów.
EK2. Student projektuje, implementuje i wykorzystuje klasy zawierające pola, właściwości, metody, konstruktory, destruktory, delegacje, zdarzenia, wykorzystując mechanizm dziedziczenia, polimorfizmu, hermetyzacji, interfejsy.
EK3. Student projektuje i realizuje proste aplikacje w środowisku programistycznym zorientowanym obiektowo, wykorzystując podstawowe kontrolki graficznego interfejsu użytkownika.

Treści programowe: wykłady		Liczba godzin
W 1 -	Wprowadzenie do środowiska Visual Studio i języka C#.	1
W 2 -	Imperatywne paradygmaty programowania: programowanie strukturalne i proceduralne.	1
W 3 -	Deklaratywne paradygmaty programowania: programowanie logiczne, funkcyjne. Rachunek lambda, LINQ. Programowanie sterowane zdarzeniami. Programowanie ekstremalne.	2
W 4 -	Paradygmat programowania obiektowego. Klasy i obiekty. Składniki klas: pola i metody.	2
W 5 -	Składniki klas: delegacje i zdarzenia.	2
W 6 -	Ograniczanie dostępu do składników klas. Implementacja właściwości. Hermetyzacja.	2
W 7 -	Metody statyczne. Konstruktory i destruktory. Przeładowywanie metod i operatorów.	2
W 8 -	Dziedziczenie, metody wirtualne, polimorfizm. Klasy abstrakcyjne i interfejsy.	2
W 9 -	Obsługa wyjątków.	1
W 10 -	Wielowątkowość. Programowanie współbieżne, równoległe, rozproszone.	2
Test zaliczeniowy		1
	SUMA	**18**

Treści programowe: laboratorium		Liczba godzin
L 1 -	Środowisko programistyczne Visual Studio – składniki i obsługa; uruchamianie, testowanie i debugowanie prostych programów; elementy języka C# (deklaracje, instrukcje sterujące, tablice, funkcje procedury, przekazywanie parametrów).	2
L 2 -	Wykorzystywanie podstawowych składników interfejsu graficznego VS (formularze, pola edycyjne, etykiety, przyciski, listy wyboru, pola wyboru, panele, dialogi komunikatów); model programowania sterowanego zdarzeniami.	2
L 3 -	Implementacja klas z polami i metodami (funkcyjnymi i proceduralnymi), przekazywanie parametrów do metod (przez wartość i przez referencję). Przeciążanie metod i operatorów.	2

L 4 -	Implementacja konstruktorów i destruktorów	2
L 5 -	Implementacja właściwości klas. Wykorzystanie hermetyzacji. Implementacja delegacji i zdarzeń.	2
L 6 -	Implementacja klas wykorzystujących dziedziczenie; metody wirtualne; realizacja polimorfizmu.	2
L 7 -	Realizacja programowej kontroli wyjątków.	2
L 8 -	Programowanie aplikacji wielowątkowych.	2
Test zaliczeniowy		2
	SUMA	**18**

Narzędzia dydaktyczne

1. Prezentacja multimedialna
2. Tablica interaktywna
3. Specjalistyczne oprogramowanie
4. Stanowiska komputerowe w laboratorium

Sposoby oceny efektów kształcenia (F – ocena Formująca, P – ocena Podsumowująca)

P1. Pisemny test zaliczeniowy. (100% końcowej oceny z wykładu).
P2. Laboratorium – wykonanie zadań programistycznych na bieżących zajęciach (50% oceny końcowej).
P3. Laboratorium - praktyczny test zaliczeniowy – (50% oceny końcowej).

Obciążenie pracą studenta

Forma aktywności	Średnia liczba godzin na zrealizowanie aktywności
Godziny kontaktowe z prowadzącym	36
Zapoznanie się ze wskazanymi źródłami	24
Opanowanie obsługi środowisk programistycznych	15
Przygotowanie do zajęć laboratoryjnych	30
Przygotowanie do testu z wykładu	15
Przygotowanie do testu z laboratorium	15
Sumaryczna liczba godzin/punktów ECTS dla przedmiotu	**135 / 6**

Wykaz literatury podstawowej i uzupełniającej

1. Jarosław Bylina, Beata Bylina. Przegląd języków i paradygmatów programowania. UMCS Lublin 2011.
2. Microsoft C#. Specyfikacja języka. Microsoft Press.
3. Beata Pańczyk, Marcin Badurowicz. Programowanie obiektowe. Język C#. Politechnika Lubelska. Lublin 2013.
4. Jesse Liberty. C#. Programowanie. O'Reilly, Helion 2015.
5. Bertrand Meyer. Programowanie zorientowane obiektowo. Helion 2005.

Macierz realizacji efektów kształcenia

Efekt kształcenia	Odniesienie efektu do efektów kształcenia dla dyscypliny naukowej Elektrotechnika*	Cele przedmiotu	Forma zajęć	Narzędzia dydaktyczne	Sposób oceny
EK1	KINF1_W07 KINF1_U13	C1, C2	W, L	1, 2, 3, 4	P1
EK2	KINF1_W07 KINF1_U09	C1, C2, C3	W, L	1, 2, 3, 4	P1, P2, P3
EK3	KINF1_W08 KINF1_W11	C3, C4	W, L	1, 2, 3, 4	P2, P3

* – wg załącznika

II. FORMY OCENY – SZCZEGÓŁY

Ocena	Efekty
EK1	**Student potrafi ocenić przydatność wybranych paradygmatów i związanych z nimi środowisk programistycznych do rozwiązywania różnego typu problemów.**
2	Student nie potrafi ocenić przydatności wybranych paradygmatów i związanych z nimi środowisk programistycznych do rozwiązywania różnego typu problemów.
3	Student potrafi wymienić i scharakteryzować cechy podstawowych paradygmatów programowania: imperatywnego, proceduralnego, strukturalnego, obiektowego, deklaratywnego, funkcyjnego i logicznego.
3.5	Student potrafi wymienić i przypisać popularnym środowiskom i/lub językom programowania realizowane tam paradygmaty.
4	Student potrafi wytłumaczyć pojęcie architektury von Neumanna i rozumie jej współczesne znaczenie. Student potrafi podać przykłady odstępstw od ścisłych paradygmatów realizowanych w różnych współczesnych językach programowania.
4.5	Student potrafi zilustrować cechy znanych paradygmatów programowania (min 5) fragmentami kodów źródłowych.
5	Student potrafi wymienić i scharakteryzować cechy, wady i zalety innych (tzw. niszowych) paradygmatów programowania (min 5).
EK2	**Student projektuje, implementuje i wykorzystuje klasy zawierające pola, właściwości, metody, konstruktory, destruktory, delegacje, zdarzenia, wykorzystując mechanizm dziedziczenia, polimorfizmu, hermetyzacji, interfejsy.**
2	Student nie potrafi zaprojektować, zaimplementować klasy zawierającej pola, właściwości, metody, konstruktor, destruktor, delegacje,

	zdarzenia, z wykorzystaniem mechanizmu dziedziczenia, polimorfizmu, hermetyzacji.
3	Student potrafi zaprojektować i zaimplementować klasę zawierającej pola, właściwości, metody funkcyjne i proceduralne, konstruktor.
3.5	Student potrafi zaprojektować i zaimplementować klasę zawierającą właściwości o zróżnicowanym poziomie złożoności i hermetyzacją
4	Student potrafi zaprojektować i zaimplementować klasę wykorzystaniem dziedziczenia i polimorfizmu.
4.5	Student potrafi zaprojektować i zaimplementować klasy z wykorzystaniem interfejsów i klas abstrakcyjnych.
5	Student potrafi zaprojektować i zaimplementować klasę z selektywnie obsługiwanymi zdarzeniami.
EK3	**Student projektuje i realizuje proste aplikacje w środowisku programistycznym zorientowanym obiektowo, wykorzystując podstawowe kontrolki graficznego interfejsu użytkownika.**
2	Student nie potrafi zaprojektować i zrealizować prostej aplikacji w środowisku programistycznym zorientowanym obiektowo, wykorzystując podstawowe kontrolki graficznego interfejsu użytkownika.
3	Student potrafi stworzyć aplikację z własnym GUI opartą na obsłudze kluczowych zdarzeń minimum pięciu podstawowych kontrolek oferowanych przez środowisko.
3.5	Student potrafi oprogramować tworzenie kontrolek różnych typów w trakcie działania programu, inicjując dla nich programowo kluczowe właściwości i obsługę kluczowych zdarzeń.
4	Student potrafi zaimplementować programową walidację interfejsu użytkownika.
4.5	Student potrafi testować i debugować aplikację wykorzystując oferowane przez środowisko programistyczne narzędzia takie jak pułapki i praca krokowa.
5	Student potrafi zaimplementować obsługę wyjątków aplikacji.

III. INNE PRZYDATNE INFORMACJE O PRZEDMIOCIE

1. Wszelkie informacje dla studentów na temat planu zajęć dostępne są na tablicy ogłoszeń oraz na stronie el.pcz.pl.
2. Informacje na temat warunków zaliczania zajęć przekazywana jest studentom podczas pierwszych zajęć.