**KONSPEKT PRZEDMIOTU**

Semestr letni, rok akad. 2020/2021

|  |  |
| --- | --- |
| Nazwa przedmiotu | Projektowanie systemów informatycznych |
| Kierunek/-i studiów / rok studiów / semestr studiów | Informatyka / III rok / 6 |
| Tryb studiów | Stacjonarny |
| Forma zajęć | Laboratorium |
| Liczba godzin | 30 |
| Koordynator przedmiotu: | Inż. Gabriela Miller |
| Jakie są ogólne cele dydaktyczne tego przedmiotu? | Celem przedmiotu jest nabycie umiejętności tworzenia systemów informatycznych i znajomość wzorców projektowych i architektonicznych systemów informatycznych. Student będzie posiadał podstawową wiedzę na temat refaktoryzacji kodu. |
| Jak są kryteria zaliczenia tego przedmiotu? | Kryterium zaliczenia jest implementacja architektonicznych i projektowych wzorców systemów informatycznych oraz posiadanie teoretycznej wiedzy w tej dziedzinie. |
| Jakie są kryteria zaliczenia tego przedmiotu na ocenę celującą? | Kryterium zaliczenia przedmiotu na ocenę celującą jest wykonanie dodatkowego projektu dotyczącego tematu refaktoryzacji kodu. |
| Pozostałe informacje, dotyczące tego przedmiotu, ważne dla studenta |  |

|  |  |
| --- | --- |
| **ZAJĘCIA 1**  **(3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość wzorców projektowych systemów informatycznych  2. Charakterystyka wzorca projektowego  3. Znajomość definicji wzorca projektowego  4. Zalety i wady użycia wzorca projektowego |
| Treści zajęć | 1. Definicja wzorca projektowego  2. Podstawowe elementy wzorca projektowego  3. Charakterystyka wzorca projektowego  4. Przedstawienie wszystkich wzorców projektowych  5. Podział wzorców projektowych  6. Zalety wykorzystywania wzorców projektowych w inżynierii oprogramowania |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  “Design Patterns Elements of Reusable Object-Oriented Software” - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides  Rozszerzające / uzupełniające:  Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów - Robert C. Martin  Pattern-Oriented Software Architecture, A System of Patterns - Regine Meunier, Hans Rohnert, Frank Buschmann, Michael Stal, Peter Sommerlad |

|  |  |
| --- | --- |
| **ZAJĘCIA 2**  **(3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość cech wzorców projektowych : singleton oraz factory method  2. Znajomość zalet i wad wykorzystania powyższych wzorców projekt  3. Umiejętność implementacji powyższych wzorców |
| Treści zajęć | 1. Przedstawienie definicji wzorców projektowych : singleton oraz factory method  2. Charakterystyka wzorców projektowych singleton oraz factory method  3. Zalety oraz wady wzorców projektowych singleton oraz factory method  4. Prezentacja przykładowych implementacji wzorców projektowych : singleton oraz factory method przez prowadzącego |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  “Design Patterns Elements of Reusable Object-Oriented Software” - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides  Rozszerzające / uzupełniające:  https://refactoring.guru/design-patterns/singleton  https://refactoring.guru/design-patterns/factory-method  https://www.dofactory.com/net/factory-method-design-pattern |

|  |  |
| --- | --- |
| **ZAJĘCIA 3**  **(3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość cech wzorców projektowych : adapter oraz facade  2. Znajomość zalet i wad wykorzystania powyższych wzorców projektowych  3. Umiejętność implementacji powyższych wzorców |
| Treści zajęć | 1. Przedstawienie definicji wzorców projektowych : adapter oraz facade  2. Charakterystyka wzorców projektowych adapter oraz facade  3. Zalety oraz wady wzorców projektowych adapter oraz facade  4. Prezentacja przykładowych implementacji wzorców projektowych : adapter oraz facade przez prowadzącego  5. Prezentacja implementacji wzorców wykonanych przez studentów |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  “Design Patterns Elements of Reusable Object-Oriented Software” - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides  Rozszerzające / uzupełniające:  https://refactoring.guru/design-patterns/adapter  https://refactoring.guru/design-patterns/facade |

|  |  |
| --- | --- |
| **ZAJĘCIA 4**  **(3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość cech wzorców projektowych : iterator oraz template method  2. Znajomość zalet i wad wykorzystania powyższych wzorców projektowych  3. Umiejętność implementacji powyższych wzorców |
| Treści zajęć | 1. Przedstawienie definicji wzorców projektowych : iterator oraz template method  2. Charakterystyka wzorców projektowych iterator oraz template method  3. Zalety oraz wady wzorców projektowych iterator oraz template method  4. Prezentacja przykładowych implementacji wzorców projektowych : iterator oraz template method przez prowadzącego  5. Prezentacja implementacji wzorców wykonanych przez studentów |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  “Design Patterns Elements of Reusable Object-Oriented Software” - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides  Rozszerzające / uzupełniające:  https://refactoring.guru/design-patterns/iterator  https://refactoring.guru/design-patterns/template-method |

|  |  |
| --- | --- |
| **ZAJĘCIA 5**  **(3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość wzorców architektonicznych dla systemów informatycznych  2. Umiejętność rozróżnienia poszczególnych wzorców architektonicznych  3. Umiejętność implementacji niektórych wzorców architektonicznych |
| Treści zajęć | 1. Przedstawienie definicji wzorca architektonicznego  2. Charakterystyka wzorców architektonicznych  3. Zalety oraz wady wzorców architektonicznych  4. Prezentacja przykładowych implementacji wzorców architektonicznych  5. Prezentacja implementacji wzorców wykonanych przez studentów |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  „Podstawy architektury oprogramowania dla inżynierów” - Mark Richards, Neal Ford  <https://www.ou.nl/documents/40554/791670/IM0203_03.pdf/30dae517-691e-b3c7-22ed-a55ad27726d6>  Rozszerzające / uzupełniające:  „Pattern-Oriented Software Architeture. A Pattern language for Distributed Computing” – Frank Buschmann, Kevlin Henney, Douglas C. Schmidt |

|  |  |
| --- | --- |
| **ZAJĘCIA 6**  **(3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość cech wzorca architektonicznego MVC  2. Znajomość zalet i wad wykorzystania powyższego wzorca  3. Umiejętność implementacji powyższego wzorca przy użyciu narzędzia Visual Studio i języka C# |
| Treści zajęć | 1. Przedstawienie definicji wzorca MVC  2. Charakterystyka wzorca MVC  3. Zalety oraz wady wzorca architektonicznego MVC  4. Prezentacja przykładowej implementacji MVC  5. Prezentacja implementacji wzorców wykonanych przez studentów |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  ASP.NET Core MVC 2. Zaawansowane programowanie. Wydanie VII - Adam Freeman  Rozszerzające / uzupełniające:  https://dotnet.microsoft.com/apps/aspnet/mvc |

|  |  |
| --- | --- |
| **ZAJĘCIA 7**  **(3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość cech wzorca architektonicznego SOA  2. Znajomość zalet i wad wykorzystania powyższego wzorca  3. Znajomość wzorca WCF do tworzenia aplikacji usługowej  4. Umiejętność implementacji powyższego wzorca przy użyciu narzędzia Visual Studio i języka C# w formie WCF |
| Treści zajęć | 1. Przedstawienie definicji wzorca SOA  2. Charakterystyka wzorca SOA  3. Zalety oraz wady wzorca architektonicznego SOA  4. Prezentacja przykładowej implementacji SOA w formie WCF  5. Prezentacja implementacji wzorców wykonanych przez studentów |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  „Wzorce SOA” - Arnon Rotem-Gal-Oz  <https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf>  Rozszerzające / uzupełniające:  https://docs.microsoft.com/en-us/dotnet/framework/wcf/getting-started-tutorial |

|  |  |
| --- | --- |
| **ZAJĘCIA 8**  **( 3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość cech wzorca architektury trójwarstwowej  2. Znajomość zalet i wad wykorzystania powyższego wzorca  3. Umiejętność implementacji powyższego wzorca przy użyciu dowolnych narzędzi programistycznych |
| Treści zajęć | 1. Przedstawienie definicji wzorca architektury trójwarstwowej  2. Charakterystyka wzorca architektury trójwarstwowej  3. Zalety oraz wady wzorca architektonicznego SOA  4. Prezentacja implementacji wzorców wykonanych przez studentów |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  <https://www.ibm.com/cloud/learn/three-tier-architecture>  „Patterns of Enterprise Application Architecture” – M. Fowler  Rozszerzające / uzupełniające:  https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648105(v=pandp.10) |

|  |  |
| --- | --- |
| **ZAJĘCIA 9**  **(3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Znajomość cech wzorca architektury trójwarstwowej  2. Znajomość zalet i wad wykorzystania powyższego wzorca  3. Umiejętność implementacji powyższego wzorca przy użyciu dowolnych narzędzi programistycznych |
| Treści zajęć | 1. Prezentacja implementacji wzorca architektury trójwarstwowej przez prowadzącego  2. Analiza i możliwość rozszerzenia pokazanej aplikacji i wyszukania jej błędów  3. Prezentacja implementacji wzorców wykonanych przez studentów |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  <https://en.wikipedia.org/wiki/Multitier_architecture>  „Patterns of Enterprise Application Architecture” – M. Fowler  Rozszerzające / uzupełniające:  https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff647095(v=pandp.10) |

|  |  |
| --- | --- |
| **ZAJĘCIA 10**  **( 3 godz.)** | |
| Po tych zajęciach student (będzie znał/ wiedział/ umiał/ potrafił / rozumiał) | Efekty uczenia się:  1. Umiejętność rozróżnienia wzorca architektonicznego od wzorca projektowego  2. Umiejętność refaktoryzacji kodu źródłowego  3. Znajomość metody do refaktoryzacji kodu źródłowego |
| Treści zajęć | 1. Różnice pomiędzy wzorcami architektonicznymi a wzorcami projektowymi  2. Metody refaktoryzacji kodu  3. Zalety i wady refaktoryzacji kodu  4. Prezentacja przykładowych metod refaktoryzacji kodu w narzędziu programistycznym przez prowadzącego |
| Źródło/a do nauki treści z tych zajęć | Minimalne / obowiązkowe:  „Refactoring. Improving the Design of Existing Code” – Martin Fowler  <https://en.wikipedia.org/wiki/Code_refactoring>  Rozszerzające / uzupełniające:  „Czysty kod. Podręcznik dobrego programisty” - Robert C. Martin |