КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра теорії та технології програмування

«ЗАТВЕРДЖУЮ»
Заступник декана
з навчальної роботи

_____Кашпур О.Ф.

«____»_____2016 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ[1]

"ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

(повна назва навчальної дисципліни)

для студентів денної форми навчання

напрям підготовки / спеціальність **__113 «Прикладна математика»**
(шифр і назва напряму підготовки)

Освітня програма: Бізнес-інформатика
(назва спеціалізації)

Пролонговано: на 2017/18 н.р. _____(Кузенко В.Ф.) 01.09.2017р.
(підпис, ПІБ, дата)

КИЇВ – 2016

[1] Робоча програма навчальної дисципліни є нормативним документом вищого навчального закладу і містить виклад конкретного змісту навчальної дисципліни, послідовність, організаційні форми її вивчення та їх обсяг, визначає форми та засоби поточного і підсумкового контролів.

Робоча програма __з дисципліни “Інформаційні системи та технології”_ для студентів *напряму підготовки* Методи та системи прийняття рішень**,** *спеціальності* 8.04030203 “Соціальна інформатика”

«01» 09. 2016 року – 16 с.

Розробники[2]: (вказати авторів, їхні посади, наукові ступені та вчені звання):

доцент Кузенко Володимир Федорович, кандидат фіз.-мат. наук, доцент

Робоча програма дисципліни “Інформаційні системи та технології”
затверджена на засіданні кафедри _теорії та технології програмування

Протокол № 10 від “17” 06 2016 року.

Завідувач кафедри теорії та технології програмування_____
(вибрати необхідне)

_____(Нікітченко М.С.)
(підпис) (прізвище та ініціали)

“17” 06 2016 року

Схвалено науково - методичною комісією факультету комп’ютерних наук та кібернетики

_____

Протокол № ___ від “___” _____ 2016 року.

Голова науково-методичної комісії _____ (Хусаїнов Д.Я.)
(підпис) (прізвище та ініціали)

«_____» _____ 2016 року

Робоча програма затверджена на засіданні Вченої Ради факультету комп’ютерних наук та кібернетики.

Протокол №12 від “18” 06 2016 року.

[2] Розробляється лектором. Робоча програма навчальної дисципліни розглядається на засіданні кафедри (циклової комісії – для коледжів), науково-методичної комісії факультету/інституту (раді навчального закладу - коледжу), підписується завідувачем кафедри (головою циклової комісії), головою науково-методичної комісії факультету/інституту (головою ради) і затверджується заступником декана/директора інституту з навчальної роботи (заступником директора коледжу).

ВСТУП

Навчальна дисципліна "Інформаційні системи та технології"
(назва дисципліни)

є складовою освітньо-професійної програми підготовки фахівців за освітньо-кваліфікаційним рівнем «магістр» галузі знань _____
(зазначити відповідний рівень) (зазначити)

з *напряму підготовки* _____, *спеціальності* - 113 «Прикладна математика»
(шифр і назва напряму підготовки) (шифр і назва спеціальності)

Дана дисципліна нормативна
(нормативна, за вибором)

за ***спеціальністю*** "Прикладна математика" .
(зазначити спеціальність, у випадку дисципліни спеціалізації відповідну спеціалізацію)

Викладається у **1** семестрі **1** курсу **магістратури** в **обсязі – 90 год.**[3]
(зазначається **загальний** обсяг)

(3 кредитів ECTS[4]) зокрема: *лекції – 16 год, лабораторні – 12 год., самостійна робота – 60 год.* У курсі передбачено **1** *змістовий модуль* та *1 модульна(і) контрольна(і) робота(и)*. Завершується дисципліна – **іспитом.**

Мета дисципліни – отримання знань та оволодіння навичками використання інформаційних технологій розробки програмних систем, зокрема, навичками програмної інженерії, застосування моделювання, *CASE*-технологій, у тому числі при розробці розподілених програмних систем, *Web*-систем, а також систем із сервісно-орієнтованою архітектурою.

Завдання – вивчення базових понять та оволодіння навичками моделювання програмних систем (із використанням мови *UML*), інженірингу та реінженірингу програмних систем, *.NET*- та *Java*-технологій, технологій розподілених систем (*NET-Remoting*, *Java-RMI*), поняття сервісно-орієнтованої архітектури, зокрема, технології *Web*-сервісів та сервісів *WCF*, технологій розробки *Web*-проектів із використанням *.NET*- та *Java*-фреймворків, технологій розробки мобільних проектів, хмарних обчислень та технологій.

Структура курсу

В результаті вивчення навчальної дисципліни студент повинен:

знати: основні поняття об'єктно-орієнтованої (ОО) парадигми, принципи ОО проектування та програмної інженерії, основні поняття мови *UML,* стратегію використання *UML*-діаграм при моделюванні програмних систем, основні архітектурні патерни, основи *.NET*- та *Java*-технологій, поняття сервісно-орієнтованої архітектури, парадигму декларативного програмування та її застосування у технологіях *.NET* та *Java* , поняття сервісно-орієнтованої архітектури, поняття фреймворку

вміти: використовувати об'єктно-орієнтований стиль програмування при проектуванні програмних систем, застосовувати моделювання програмних систем на основі *UML*-діаграм, розробляти розподілені системи з використанням різноманітних технологій (*.NET-Remoting*, *Java-RMI*), застосовувати сервісно-орієнтовану архітектуру, проектувати і розробляти *Web*-системи з використанням сучасних фреймворків, проектувати і розробляти мобільні проекти, застосовувати хмарні обчислення та технології.

Місце дисципліни ***(****в структурно-логічній схемі підготовки фахівців відповідного напряму)*. Навчальний курс "Інформаційні технології" є складовою частиною циклу професійної підготовки фахівців освітньо-кваліфікаційного рівня "магістр".

Зв'язок з іншими дисциплінами. Курс "Інформаційні технології" ґрунтується на нормативних курсах "Програмування", "Бази даних".

---

[3] Зазначається загальна кількість годин, які виділено на дану дисципліну згідно навчального плану відповідного освітньо-кваліфікаційного рівня.

[4] кредитів ECTS – кредит кратний 36 годинам (Наприклад, 3 кредити ECTS відповідає 108 год.).

Контроль знань і розподіл балів, які отримують студенти.

Контроль здійснюється за модульно-рейтинговою системою.

Контроль здійснюється за модульно-рейтинговою системою.

Обов'язковим для іспиту/заліку є виконання
індивідуальних завдань лабораторного практикуму.

(зазначаються умови, невиконання яких унеможливлює допуск до іспиту чи заліку)

Оцінювання за формами контролю[5]*:*

Семестрова оцінка – 60 балів; із них модульні оцінки – 60 балів.

Підсумкова екзаменаційна робота – 40 балів.

Максимальна підсумкова оцінка – 100 балів.

Порядок розрахунку загальної оцінки.

Модульні оцінки (один модуль) – 60.

Модульна оцінка є результатом виконання індивідуальних завдань лабораторного практикуму та тестової контрольної.

Студент допускається до складання іспиту, якщо кількість набраних за семестр балів не менше 25.

Для студентів, які набрали сумарно меншу кількість балів ніж *критично-розрахунковий мінімум – **25** балів* для одержання іспиту обов'язково слід довиконати всі завдання лабораторного практикуму.

У випадку відсутності студента з поважних причин відпрацювання та перездачі МКР здійснюються у відповідності до „Положення про порядок оцінювання знань студентів при кредитно-модульній системі організації навчального процесу" від 1 жовтня 2010 року.

При цьому, кількість балів:

- **1-34** відповідає оцінці «незадовільно» з обов'язковим повторним вивченням дисципліни;
- **35-59** відповідає оцінці «незадовільно» з можливістю повторного складання;
- **60-64** відповідає оцінці «задовільно» («достатньо»);
- **65-74** відповідає оцінці «задовільно»;
- **75 - 84** відповідає оцінці «добре»;
- **85 - 89** відповідає оцінці «добре» («дуже добре»);
- **90 - 100** відповідає оцінці «відмінно».

Шкала відповідності *(за умови іспиту*За 100 – бальною шкалою	За національною шкалою	
90 – 100	5	відмінно
85 – 89	4	добре
75 – 84		
65 – 74	3	задовільно
60 – 64		
35 – 59	2	не задовільно
1 – 34	2	не задовільно

[5] Див. Положення про порядок оцінювання знань студентів при кредитно-модульній системі організації навчального процесу від 1 жовтня 2010 року, а також Розпорядження ректора «Про методику розрахунку підсумкової оцінки дисциплін, які читаються два і більше семестри» від 29 вересня 2010 року

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Тема 1. Основи інженерії програмних систем. Уніфікована мова моделювання *UML* (огляд). Діаграми прецедентів.

Поняття програмної інженерії. Основні етапи життєвого циклу програмних систем. Варіанти (моделі) життєвого циклу. Ітеративно-інкрементні моделі життєвого циклу. Керування ризиками. Поняття програмної архітектури. Моделювання у програмній інженерії. Основні принципи моделювання програмних систем. Візуальне моделювання та *CASE*-технології. Призначення *UML*. Види діаграм *UML*. Діаграми структури та діаграми поведінки. Спрощена стратегія використання *UML*-діаграм при моделюванні ПС. Засоби розширення *UML: стереотипи (stereotype); помічені значення (tagged value); обмеження (constraint).* Профілі предметних областей. Кодогенерація (інженірінг) та реінженірінг. Діаграми прецедентів та їх використання. Моделювання контексту та вимог до програмної системи. Специфікація прецедентів. Прецеденти та потоки подій. Описи потоків подій. Потоки подій та сценарії як типи й екземпляри. Використання сценаріїв при плануванні версій. Зв'язки між акторами та прецедентами. Організація прецедентів (відношення залежності, включення та розширення). Відношення узагальнення для прецедентів та акторів. [1-8, 10].

Тема 2. Використання діаграм послідовностей та діаграм класів при проектуванні програмних систем

Реалізація прецедентів та використання діаграм послідовностей. Анатомія діаграм послідовності. Двохетапне розроблення діаграм послідовностей. Узгодженість (цілісність) моделей. Виявлення класів. Діаграми співробітництва. Класи етапу аналізу: прикордонні (*boundary*) або інтерфейсні класи; класи-сутності (*entity*); управляючі (*control*) класи (класи-менеджери). Класи етапу проектування. Відношення між класами (узагальнення, залежність, асоціація, агрегація, композиція) та їх виявлення. Проектування класів, відношень між класами. Пакетування класів. [3-8].

Тема 3. Сервісно-орієнтована архітектура. Веб-служби.

Веб-служби (*Web Services*) та сервісно-орієнтована архітектура. Стандарти веб-служб. Документування веб-служб: генерація документації для сприйняття людиною (з використанням веб-браузерів), генерація документації, орієнтованої на використання програмами – *wsdl*-файли. [16].

Тема 4. Технологія *Windows Communication Foundation***.**

Розробка клієнтських програм для веб-служб. Утиліта *Wsdl.exe.* Протокол *SOAP.* Конверт, заголовок *SOAP*-повідомлення. Стандарти *XML, XML-Schema.* Простори імен, монікери *XML.* Розробка веб-служб. Директива *@ WebService*. Тест-форми веб-служб. Утиліта *.NET WebService Studio* як універсальний клієнт. Структура *wsdl*-файлів. Оркестрування *Web*-сервісів. *BPEL* (*BPEL4WS – Business Process Execution Language for Web Services*). Технологія *Windows Communication Foundation*. Кінцеві точки (*Endpoints*) та "алфавітні трійки ***ABC***": ***A****ddress* (адреса); ***B****inding* (прив'язка); ***C****ontract* (контракт). [16, 22].

Тема 5. *Web*-технології *Java Servlet* та *JSP.*

Технологія *Java Servlet.* Методи життєвого циклу сервлетів, *java*-класи сервлетів. Варіанти реалізації сервлетів. Сервлети дії (*action servlet*). Основи мови *JSP.* Основні підходи до реалізації *Web*-проектів на основі *JSP. Model2* та патерн *Model-View-Controller*. Створення та використання "власних" дескрипторів *JSP.* [17-20].

Тема 6. *Web*-розробка із використанням *ASP Net.*

Особливості технологій *Active Server Pages* (*ASP*) *.NET* та *ASP.NET MVC.* Можливості візуального проектування в *ASP Net.* [22].

Тема 7. Хмарні обчислення та технології. Платформа *MS Azure.*

Microsoft Azure та реалізація двох хмарних моделей: платформи як сервіса (*Platform as a Service, PaaS*) та інфраструктури як сервіса (*Infrastructure as a Service, IaaS*). Приклад розробки *MS Azure* проекта [22].

Тема 8. Програмування під мобільні платформи. Приклад розробки під *OS Android.***.**

Засади мобільних систем та програмування під мобільні ОС. Архітектура *OS Android.* Приклад розробки проекта під *OS Android* [28].

ТЕМАТИЧНИЙ ПЛАН ЛЕКЦІЙ

№ лекції	Назва лекції	Кількість годин		
		Лекції	Лабор.	Сам. р-та
1	Основи інженерії програмних систем. Уніфікована мова моделювання *UML* (огляд). Діаграми прецедентів	2	1	4
2	Використання діаграм послідовностей та діаграм класів при проектуванні програмних систем	2	1	8
3	Сервісно-орієнтована архітектура. Веб-служби.	2	2	8
4	Технологія *Windows Communication Foundation*	2	2	8
5	*Web*-технології *Java Servlet* та *JSP*	2	1	8
6	*Web*-розробка із використанням *ASP Net*	2	1	8
7	Хмарні обчислення та технології. Платформа *MS Azure.*	2	2	8
8	Програмування під мобільні платформи. Приклад розробки під *OS Android*	2	2	8
	ВСЬОГО	16	12	60

Загальний обсяг годин – 90, у тому числі

Лекцій – 16 год.

Лабораторних – 12 год.

Консультації – 2 год.

Самостійна робота – 60 год.

Лекція 1**. Основи інженерії програмних систем. Уніфікована мова моделювання *UML* (огляд). Діаграми прецедентів. –** 2 год**.**

Поняття програмної інженерії. Основні етапи життєвого циклу програмних систем. Варіанти (моделі) життєвого циклу. Ітеративно-інкрементні моделі життєвого циклу. Керування ризиками. Поняття програмної архітектури. Моделювання у програмній інженерії. Основні принципи моделювання програмних систем. Візуальне моделювання та *CASE*-технології. Призначення *UML*. Види діаграм *UML*. Діаграми структури та діаграми поведінки. Спрощена стратегія використання *UML*-діаграм при моделюванні ПС. Засоби розширення *UML: стереотипи (stereotype); помічені значення (tagged value); обмеження (constraint).* Профілі предметних областей. Кодогенерація (інженірінг) та реінженірінг. Діаграми прецедентів та їх використання. Моделювання контексту та вимог до програмної системи. Специфікація прецедентів. Прецеденти та потоки подій. Описи потоків подій. Потоки подій та сценарії як типи й екземпляри. Використання сценаріїв при плануванні версій. Зв'язки між акторами та прецедентами. Організація прецедентів (відношення залежності, включення та розширення). Відношення узагальнення для прецедентів та акторів. [1-8, 10].

Лабораторний практикум - 1 год. Постановка задач лабораторного практикуму на семестр. Обговорення основних етапів лабораторного практикуму. [1-2, 10].

Завдання для самостійної роботи - 4 год. Обрання та підготовка інструментарію для лабораторного практикуму. Розробка ієрархії класів для представлення табличних даних, вибір поліморфних методів. Розробка діаграм прецедентів (моделювання контексту та вимог до програмної системи). Опис потоків подій для окремих прецедентів. [1-8].

Лекція 2. Використання діаграм послідовностей та діаграм класів при проектуванні програмних систем. – **2 год**.

Реалізація прецедентів та використання діаграм послідовностей. Анатомія діаграм послідовності. Двохетапне розроблення діаграм послідовностей. Узгодженість (цілісність) моделей. Виявлення класів. Діаграми співробітництва. Класи етапу аналізу: прикордонні (*boundary*) або інтерфейсні класи; класи-сутності (*entity*); управляючі (*control*) класи (класи-менеджери). Класи етапу проектування. Відношення між класами (узагальнення, залежність, асоціація, агрегація, композиція) та їх виявлення. Проектування класів, відношень між класами. Пакетування класів. [3-8].

Лабораторний практикум - 1 год. Розробка діаграм послідовностей. Виявлення класів. Проектування класів, відношень між класами. Розробка діаграм класів. Проектування класів, відношень між класами, розробка діаграм класів для проектів лабораторного практикуму. [3-8].

Завдання для самостійної роботи - 8 год. Розробка діаграм послідовностей, та виявлення класів для проектів лабораторного практикуму. Проектування класів, відношень між класами, розробка діаграм класів. Патерни проектування. Класифікація патернів: породжуючі патерни (пов'язані з процесом створення об'єктів); структурні патерни (ґрунтуються на композиціях – структурних об'єднаннях об'єктів чи класів); патерни поведінки (характеризуються взаємодією об'єктів між собою). Головна теза структурних патернів: замість успадкування – композиції. Простір патернів проектування**.** Приклади патернів: адаптер *Adapter*; міст *Bridge*; компоновник *Composite*; декоратор *Decorator*; фасад *Facade*; заступник *Proxy*; спостерігач *Observer.* Патерн (принцип) *IOC&DI - Inversion of Control* (*IoC*) *and Dependency Injection* (*DI*). *IoC Container* – ядро *Spring Framework.* [3-8, 9,20].

Лекція 3. Сервісно-орієнтована архітектура. Веб-служби. – **2 год.**

Веб-служби (*Web Services*) та сервісно-орієнтована архітектура. Стандарти веб-служб. Документування веб-служб: генерація документації для сприйняття людиною (з використанням веб-браузерів), генерація документації, орієнтованої на використання програмами – *wsdl*-файли. Розробка клієнтських програм для веб-служб. Утиліта *Wsdl.exe.* Протокол *SOAP.* Конверт, заголовок *SOAP*-повідомлення. Стандарти *XML, XML-Schema.* Простори імен, монікери *XML.* Розробка веб-служб. Директива *@ WebService*. Тест-форми веб-служб. Утиліта *.NET WebService Studio* як універсальний клієнт. Структура *wsdl*-файлів. Оркестрування *Web*-сервісів. *BPEL* (*BPEL4WS – Business Process Execution Language for Web Services*). [16].

Лабораторний практикум - 2 год. Розробка веб-служб. Розробка клієнтських програм для веб-служб. [16].

Завдання для самостійної роботи - 8 год. Розробка веб-служб для проекту лабораторного практикуму. Розробка клієнтської програми для веб-служби проекту лабораторного практикуму.

Лекція 4. Технологія *Windows Communication Foundation* – **2 год.**

Технологія *Windows Communication Foundation (WCF)*. Кінцеві точки (*Endpoints*) та "алфавітні трійки ***ABC***": ***A****ddress* (адреса); ***B****inding* (прив'язка); ***C****ontract* (контракт). [16, 22].

Лабораторний практикум - 2 год. Розробка *WCF*-служб. Розробка клієнтських програм для *WCF*-служб. [16, 22].

Завдання для самостійної роботи - 8 год. Розробка *WCF*-служб для проекту лабораторного практикуму. Розробка клієнтської програми для *WCF*-служби проекту лабораторного практикуму.

Лекція 5. *Web*-технології *Java Servlet* та *JSP.* – **2 год.**

Технологія *Java Servlet.* Методи життєвого циклу сервлетів, *java*-класи сервлетів. Варіанти реалізації сервлетів. Сервлети дії (*action servlet*). Основи мови *JSP.* Основні підходи до реалізації *Web*-проектів на основі *JSP. Model2* та патерн *Model-View-Controller*. [17-20].

Лабораторний практикум - 1 год. Основні підходи до реалізації *Web*-проектів на основі *JSP*. Розробка сторінок *JSP.* [23-26].

Завдання для самостійної роботи - 8 год. Розробка сторінок *JSP Web*-проектів індивідуального завдання лабораторного практикуму. Патерн *Model-View-Controller* у *Web*-проектах. *Web MVC*- проектування. Технологія *action servlet.* Поняття фреймворку. *Action-oriented Frameworks* (фреймворки *Struts*, *WebWork* та *Spring*). Валідація даних у *Web*-проектах: програмна та декларативна (із використанням конфігураційних файлів чи анотацій, орієнтована на поля чи не на поля, здійснюється на боці сервера чи на боці клієнта). Підтримка *AJAX* у *Web*-фреймворках. Використання бібліотек *JavaScript.* [23-26].

Лекція 6. *Web*-розробка із використанням *ASP Net.* – **2 год.**

Особливості технологій *Active Server Pages* (*ASP*) *.NET* та *ASP.NET MVC.* Можливості візуального проектування в *ASP Net.* [22].

Лабораторний практикум - 1 год. *Web*-проекти [22].

Завдання для самостійної роботи - 8 год. Розробка *Web*-проектів індивідуального завдання лабораторного практикуму [22].

Лекція 7. Хмарні обчислення та технології. Платформа *MS Azure.* – **2 год.**

Парадигма хмарних обчислень. Програмне забезпечення як Інтернет-сервіс. Моделі хмарних обчислень. *Microsoft Azure* та реалізація двох хмарних моделей: платформи як сервіса (*Platform as a Service, PaaS*) та інфраструктури як сервіса (*Infrastructure as a Service, IaaS*). Приклад розробки *MS Azure* проекта [22].

Лабораторний практикум - 2 год. Проект *Microsoft Azure*. [25].

Завдання для самостійної роботи - 8 год. Розробка веб-проекту індивідуального завдання лабораторного практикуму під *Microsoft Azure* [25].

Лекція 8. Програмування під мобільні платформи. Приклад розробки під *OS Android.* – **2 год.**

Засади мобільних систем та програмування під мобільні ОС. Архітектура *OS Android.* Приклад розробки проекта під *OS Android* [28].

Лабораторний практикум - 2 год. Приклад розробки проекта під *OS Android* [28].

Завдання для самостійної роботи - 8 год. Розробка мобільного проекту індивідуального завдання лабораторного практикуму [28].

Індивідуальні завдання лабораторного практикуму

Загальна постановка задачі та варіанти лабораторного практикуму

Задача:

"Фрагментарна реалізація систем управління табличними базами даних"

І. Загальні вимоги

Основні вимоги щодо структури бази:

* кількість таблиць принципово не обмежена (реляції між таблицями не враховувати);
* кількість полів та кількість записів у кожній таблиці також принципово не обмежені.

У кожній роботі треба забезпечити підтримку (для полів у таблицях) наступних (загальних для всіх варіантів!) типів:

* integer;
* real;
* char.

Також у кожній роботі треба реалізувати функціональну підтримку для:

* створення бази;
* створення та знищення таблиці з бази;
* перегляду та редагування рядків таблиці;
* збереження табличної бази на диску та, навпаки, зчитування її з диску.

ІІ. Варіанти додаткових типів

Потрібно забезпечити підтримку (для можливого використання у таблицях) деяких додаткових типів у відповідності з одним із наступних варіантів:

0) string[n], n <= 255; stringInvl;
1) текстові файли; integerInvl;
2) html-файли; longint;
3) charInvl; string(charInvl);
4) перелічуваний тип (множину значень складає набір рядків);
5) date; dateInvl;
6) time; timeInvl;
7) $ (тип для грошових розрахунків, max$ =10,000,000,000,000.00); $Invl;
8) complexInteger; complexReal;
9) picture-файли (один з форматів); realInvl;

Примітка. Типи integerInvl, charInvl, dateInvl тощо є "інтервальними" типами.

III. Варіанти додаткових операцій над таблицями

Потрібно реалізувати операції над таблицями у відповідності з варіантом:

0) сортування таблиці за одним ключем;
1) пошук (за шаблоном) та перегляд знайдених рядків таблиці;
2) об'єднання таблиць;
3) різниця таблиць;
4) перетин таблиць;
5) вилучення повторюваних рядків у таблиці;
6) прямий добуток двох таблиць;
7) проекція таблиць;
8) сполучення таблиць (за спільним полем);
9) перейменування колонок таблиці (з відповідною зміною схеми таблиці).

IV. Варіанти індивідуальних завдань

Загалом, окремий варіант індивідуального завдання визначається парою:

- варіант додаткових типів даних (розділ II);
- варіант додаткових операцій над таблицями бази (розділ III).

V. Етапи (завдання) лабораторного практикуму

1) Підготовчий етап. Функціональна специфікація системи управління табличними базами даних (СУТБД) у вигляді діаграм варіантів використання (діаграм прецедентів) UML.

Наступні основні етапи можна виконувати у довільному порядку.

2) Використання UML при проектуванні та специфікації програмних систем.

Розробити щонайменше 12 UML-діаграм:

діаграм варіантів використання (діаграм прецедентів) - 1..* (щонайменше до трьох прецедентів описати потоки подій);

діаграм класів - 2..*. Щонайменше одна з них має містити класи "Таблиця" та "База", подаючи відношення як між цими класами, так і з класами, що використовуються для опису схем, рядків таблиці тощо. Щонайменше одна з діаграм має бути VOPC-діаграмою для “індивідуального” прецедента (у відповідності до варіанта додаткової операції - див. розділ III). (VOPC - це абревіатура від View Of Participating Classes);

діаграм діяльності - 0..*;.

діаграм взаємодії - 4..*. Щонайменше одна з діаграм має бути побудованою для “індивідуального” прецедента (у відповідності до варіанта додаткової операції).

діаграм станів - 0..*;

діаграм компонентів - 2..* (щонайменше по одній - для нерозподіленої та розподіленої версій системи відповідно);

діаграм розгортання - 1..* (щонайменше - для розподіленої системи).

3) Розроблення (власних!) класів для понять "Таблиця", "База" та, можливо, дяких інших класів, спряжених із поняттям "Таблиця" (наприклад, "Схема таблиці", "Атрибут", "Рядок таблиці" тощо): створити UML-діаграму для таких класів та провести unit-тестування (надати 3..* тести, один з яких призначений для тестуванням варіантної операції з розділу III).

4) Розробка локальної (нерозподіленої) версії СУТБД із власною реалізацією класів "Таблиця" та "База".

5-6) Два варіанти розподілених версій системи (із реалізацією програм-клієнтів та програм-серверів), використовуючи за власним вибором будь-які дві з наступних технологій: Java RMI/JRMP, Java RMI/IIOP, Net Remoting, WCF, IIOP Net, EJB тощо.

7) Варіант розподіленої версії з використанням COM або CORBA (сервер, клієнт).

8) Рефлексія (reflection). Реалізація "динамічних викликів" на прикладі одного з об'єктів клієнтської частини розподіленої версії.

9) Web-сервіси. Реалізація СУТБД на основі технології web-сервісів (сервер, клієнт).

10) Web-сервіси. Для web-сервісу (серверної частини з попереднього етапу) розробити варіант клієнтської програми на "альтернативній платформі". Наприклад, для реалізованого на Java web-сервісу клієнтську частину можна розробити під .NET чи, навпаки, для реалізованого ASMX web-сервісу.NET клієнтську частину можна розробити на Java). Клієнтський проект може бути функціонально обмеженим (*).

11) Сумісність технологій. Серед можливих варіантів пропонуються такі: ASMX web-сервіси - WCF, Java RMI/IIOP - CORBA). Треба, наприклад, для ASMX web-сервісу, реалізованого на сьомому етапі, розробити WCF-клієнт або, навпаки, для WCF-сервісу - ASMX-клієнт.

12) Web-проект. (Технології на вибір: ASP .NET, ASP .NET MVC, WPF, JSP, JavaServlet та інші, у тому числі на основі фреймворків Spring, Struts, Struts 2, JSF, Tapestry, Wicket, GWT тощо. Проект може бути функціонально обмеженим (*).

13) Web-проект із використанням AJAX. Проект може бути функціонально обмеженим (*).

14) Мобільний проект (Android, iOS, Windows Phone тощо). Проект може бути функціонально обмеженим(*).

15) Варіант проекту із використанням хмарних технологій (Microsoft Azure, Google App Engine, Amazon Web Services тощо). Проект може бути функціонально обмеженим (*).

16 Extra) Ще одна версія Web-проекту (додатково до пункту 12) з використанням іншої технології (іншого фреймворку). Проект може бути функціонально обмеженим (*).

(*) - як мінімум має забезпечуватись демонстрація виконання варіантної операції (див. розділ III) над таблицями даних, що також відповідають варіанту (див. розділ II).

Рекомендовані джерела

Презентації лектора до курсу (*www.magistrs2016.github.io*)

Основна література

1. Соммервилл И. Инженерия программного обеспечения – М.: Вильямс, 2002.
2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на *C++*. Второе издание М.: Бином, СПб.: Невский диалект, 2000.
3. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения – СПб.: Питер, 2002.
4. Коберн А. Современные методы описания требований к системам.– М.: Лори, 2002.
5. Буч. Г. , Рамбо Дж. , Джекобсон А. Язык *UML*. Руководство пользователя –М.: ДМК, 2000.
6. Рамбо Дж., Якобсон А., Буч Г. *UML*: Специальный справочник – СПб.: Питер, 2002.
7. Фаулер М., Скотт К. *UML* в кратком изложении – М.: Мир, 1999.
8. Кватрани Т. *Rational Rose* 2000 и *UML*. Визуальное моделирование. М.: ДМК, 2001.
9. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования – СПб.: Питер-ДМК, 2001.
10. Эммерих В. Конструирование распределенных объектов – М.: Мир, 2002.
11. Троелсен Э. C# и платформа *.NET*.— СПб.: Питер,2004.—796 с.
12. Платт Д.С. Знакомство с *Microsoft .NET*— М.: Издательско-торговый дом «Русская Редакция »,2001.— 240с.

13. Просиз Дж. Программирование для *Microsoft .NET* — М.: Издательско-торговый дом "Русская Редакция",2003.— 704 стр.
14. Маклин С.,Нафтел Дж,,Уильямс К. *Microsoft .NET Remoting* — М.: Издательско-торговый дом «Русская Редакция »,2003.— 384 с.
15. Рихтер Дж. Программирование на платформе *Microsoft .NET Framework* — М.: Издательско-торговый дом «Русская Редакция »,2003 — 512 стр.
16. *Microsoft Corporation* Основы *Microsoft Visual Studio .NET* 2003 *М.*: Издательско-торговый дом «Русская Редакция », 2003.— 464 стр.
17. Шилдт Г. , Холмс Дж. Искусство программирования на *Java*.
18. Хорстманн К., Корнелл Г. *.Java* 2. Библиотека профессионала. Том 2.Тонкости программирования. — М.:Издательский дом "Вильямс", 2002.— 1120 с.
19. Ахмед Х.З., Амриш К.Е. Разработка корпоративных *Java*-приложений с помощью *J2EE* и *UML* . К.: Вильямс, 2002.
20. Хемраджани А. Гибкая разработка приложений на *Java* с помощью *Spring*, *Hibernate* и *Eclipse*. — М.:Издательский дом "Вильямс", 2008.— 352 с.

Додаткова

21. http://java.sun.com/javaee/5/javatech.html
22. http://msdn.microsoft.com
23. http://java.sun.com/j2ee/index.jsp
24. http://java.sun.com/docs/index.html
25. https://developers.google.com/appengine/
26. http://www.hibernate.org/
27. http://www.springframework.org/
28. http://developer.android.com/index.html

ПИТАННЯ НА ІСПИТ

1. Поняття програмної інженерії. Моделювання у програмній інженерії.
2. Життєвий цикл програмних систем (ПС). Моделі життєвого циклу ПС. Ітеративно-інкрементні моделі життєвого циклу. Керування ризиками.
3. Візуальне моделювання. Моделювання та CASE-технології. Уніфікована мова моделювання UML. Призначення UML у розрізі проектування ПС. Види діаграм UML. Спрощена стратегія використання UML-діаграм при моделюванні ПС.
4. Засоби розширення UML: стереотипи (stereotype), помічені значення (tagged value), обмеження (constraint). Профілі предметних областей.
5. Діаграми прецедентів. Моделювання контексту та вимог до ПС. Прецеденти. Специфікація прецедентів у Rational Rose. Потоки подій та сценарії.
6. Актори, основні актори. Відношення між акторами та прецедентами. Відношення узагальнення для прецедентів та акторів.
7. Організація прецедентів. Відношення залежності між прецедентами. Відношення включення (include) та розширення (extend). Варіанти діаграм прецедентів.
8. Реалізація прецедентів. Використання діаграм послідовностей. Анатомія діаграм послідовності. Двохетапне розроблення діаграм послідовностей. Узгодженість (цілісність) моделей. Діаграми класів-учасників VOPC (View of Participating Classes) прецедентів.
9. Використання класів при проектуванні ПС. Класи етапу аналізу: прикордонні (boundary) або інтерфейсні класи, класи-сутності (entity), управляючі (control) класи (класи-менеджери). Класи етапу проектування. Діаграми співробітництва (collaboration) та їх використання.
10. Відношення між класами та їх виявлення (узагальнення, залежність, асоціація, агрегація, композиція). Проектування класів, відношень між класами. Проектування атрибутів та операцій. Пакетування класів.
11. Діаграми класів та патерни проектування. Структура патернів. Класифікація патернів: породжуючі, структурні, поведінкові.
12. Приклади патернів: «Singleton», «Adapter», «Proxy», «Decorator», «Composite», «Bridge», «Observer».
13. Патерн IOC&DI. Ілюстративний приклад.
14. Патерн IOC&DI. Spring: IoC + декларативний стиль.
15. Використання діаграм класів для кодогенерації. Кодогенерація та реінженіринг.
16. Платформа .NET. Основні складові частини CLI. Середовище виконання .NET. Загальна система типів CTS.
17. Поняття керованого коду. .NET-компіляція. Міжмовна інтеграція у .NET.
18. Метадані. Самоопис керованого коду .NET. Механізм рефлексії.
19. Мова CIL. Just In Time (JIT) компіляція.

20. Платформа .NET. Збірки .NET.
21. Платформа .NET. Управління пам'яттю. Збирання сміття.
22. Типи об'єктів для віддаленої взаємодії. Домени. Типи marshal-by-value. Серіалізація. Типи marshal-by-reference.
23. Серверна активізація. Режими Singleton та SingleCall. Клієнтська активізація.
24. Канали. Стандартні типи каналів: TcpChannel та HttpChannel. Конфігурування інфраструктури .NET Remoting: програмне; з використанням конфігураційних файлів.
25. Основи управління часом життя об'єктів .NET.
26. Веб-служби (Web Services) та сервісно-орієнтована архітектура (СОА). Стандарти веб-служб.
27. Документування веб-служб: генерація документації для сприйняття людиною (з використанням веб-браузерів), генерація документації, орієнтованої на використання програмами – wsdl-файли.
28. Розробка веб-служб на платформі .NET. Директива @ WebService. Тест-форми веб-служб. Утиліта .NET WebService Studio як універсальний клієнт.
29. Розробка веб-служб на платформі Java.
30. Розробка клієнтських програм для веб-служб на платформі .NET. Утиліта Wsdl.exe.
31. Розробка клієнтських програм для веб-служб на платформі Java.
32. Протокол SOAP. Конверт, заголовок SOAP-повідомлення.
33. Стандарти XML, XML-Schema. Простори імен, монікери XML.
34. Структура wsdl-файлів. Оркестровка Web-сервісів. BPEL (BPEL4WS – Business Process Execution Language for Web Services). Візуальне проектування BPEL-програм.
35. Основи архітектури WCF. Кінцеві точки.
36. Прив’язки WCF. Стандартні прив’язки.
37. Метадані WCF-служб.
38. WCF. Behavior. Режими інстанціації (instance).
39. WCF. Підтримка асинхронних викликів. Використання транзакцій.
40. WCF. DataContract.
41. Сумісність служб ASMX та WCF.
42. Спрощена архітектура RMI. Особливості програмування RMI/JRMP-проектів.
43. “Віддалені” інтерфейси та класи реалізації “віддалених” інтерфейсів. Порівняння RMI/JRMP та RMI/IIOP. «Експортування» об'єктів. Статичний метод ExportObject.
44. RMI/IIOP-проекти. Використання rmic та orbd.
45. JNDI та конкретні служби іменування. Особливості налаштування JNDI.
46. (JRMP – IIOP) портабельність Java RMI-проектів.
47. Сервлети. Сценарій використання сервлетів.
48. Web-технологія JSP. Основні підходи до реалізації Web-проектів на основі JSP. Патерн Model-View-Controller у web-проектуванні.
49. Основи Action-oriented Frameworks: диспетчерський сервлет, mapping; action-класи.
50. Використання технології action-класів у web-проектуванні. Приклад.
51. Struts, WebWork, Spring як приклади Action-орієнтованих фреймворків.
52. ASP.NET-проекти. Життєвий цикл сторінок.
53. Структура ASP.NET-проектів та їх розробка. Використання технології відокремлення коду. Засоби візуального проектування, управляючі елементи (контроли). Валідація даних.
54. Проблеми postback-запитів та можливості збереження стану сторінок. Сховані поля ASP.NET-проектів. Використання ViewState та ControlState. Можливості використання Cookie, Session- та Application-контейнерів.
55. Знайомство з AJAX на модельному проекті електронного магазину.
56. Засоби ASP.NET AJAX Extensions.
57. Приклад Java-проекту з AJAX на основі JSP та сервлетів.
58. Фреймворк Google Web Toolkit (GWT).
59. AJAX-проекти із використанням GWT.
60. Розробка Android-програм. Android Developer Tools, Android SDK Manager, Virtual Device Manager. Архітектура Android OS. Розробка клієнтів веб-служб на платформі Android.
61. Можливості візуального проектування програм під Android OS. Активності (activities) та розмітки (layouts). Віджети. Програмне оперування віджетами. Приклад.
62. Android OS. Управління активностями. Передача даних в іншу активність. Динамічне наповнення активності віджетами. Приклад.

Завдання для самостійної роботи з елементами дистанційного навчання
з дисципліни «Інформаційні системи та технології»
на період з 24 січня до 28 лютого 2018 р.

для студентів

1 курсу
другого рівня (магістр)
освітньої програми «Інформаційні системи та технології»
викладач (лекції та лабораторні заняття): к.ф.-м.н., доц. Кузенко В.Ф. (електронна пошта kuzenko@knu.ua)

Види та форми контрольних заходів з перевірки самостійної роботи студентів, критерії оцінювання

Контроль за виконанням самостійної роботи студентами здійснюється у двох формах:

- у січні-лютому за допомогою електронних засобів (електронною поштою);
- у березні – шляхом проведення письмової контрольної роботи (у формі тесту) та провелення захисту проектів лабораторного практикуму.

Додаткові матеріали для самостійної роботи студентів та умови лабораторного практикуму розміщені за посиланням: ***https://magistrs2016.github.io***.

Контроль у січні-лютому 2018 р. відбувається у два етапи.

Під час **першого етапу** (24 січня – 7 лютого 2018 р.) студенти мають вивчити запропоновані теми на базовому рівні.

Для підтвердження виконання завдання студенти повинні надіслати не пізніше **7 лютого 2018 р**. звіт з виконання попереднього етапу лабораторного практикуму. Викладач оцінює виконане завдання в категоріях «**зараховано**» або «**не зараховано**».

Якщо студент отримає оцінку **«не зараховано»**, у нього є час до **10 лютого** переробити завдання та надіслати їх викладачу повторно.

На **другому етапі** самостійної роботи (7 лютого – 28 лютого 2018 р.) кожен студент має опанувати питання винесені на самостійну роботу на поглибленому рівні та реалізувати щонайменше ще один (вже один з основних) етап лабораторного практикуму.

Теми та питання для самостійного опрацювання

Тема 2. Основи інженерії програмних систем. Уніфікована мова моделювання *UML* (огляд). Діаграми прецедентів.

Поняття програмної інженерії. Основні етапи життєвого циклу програмних систем. Варіанти (моделі) життєвого циклу. Ітеративно-інкрементні моделі життєвого циклу. Керування ризиками. Поняття програмної архітектури. Моделювання у програмній інженерії. Основні принципи моделювання програмних систем. Візуальне моделювання та *CASE*-технології. Призначення *UML*. Види діаграм *UML*. Діаграми структури та діаграми поведінки. Спрощена стратегія використання *UML*-діаграм при моделюванні ПС. Засоби розширення *UML: стереотипи (stereotype); помічені значення (tagged value); обмеження (constraint).* Профілі предметних областей. Кодогенерація (інженірінг) та реінженірінг. Діаграми прецедентів та їх використання. Моделювання контексту та вимог до програмної системи. Специфікація прецедентів. Прецеденти та потоки подій. Описи потоків подій. Потоки подій та сценарії як типи й екземпляри. Використання сценаріїв при плануванні версій. Зв'язки між акторами та прецедентами. Організація прецедентів (відношення залежності, включення та розширення). Відношення узагальнення для прецедентів та акторів. [1-8, 10].

Тема 2. Використання діаграм послідовностей та діаграм класів при проектуванні програмних систем

Реалізація прецедентів та використання діаграм послідовностей. Анатомія діаграм послідовності. Двохетапне розроблення діаграм послідовностей. Узгодженість (цілісність) моделей. Виявлення класів. Діаграми співробітництва. Класи етапу аналізу: прикордонні (*boundary*) або інтерфейсні класи; класи-сутності (*entity*);

управляючі (*control*) класи (класи-менеджери). Класи етапу проектування. Відношення між класами (узагальнення, залежність, асоціація, агрегація, композиція) та їх виявлення. Проектування класів, відношень між класами. Пакетування класів. [3-8].

Тема 3. Сервісно-орієнтована архітектура. Веб-служби.

Веб-служби (*Web Services*) та сервісно-орієнтована архітектура. Стандарти веб-служб. Документування веб-служб: генерація документації для сприйняття людиною (з використанням веб-браузерів), генерація документації, орієнтованої на використання програмами – *wsdl*-файли. [16].

Завдання лабораторного практикуму

Загальне формулювання задачі:"Фрагментарна реалізація систем управління табличними базами даних"

I. Загальні вимоги

- Основні вимоги щодо структури бази:
  - кількість таблиць принципово не обмежена (реляції між таблицями не враховувати);
  - кількість полів та кількість записів у кожній таблиці також принципово не обмежені.
- У кожній роботі треба забезпечити підтримку (для полів у таблицях) наступних (загальних для всіх варіантів!) типів:
  - integer;
  - real;
  - char.

- Також у кожній роботі треба реалізувати функціональну підтримку для:
  - створення бази;
  - створення та знищення таблиці з бази;
  - перегляду та редагування рядків таблиці;
  - збереження табличної бази на диску та, навпаки, зчитування її з диску.

II. Варіант індивідуального завдання (від 00 до 99) визначається двома останніми цифрами номера особистого студентського квитка. При тому передостання цифра задає варіант додаткових типів даних (**див. розділ III**), а остання — варіант додаткових операцій над таблицями бази (**див. розділ IV**).

III. Варіанти додаткових типів

Потрібно забезпечити підтримку (для можливого використання у таблицях) деяких додаткових типів у відповідності з одним із наступних варіантів:

0) string[n], n <= 255; stringInvl;
1) текстові файли; integerInvl;
2) html-файли; longint;
3) charInvl; string(charInvl);
4) перелічуваний тип (множину значень складає набір рядків);
5) date; dateInvl;
6) time; timeInvl;
7) $ (тип для грошових розрахунків, max$ =10,000,000,000,000.00); $Invl;
8) complexInteger; complexReal;
9) picture-файли (один з форматів); realInvl;

Примітка. Типи integerInvl, charInvl, dateInvl тощо є "інтервальними" типами.

IV. Варіанти додаткових операцій над таблицями

Потрібно реалізувати операції над таблицями у відповідності з варіантом:

0) сортування таблиці за одним ключем;
1) пошук (за шаблоном) та перегляд знайдених рядків таблиці;
2) об'єднання таблиць;
3) різниця таблиць;
4) перетин таблиць;
5) вилучення повторюваних рядків у таблиці;
6) прямий добуток двох таблиць;
7) проекція таблиць;
8) сполучення таблиць (за спільним полем);
9) перейменування колонок таблиці (з відповідною зміною схеми таблиці).

V. Етапи (завдання) лабораторного практикуму

1) Підготовчий етап. Функціональна специфікація системи управління табличними базами даних (СУТБД) у вигляді діаграм варіантів використання (діаграм прецедентів) UML.

Наступні основні етапи можна виконувати у довільному порядку.

2) Використання UML при проектуванні та специфікації програмних систем.

Розробити щонайменше 12 UML-діаграм:

діаграм варіантів використання (діаграм прецедентів) - 1..* (щонайменше до трьох прецедентів описати потоки подій);
діаграм класів - 2..*. Щонайменше одна з них має містити класи "Таблиця" та "База", подаючи відношення як між цими класами, так і з класами, що використовуються для опису схем, рядків таблиці тощо. Щонайменше одна з діаграм має бути VOPC-діаграмою для “індивідуального” прецедента (у відповідності до варіанта додаткової операції - див. розділ III). (VOPC - це абревіатура від View Of Participating Classes);
діаграм діяльності - 0..*;.
діаграм взаємодії - 4..*. Щонайменше одна з діаграм має бути побудованою для “індивідуального” прецедента (у відповідності до варіанта додаткової операції).
діаграм станів - 0..*;
діаграм компонентів - 2..* (щонайменше по одній - для нерозподіленої та розподіленої версій системи відповідно);
діаграм розгортання - 1..* (щонайменше - для розподіленої системи).
3) Розроблення (власних!) класів для понять "Таблиця", "База" та, можливо, дяких інших класів, спряжених із поняттям "Таблиця" (наприклад, "Схема таблиці", "Атрибут", "Рядок таблиці" тощо): створити UML-діаграму для таких класів та провести unit-тестування (надати 3..* тести, один з яких призначений для тестуванням варіантної операції з розділу III).

4) Розробка локальної (нерозподіленої) версії СУТБД із власною реалізацією класів "Таблиця" та "База".

5-6) Два варіанти розподілених версій системи (із реалізацією програм-клієнтів та програм-серверів), використовуючи за власним вибором будь-які дві з наступних технологій: Java RMI/JRMP, Java RMI/IIOP, Net Remoting, WCF, IIOP Net, EJB тощо.

7) Варіант розподіленої версії з використанням COM або CORBA (сервер, клієнт).

8) Рефлексія (reflection). Реалізація "динамічних викликів" на прикладі одного з об'єктів клієнтської частини розподіленої версії.

9) Web-сервіси. Реалізація СУТБД на основі технології web-сервісів (сервер, клієнт).

10) Web-сервіси. Для web-сервісу (серверної частини з попереднього етапу) розробити варіант клієнтської програми на "альтернативній платформі". Наприклад, для реалізованого на Java web-сервісу клієнтську частину можна розробити під .NET чи, навпаки, для реалізованого ASMX web-сервісу.NET клієнтську частину можна розробити на Java). Клієнтський проект може бути функціонально обмеженим (*).

11) Сумісність технологій. Серед можливих варіантів пропонуються такі: ASMX web-сервіси - WCF, Java RMI/IIOP - CORBA). Треба, наприклад, для ASMX web-сервісу, реалізованого на сьомому етапі, розробити WCF-клієнт або, навпаки, для WCF-сервісу - ASMX-клієнт.

12) Web-проект. (Технології на вибір: ASP .NET, ASP .NET MVC, WPF, JSP, JavaServlet та інші, у тому числі на основі фреймворків Spring, Struts, Struts 2, JSF, Tapestry, Wicket, GWT тощо. Проект може бути функціонально обмеженим (*).

13) Web-проект із використанням AJAX. Проект може бути функціонально обмеженим (*).

14) Мобільний проект (Android, iOS, Windows Phone тощо). Проект може бути функціонально обмеженим(*).

15) Варіант проекту із використанням хмарних технологій (Microsoft Azure, Google App Engine, Amazon Web Services тощо). Проект може бути функціонально обмеженим (*).

16 Extra) Ще одна версія Web-проекту (додатково до пункту 12) з використанням іншої технології (іншого фреймворку). Проект може бути функціонально обмеженим (*).

(*) - як мінімум має забезпечуватись демонстрація виконання варіантної операції (див. розділ III) над таблицями даних, що також відповідають варіанту (див. розділ II).