Министерство образования и науки Российской Федерации

Российский университет дружбы народов

Инженерный факультет

Кафедра Кибернетики и мехатроники

Программа дисциплины

"Объектно-ориентированное программирование"

для направления 550200 (220200) “Автоматизация и управление”
подготовки бакалавра

Автор программы
профессор, д.т.н. Е.И.Забудский
e-mail: zei@inbox.ru , web-site: http://zei.narod.ru

Одобрена на заседании кафедры
Кибернетики и мехатроники

Зав. кафедрой
профессор, д.т.н._____ К.А. Пупков

“___”_____ 2010 г.

Москва

I. Пояснительная записка

Автор программы:

профессор, доктор техн. наук Е.И. Забудский

Общие сведения об учебном курсе:

дисциплина читается студентам Инженерного факультета Российского университета дружбы народов. Она входит в блок общепрофессиональных дисциплин и читается во втором, третьем, четвертом и пятом модулях второго учебного года. Количество кредитов – 6. Продолжительность курса составляет 72 аудиторных учебных часа (18 недель), в том числе: 36 часов лекционных занятий, 36 часов практических занятий и 117 часов самостоятельной работы. Формы контроля знаний студентов – две контрольные работы и домашнее задание. По окончании пятого модуля принимается экзамен.

Требования к студентам:

освоение курса предполагает предварительное знакомство студентов с содержанием учебной дисциплины: “Информатика и программирование.

Цель курса:

научить студентов разрабатывать в соответствии с парадигмой компонентно-ориентированного программирования компьютерные модели реальных и концептуальных систем соответствующих направлению «Автоматизация и управление».

Аннотация.

В процессе изучения дисциплины рассматриваются следующие вопросы**:** объектно-ориентированный анализ (OOA), объектно-ориентированное проектирование (OOD), объектно-ориентированное программирование (ООП), шаблоны проектирования, унифицированный язык моделирования UML (**U**nified **M**odeling **L**anguage), а также **спец**вопросы объектно-ориентированного языка программирования C# (C_Sharp), касающиеся парадигмы ООП, etc.

В основе всех этих вопросов лежит один и тот же фундамент**:** *способность и необходимость мыслить категориями объектов реального мира*, так как специалисту-программисту необходимо разрабатывать Windows-приложения, эмулирующие те или иные *системы реального мира*. Поэтому изучение концепции объектного подхода не заканчивается изучением отдельно взятого метода или набора средств разработки. Иными словами, объектный подход является образом *объектно-ориентированного мышления*, которому также обучаются студенты.

Переходить на новый способ мышления всегда непросто, поэтому вербальный метод обучения сопровождается активным привлечением компьютерных и информационных технологий. Это позволяет сопровождать рассуждения о концепциях объектов демонстрацией и анализом соответствующих фрагментов программного кода, а также иллюстративной графики.

Особое внимание уделяется организации самостоятельной работы студентов и ее методическому обеспечению.

С этой целью автором программы разработан Учебно-методический комплекс дисциплины «Объектно-ориентированный анализ и программирование», который помещен на сайт http://zei.narod.ru. Комплекс включает: методические рекомендации преподавателям и студентам, программу дисциплины и нормативно-справочные материалы, материалы к лекциям, материалы к практическим занятиям, темы и рекомендации по выполнению домашнего задания и др. Все материалы комплекса доступны студентам и используются ими в процессе обучения.

Учебные задачи курса.

В результате освоения учебного курса студенты должны:

- ✓ иметь представление о различных парадигмах разработки программных продуктов в историческом контексте;
- ✓ получить углубленные знания по методологии объектно-ориентированного

программирования;

- ✓ уметь разрабатывать компьютерные модели реальных и концептуальных систем на основе парадигмы компонентно-ориентированного программирования.

II. Тематический план учебной дисциплины

№ темы	Название темы	Всего часов по дисциплине	Аудиторные часы		Самостоятельная работа
			Лекции	Практич. занятия	
Второй модуль (16 часов)					
1	Введение. Методология разработки объектно-ориентированного программного обеспечения	8	2	–	6
2	Объектно-ориентированный анализ и проектирование: основные понятия и терминология. Цели (результаты) анализа и проектирования. Сопоставление ОО языков программирования C# и Java	10	2	2	6
3	Инкапсуляция – центральное понятие в парадигме объектно-ориентированного программирования	18	2	3	13
4	Наследование – базовое понятие объектно-ориентированного программирования	18	2	3	13
Третий модуль (14 часов)					
5	Полиморфизм – базовое понятие объектно-ориентированного программирования	20	3	4	13
6	Основы UML – унифицированного языка моделирования объектно-ориентированных систем	20	4	3	13
Четвертый модуль (14 часов)					
7	Основы объектно-ориентированного анализа	20	3	4	13
8	Основы объектно-ориентированного проектирования	20	4	3	13
Пятый модуль (28 часов)					
9	Объектно-ориентированный подход к созданию пользовательского интерфейса, использование шаблонов проектирования	21	4	4	13
10	Разработка компьютерных моделей реальных и концептуальных систем на основе методологии компонентно-ориентированного программирования	34	10	10	14
Итого		**189**	**36**	**36**	**117**

III. Литература

Базовый учебник

1. Мейер Б. Объектно-ориентированное конструирование программных систем. М.: Русская Редакция, 2005.

Основная

2. Буч Г., Якобсон А., Рамбо Дж. **UML**. С.-Петербург: Питер, 2006.

3. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования. С.-Петербург: Питер, 2006.

4. Забудский Е.И. Учебно-методический комплекс дисциплины «Объектно-ориентированный анализ и программирование». М.: Кафедра АПС ГУ- ВШЭ; кафедра Кибернетики и мехатроники, 2005 - 2013

Internet-ресурс – http://zei.narod.ru .

5. Кватрани Т. Визуальное моделирование с помощью Rational Rose 2002 и UML. М.: Вильямс, 2003.
6. Лафоре Р. Объектно-ориентированное программирование в C++. С.-Петербург: Питер, 2005.
7. Троелсен Э. C# и платформа .NET. С.-Петербург: Питер, 2006.
8. Синтес А. Освой самостоятельно объектно-ориентированное программирование за 21 день. Москва; С.-Петербург; Киев: Вильямс, 2002.

Дополнительная – Internet-ресурсы

9. Новые книги раздела **C#** – http://books.dore.ru/bs/f6sid16.html .
10. **C#** и ***.NET*** по шагам – http://www.firststeps.ru .
11. ***UML*** – язык графического моделирования – http://www.uml.org/ .
12. ***NUnit, JUnit*** – каркасы тестирования для испытания классов – http://www.junit.org , http://www.nunit.org .
13. Пакет объектного моделирования ***Rational Rose*** – http://www-306.ibm.com/software/rational/ .
13*a*. Steve Burbeck "Applications Programming in Smalltalk-80(TM): How to use **M**odel-**V**iew-**C**ontroller (**MVC**)" – http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html .
13*b*. Информация о языке C# и платформе .NET – http://msdn2.microsoft.com/ru-ru/default.aspx , http://www.gotdotnet.com , http://www.gotdotnet.ru .

Дополнительная – книги

14. Мэтт Вайсфельд. Объектно-ориентированный подход: Java, .NET, C++. М.: КУДИЦ-ОБРАЗ, 2005.
15. Дж. Кьоу, М. Джеанини. Объектно-ориентированное программирование. С.-Петербург: Питер, 2005.
16. Уоткинз Д., Хаммонд М, Эйбрамз Б. Программирование на платформе .NET.: М.: Вильямс, 2003.

IV. Формы контроля

- ✓ промежуточный контроль: зачет в конце 3 и 4-го модулей;
- ✓ итоговый контроль: экзамен в конце 5-го модуля;

Оценки промежуточного и итогового контроля складывается из следующих элементов:

работа на практических занятиях

текущий контроль осуществляется посредством ведения учета посещаемости аудиторных занятий (лекций и практических) и оценки качества подготовки и результатов работы на практических (работа на практических включает решение задач и ответы на вопросы по существу темы занятия, etc.); при непосещении занятий ставится оценка "**0**";

контрольная работа

разработка программного продукта по итогам изучения дисциплины в течение модуля и компьютерная реализация; при пропуске контрольной работы ставится оценка "**0**";

домашнее задание

разработка компьютерной модели реальной (концептуальной) системы в соответствии с методологией компонентно-ориентированного программирования. В конце 3 и 4-го модулей осуществляется контроль хода *выполнения* домашнего задания. *Выполненное* и оформленное домашнее задание сдается на проверку в конце 5-го модуля, до экзамена.

- ✓ оценка промежуточного контроля ***З*** по 10-балльной шкале формируется как взвешенная сумма:

$$\boldsymbol{З} = 0.3\ Пз + 0.3\ Дз + 0.4\ Кр\ ,$$

где *Пз, Дз, Кр* – 10-балльные оценки соответственно за работу на практических занятиях, за работу над *выполнением* домашнего задания и за контрольную работу с округлением до целого числа баллов. При 10-балльной оценке не менее 4 баллов проставляется **зачет**, иначе – ***не***зачет.

- ✓ итоговая оценка ***К*** по 10-балльной шкале формируется как взвешенная сумма:

$$\boldsymbol{К} = 0.3\ Дз + 0.3\ \boldsymbol{З} + 0.4\ Э\ ,$$

где *Дз,* ***З*** и *Э* – 10-балльные оценки соответственно за *выполненное* домашнее задание, за зачет (учитывается также качество работы на практических занятиях в пятом модуле) и за экзамен с округлением до целого числа баллов. Перевод в пятибалльную оценку осуществляется в соответствии со следующей таблицей.

Таблица соответствия оценок по десяти- и пятибалльной системам

Десятибалльная шкала	Пятибалльная шкала
неудовлетворительно – 1 очень плохо – 2 плохо – 3	2 – неудовлетворительно
удовлетворительно – 4 весьма удовлетворительно – 5	3 – удовлетворительно
хорошо – 6 очень хорошо – 7	4 – хорошо
почти отлично – 8 отлично – 9 блестяще –1 0	5 – отлично

V. Содержание программы

Тема 1. *Введение. Методология разработки объектно-ориентированного программного обеспечения*

Объектно-ориентированное мышление. Принципы объектно-ориентированного подхода. Объектно-ориентированное программирование в историческом контексте. *Шесть этапов* объектно-ориентированной методологии: *Этап предпроектных исследований* – Учет технических, временных и финансовых ограничений. Определение целесообразности продолжения работы над приложением; *Этап анализа* – Сбор информации, необходимой для продолжения работы; *Этап проектирования* – Создание схемы интерфейса и структуры программы, без создания каких-либо фактических программ; *Этап разработки* (*реализации*) – Создание приложения, включая все элементы интерфейса и программного кода; *Этап внедрения* – Применение и тестирование программы; *Этап опытной эксплуатации* – Совершенствование продукта, призванное устранить возможные проблемы или удовлетворить новые запросы.
Демонстрация реализации этапов объектно-ориентированной методологии на примере разработки проекта «Расчет оценки студента»: *C_Sharp*-программа в вариантах ***GUI*** (***G****raphical* ***U****ser* ***I****nterface*) и консольном.

Литература [1, гл. 19] [4, Пр_Зан. 8, стр. 4сл.] [6, гл. 1, стр. 32…42]

Контрольные вопросы
1. Что такое процедурное программирование?
2. Какими преимуществами обладает процедурное программирование по сравнению с неструктурным программированием?
3. Что такое модульное программирование?
4. Какими преимуществами обладает модульное программирование по сравнению с процедурным программированием?
5. Перечислите недостатки процедурного и модульного программирования.
6. Что такое объектно-ориентированное программирование?
7. Каковы преимущества и цели объектно-ориентированного программирования?
8. Объясните одну из целей объектно-ориентированного программирования.
9. Дайте определения следующих терминов: класс, объект, поведение.
10. Как объекты обмениваются информацией?
11. Что такое конструктор?

Тема 2. *Объектно-ориентированный анализ и проектирование: основные понятия и терминология. Цели анализа и проектирования. Сопоставление ОО языков программирования C# и Java*

Основные понятия, терминология и цель (результат) ОО анализа. Основные понятия, терминология и цель (результат) ОО проектирования. Сопоставление синтаксиса и семантики ОО языков программирования C# и Java.

Литература [4, см. web-ссылки на стр. 2] [8, стр. 558-580] [14, 15]

Тема 3. *Инкапсуляция – центральное понятие объектно-ориентированного программирования*

Инкапсуляция – объектно-ориентированная характеристика модульности. Внешний интерфейс и внутренняя реализация инкапсулированного программного объекта. Характерные признаки эффективной инкапсуляции: абстракция, общедоступный интерфейс и сокрытие реализации.

Демонстрация и анализ концепций инкапсуляции в *C_Sharp*-программе «Объектно-ориентированный Банк» и др.

Литература [1, гл. 3] [4, Пр_Зан. 3, стр. 4сл.] [7, гл. 3, стр. 149…157] [8, стр. 44…88]

Контрольные вопросы

1. Каким образом использование инкапсуляции помогает достичь целей объектно-ориентированного программирования?
2. Дайте определение понятию «абстракция» и приведите пример применения абстракции.
3. Дайте определение понятию «реализация».
4. Дайте определение понятию «интерфейс».
5. Объясните разницу между интерфейсом и реализацией.
6. Почему для достижения эффективной инкапсуляции важно четко распределить ответственность?
7. Определите понятие типа.
8. Что такое абстрактный тип данных?
9. Как можно получить эффективное сокрытие реализации в сильносвязанной программе?
10. Какие опасности таит абстракция?

Тема 4. *Наследование – базовое понятие объектно-ориентированного программирования*

Наследование – механизм, дающий возможность создавать новый класс на основе уже существующего класса. Базовый и производный классы. Абстрактные метод и класс. Наследование реализации, поведения и свойства. Переопределение метода. Типы наследования. Множественное наследование: проблемы (*the diamond problem*) и решения (*interface* – особый абстрактный класс). Сравнение отношений «*Is-a*» («Является») и «*Has-a*» («Содержит»)**:** когда использовать наследование?
Демонстрация и анализ концепций наследования в *C_Sharp*-программе «Банковский счет» и др.

Литература [1, гл. 4, 14, 15, 16] [4, Пр_Зан. 6, стр. 4сл.] [6, гл. 9, стр. 362…380]
[7, гл. 3, стр. 158…163] [8, стр. 92…118]

Контрольные вопросы

1. Что такое наследование?
2. Что такое простое наследование?
3. Что такое многоуровневое наследование?
4. Что такое множественное наследование?
5. Что такое тесты «*Is-a*» и «*Has-a*»?
6. Когда используется множественное наследование?
7. Когда используется многоуровневое наследование?
8. Каково максимальное количество уровней в многоуровневом наследовании?
9. Какие члены класса может наследовать другой класс?
10. Какова разница между базовым классом и производным классом?
11. Как наследование разрушает инкапсуляцию?

Тема 5. *Полиморфизм – базовое понятие в парадигме объектно-ориентированного программирования*

Полиморфизм – самое радикальное, универсальное средство – *одно* имя класса или метода представляет *различный*, выбранный *автоматическим* механизмом, *программный код* (полиморфизм – одно имя представляет различные поведения). Связь полиморфизма с инкапсуляцией и наследованием. Формы полиморфизма: полиморфизм включения, параметрический полиморфизм, переопределение метода, перегрузка метода. Ранее связывание (во время компиляции) и позднее связывание (во время выполнения).
Демонстрация и анализ концепций полиморфизма в модифицированной *C_Sharp*-программе «Банковский счет», в *C_Sharp*-программе «Служащие с разной формой оплаты**:** начисление зарплаты» и др.

Литература [1, гл. 14] [4, Пр_Зан. 7, стр. 4сл.] [7, гл. 3, стр. 168…174] [8, стр. 139…162]

Контрольные вопросы

1. Что такое полиморфизм?

2. Как реализуется полиморфизм?
3. Что такое позднее связывание?
4. Что такое раннее связывание?
5. Каковы преимущества полиморфизма времени выполнения?
6. Каковы преимущества полиморфизма времени компиляции?
7. Что такое интерфейс?
8. Как полиморфизм позволяет реализовать интерфейсы?
9. Что такое виртуальная функция?
10. Что такое перегрузка метода?
11. Как инкапсуляция и наследование влияют на полиморфизм включения?

Тема 6. *Основы UML – унифицированного языка моделирования объектно-ориентированных систем*

U**nified **M**odeling **L**anguage* (UML***) – язык графического моделирования, используемый для представления объектно-ориентированных программ. Краткая история *UML*. Псевдокод – представление алгоритма в пределах метода. Система обозначений языка *UML* для описания отношений классов и общей архитектуры программы. Моделирование отношений между классами: зависимость, ассоциация, агрегация, композиция, обобщение. Интерактивный пакет *Rational Rose* – использование языка *UML на стадии проработки проекта.* Среда *MS VS* ***.****NET* **2005** – генерация *UML*-диаграмм классов *после разработки проекта и написания программного кода*.
Демонстрация и анализ *UML*-диаграмм классов проектов «Банковский счет», «Служащие с разной формой оплаты**:** начисление зарплаты» и «Расчет оценки студента».

Литература [2, гл. 1, стр. 22…34; гл. 4, стр. 65…84]
[5, гл. 1, стр. 19…26; гл. 2, стр. 29…32; гл 6, стр. 83…91]
[4, Пр_Зан. 3, стр. 26сл.; Пр_Зан. 8, стр. 38сл.] [8, стр. 197…212] [10] [11] [13]

Контрольные вопросы
1. Что такое *UML*?
2. Что в *UML* обозначают следующие символы: +, #, -?
3. Каким образом выделяются абстрактные классы на *UML*-диаграмме?
4. Что является конечной целью моделирования? Какие из этих целей являются более важными?
5. Объясните суть понятий «ассоциация», «агрегация» и «композиция».
6. Объясните, когда следует использовать каждое из отношений: «ассоциация», «агрегация» и «композиция».
7. Что такое *UML*-диаграмма взаимодействия?
8. Что такое сообщение с технической точки зрения?
9. Что такое псевдокод, с какой целью используется?
10. Есть ли специальные программные средства для использования *UML*?

Тема 7. *Основы объектно-ориентированного анализа*

Итеративная технология разработки программного обеспечения. Объектно-ориентированный анализ (ООА) –это объектно-ориентированный подход к осмыслению разрабатываемого проекта. Результат ООА – понимание предметной области, формулировка технических требований к системе в терминах классов и взаимодействий между объектами (система – множество взаимодействующих объектов). Модель прецедентов – модель способов взаимодействия пользователей с системой. Концептуальная модель (модель предметной области) – скелет создаваемой системы. Паралич анализа (*analysis paralysis*).
Демонстрация реализации этапа объектно-ориентированного анализа на примере разработки проекта «Расчет оценки студента»: *C_Sharp*-программа в вариантах ***GUI*** (***G**raphical **U**ser **I**nterface*) и консольном.

Литература [1, гл. 3, 19, 27] [4, Пр_Зан. 8, стр. 4сл.] [6, гл. 16, стр. 752…762] [8, стр. 213…235]

Контрольные вопросы
1. Что такое технология разработки программного обеспечения?
2. Что такое итеративный процесс (итеративная технология)?
3. Что является результатом проведения объектно-ориентированного анализа?
4. Что описывают требования к системе?

5. Что такое прецедент?
6. Какие действия необходимо выполнить для определения прецедентов?
7. Что такое действующий субъект?
8. Какие вопросы помогают найти действующих субъектов?
9. Какие отношения могут существовать между прецедентами?
10. Что такое вариант прецедента?
11. Что такое сценарий?
12. Какие существуют способы описания прецедентов?
13. Опишите разницу между различными моделями, используемыми для визуализации прецедентов.
14. Чем полезна концептуальная модель?
15. Чем полезно использование прецедентов?

Тема 8. *Основы объектно-ориентированного проектирования*

Объектно-ориентированное проектирование (OOD) – итеративный процесс, определяющий назначение объектов и взаимодействие между ними. Пять шагов OOD: *шаг 1* – создание исходного списка объектов; *шаг 2* – определение и уточнение назначений объектов при помощи карточек ***CRC*** (карточки связи и взаимодействия между классами – ***C****lass* ***R****esponsibility* ***C****ollaboration cards*); *шаг 3* – разработка точек взаимодействия; *шаг 4* – детализация отношений между объектами; *шаг 5* – построение объектной модели. Объектная модель – описание объектной структуры и отношения между объектами. Паралич проектирования (*design paralysis*).

Литература [4, Пр_Зан. 8, стр. 6сл.] [5, гл. 11, стр. 129…139] [6, гл. 16, стр. 752…762] [8, стр. 236…252]

Контрольные вопросы
1. Что такое ООПр?
2. Что такое объектная модель?
3. Каковы недостатки излишнего проектирования?
4. Как определить, какие аспекты системы являются архитектурно важными*?*
5. Каковы пять основных шагов ООПр?
6. Как создать начальный список объектов?
7. Что входит в полный проект?
8. Что помогают определить карточки *CRC*?
9. Что такое сотрудничество?
10. Почему важно глубоко понять назначения и связи объектов?
11. Что такое карточка *CRC*?
12. Назовите хотя бы одну причину, по которой карточки *CRC* сделаны настолько простыми.
13. Когда следует использовать карточки *CRC*?
14. Какова основная проблема при использовании карточек *CRC*?
15. Что такое точка взаимодействия?

Тема 9. *Объектно-ориентированный подход к созданию пользовательского интерфейса*

Формы пользовательского интерфейса (*User Interface*). Значение развязки пользовательских интерфейсов: не «навешивать» интерфейс на систему после ее создания. Развязка пользовательского интерфейса с помощью шаблона проектирования Модель/Вид/Контроллер (***M****odel* ***V****iew* ***C****ontroller* – ***MVC***) – основная система полностью независима от интерфейса.
Демонстрация и анализ пользовательского интерфейса проекта «Расчет оценки студента»: *C_Sharp*-программа в вариантах ***GUI*** (***G****raphical* ***U****ser* ***I****nterface*) и консольном.

Литература [3, гл. 1, стр. 18…20] [4, Пр_Зан. 8, стр. 11сл.] [13*a*] [7, гл. 8, стр. 343…375; гл. 9, стр. 409…416; гл. 10, стр. 471…490] [8, стр. 302…323]

Контрольные вопросы
1. В чем отличие анализа, проектирования и реализации пользовательского интерфейса от соответствующих шагов разработки других частей программы?
2. Почему нужно отделять пользовательский интерфейс от основной программы?
3. Назовите составные *MVC*-триады.
4. Назовите две возможные альтернативы шаблону *MVC*.

5. Опишите обязанности модели.
6. Опишите обязанности вида.
7. Опишите обязанности контроллера.
8. Сколько моделей может быть в системе? Сколько видов может быть в системе? Сколько контроллеров может быть в системе?
9. Что может быть причиной снижения эффективности при использовании шаблона *MVC*?
10. Какие допущения предполагает шаблон *MVC*?

Тема 10. *Разработка компьтерных моделей реальных и концептуальных систем на основе методологии компонентно-ориентированного программирования*

Объектно-ориентированный анализ: выявление прецедентов – способов взаимодействия пользователей с системой; определение сценариев – последовательности событий для каждого прецедента; построение диаграммы прецедентов – диаграмма последовательности событий, диаграмма сотрудничества; построение концептуальной модели и словаря предметной области. *Объектно-ориентированное проектирование*: создание исходного списка объектов (шаг 1-й); определение назначения объектов (использование карточек *CRC*) (шаг 2-й); определение точек взаимодействия – мест обращения объектов друг к другу (шаг 3-й); детализация отношений между объектами (шаг 4-й); построение объектной модели – диаграммы классов и взаимодействий между ними (шаг 5-й). *Разработка кода системы и интерфейса.* Реализация *итеративной технологии* на всех этапах разработки.

Литература [4, см. web-ссылки на стр. 6] [6, гл. 16, стр. 752…791] [8, стр. 357…458]

VI. Тематика заданий по различным формам контроля

Аудиторные письменные работы

1. Абстракция – учимся думать и программировать абстрактно.
2. Эффективное применение инкапсуляции в объектно-ориентированном программировании.
3. Открытый интерфейс и эффективное сокрытие реализации.
4. Использование концепции наследования при объектно-ориентированном программировании.
5. Применение простого наследования.
6. Использование абстрактных классов при наследовании.
7. Адаптация объектно-ориентированных программ к изменяющимся требованиям средствами полиморфизма.
8. Применение полиморфизма включения.
9. Переопределение – важный тип полиморфизма.
10. Перегрузка – частный случай полиморфизма.
11. Необходимое условие эффективного полиморфизма – эффективное применение инкапсуляции и наследования.
12. Построение *UML*-диаграмм классов программных продуктов, разрабатываемых в домашних и аудиторных работах.
13. Анализ прецедентов – случаев взаимодействия пользователя с системой – при выполнении объектно-ориентированного анализа с целью уяснения смысла задач, разрабатываемых в домашних и аудиторных работах.
14. Построение концептуальной модели – выявление объектов предметной области, необходимых для адекватного описания системы – при выполнении объектно-ориентированного анализа с целью уяснения смысла задач, разрабатываемых в домашних и аудиторных работах.
15. Построение объектной модели – установление взаимосвязей и структуры объектов – при выполнении объектно-ориентированного проектирования с целью приближения к искомой конструкции систем, разрабатываемых в домашних и аудиторных работах.
16. Практическое применение шаблонов при выполнении объектно-ориентированного программирования.
17. Программирование пользовательского интерфейса на основе объектно-ориентированного подхода в задачах, разрабатываемых в домашних и аудиторных работах.

Контрольная работа

Содержанием контрольных работ является решение задач (разработка программных продуктов) на компьютере по тематике пройденного материала.

Темы домашнего задания

1. Компьютерное моделирование работы банка
2. Компьютерное моделирование работы и облуживания банкомата
3. Компьютерное моделирование работы четырех светофоров на перекрестке
4. Компьютерное моделирование системы лифтов здания
5. Компьютерное моделирование работы склада
6. Компьютерное моделирование работы отдела кадров
7. Компьютерное моделирование работы Дома Мод
8. Компьютерное моделирование Салона Красоты
9. Компьютерное моделирование игровой ситуации – компьютерная игра
10. Компьютерное моделирование концептуальной системы тестирования знаний
11. Компьютерное моделирование системы определения цены на новый товар
12. Компьютерное моделирование электронного магазина, специализирующегося на комиссионной торговле
13. Компьютерное моделирование электронного магазина с организацией лотерей
14. Компьютерное моделирование электронного магазина розничной торговли по оптовым ценам
15. Компьютерное моделирование концептуальной системы анализа поведения посетителей супермаркета
16. Компьютерное моделирование системы оплаты заказа с использованием терминала автоматизированной системы
17. Компьютерное моделирование поведения потребителя на рынке
18. Компьютерное моделирование системы формирования предложений для клиента торговой фирмы
19. Авторские темы, предлагаемые студентами в соответствии с содержанием дисциплины

Экзаменационные вопросы для оценки качества освоения дисциплины **ООР, C#**			
# пп	**Экзаменационный вопрос**	**# лекц.**	**# пр.зан.**
1	Концепция и технологии **.NET**		**0. Введ в .NET**
2	Парадигма объектно-ориентированного программирования и ее предшественники	**1**	**1**
3	Терминология объектно-ориентированного программирования: класс, объект, переменные экземпляра, метод, интерфейс, реализация, поведение, **etc**.	**1**	**1, 2**
4	Три базовых понятия парадигмы объектно-ориентированного программирования	**2, 3, 4, 5**	
5	Инкапсуляция: абстракция, интерфейс и реализация	**2**	**3**
6	Инкапсуляция: средства защиты и доступа	**2**	**2, 3, *6***
7	Наследование: отношения **"Is_A"** и **"Has-A".** Наследование для многократного использования реализации и наследование для отличия.	**3**	**6, 7**
8	Типы наследования: простое наследование.	**5**	**6**
9	Типы наследования: многоуровневое наследование.	**5**	**6**
10	Типы наследования: множественное наследование и «**проблема бриллианта**»	**5**	**6, 7**
11	Интерфейсы в **C#** - аналог множественного наследования	**5**	**7**
12	Стандартные интерфейсы в объектно-ориентированном языке программирования **C#**	**5**	**7**
13	Абстрактные классы и методы	**5**	**7**
14	Формы полиморфизма: полиморфизм включения	**4**	**7**
15	Формы полиморфизма: полиморфизм посредством переопределение методов	**4**	**6**
16	Формы полиморфизма: полиморфизм посредством перегрузки методов	**4**	**6, 7**

17	Раннее и позднее (динамическое) связывание. Полиморфизм времени выполнения	**4**	**7**
18	Парадигма компонентно-ориентированного программирования: компоненты и клиенты		**1, 4**
19	Основные стандартные классы библиотеки **System .Windows. Forms** и простанство имен **System.Drawing**	**7,11**	**8**
20	Стандартный класс **System.Delegate** и использование делегатов и событий	**7,11**	**8**
21	Реализация обработчика событий в Сопрограммах, управляемых событиями	**11**	**8**
22	Оконное **Windows-приложение** с основными элементами управления на форме: создание приложения в **Visual Studio .NET** и компиляция в интегрированной среде разработки	**7,11**	**8**
23	Оконное **Windows-приложение** с основными элементами управления на форме: разработка **C#**-программы в редакторе, компиляция в командной строке и компиляция в интегрированной среде разработки	**7,11**	**6, 8**
24	Анатомия классов и их разработка в парадигме объектно-ориентированного программирования	**12**	**1, 5**
25	Основы языка моделирования (**UML**) для графического представления объектно-ориентированного программного обеспечения	**12**	**3**
26	Стадии разработки объектно-ориентированных компьютерных моделей реальных и концептуальных систем	**12**	**8**
27	**Основы объектно-ориентированного анализа**: прецеденты и сценарии	**12**	**8**
28	Основы объектно-ориентированного анализа: диаграммы прецедентов, диаграммы взаимодействия, диаграммы активности	**12**	**8**
29	Основы объектно-ориентированного анализа: концептуальная модель - скелет разрабатываемой системы	**12**	**8**
30	**Основы объектно-ориентированного проектирования**: использование карточек **CRC (Class Responsibility Collaboration)** для определения назначения и связи объекта	**14**	**8**
31	Основы объектно-ориентированного проектирования: объектная модель разрабатываемой системы и ее значение для написания кода	**14**	**3, 5**
32	Объектно-ориентированный подход к программированию пользовательского интерфейса	**15**	**8**

* На экзамене (120 мин) студент в письменной форме дает ответ на вопрос и решает задачу на компьютере

Методические рекомендации и материалы преподавателю

Практические занятия по дисциплине проводятся в компьютерном классе в следующих средах:

- ✓ Microsoft Visual Studio .NET **2005**,
- ✓ Rational Rose [13],
- ✓ World Wide Web.

Рекомендуются для использования следующие Internet-ресурсы, см. [4, 9…13, 13*a*, 13*b*].

Особенность дисциплины заключается в том, что для успешного усвоения и реализации парадигмы объектно-ориентированного программирования необходимо знание совокупности вопросов, которые по своему объему и сложности должны были бы составить предмет самостоятельных курсов. Имеются в виду: среда MS VS .NET; спецвопросы языка C#, касающиеся парадигмы ООП; язык *UML*; объектно-ориентированный анализ; объектно-ориентированное проектирование. Весьма непростым для студентов является анализ предметной области, рассматриваемой с целью ее компьютерного моделирования. Сложность обучения дисциплине также заключается в крайнем дефиците выделенного аудиторного времени. Поэтому *особое внимание необходимо уделить организации **самостоятельной** работы студентов и ее методическому обеспечению*.

Для решения вопроса автором программы разработан Учебно-методический комплекс дисциплины**,** который помещен на сайт автора программы URL-адрес:

http://zei.narod.ru ,
при запросе пароля - кликнуть на кнопке «отмена».

Далее приведен 6-й раздел комплекса, а также краткие методические рекомендации по проведению практических занятий.

6. Компьютерные модели реальных и концептуальных систем, разработанные в соответствии с парадигмой ООП

Наименование моделируемой предметной области	Объем файла, КБ
Система Домовладелец (**LandLord**)	758
Система Высотные Лифты Здания (**Elevator**)	373
Компьютерная Игра (**Blackjack**) /см. предварительно Дополн. 1/	677
Дополнение 1 к проекту Компьютерная игра (**Blackjack**)	602
Система Ипподромные Состязания (**Derby**)	291
Система **Р**асчет **О**ценок **С**тудента (см. Практическое занятие **8**)	1151
….	

В Материалах к практическим занятиям (4-й раздел комплекса) приведены**:** тема и примерное количество аудиторных часов, отводимых на ее изучение; содержание; теория и листинги программ; резюме; контрольные вопросы и задания (упражнения) по объектно-ориентированному программированию; список литературы; ссылки на Internet-ресурсы и др.

Существенным и необходимым для усвоения дисциплины является самостоятельная работа студентов во *вне*аудиторное время с содержательной частью разработанных Материалов …, составление ответов на контрольные вопросы и разработка программ согласно упражнениям, которые приведены в заключительной части (следует рекомендовать студентам скопировать и, по возможности, распечатать “Материалы …”). На практических занятиях необходимо проводить блиц-опрос студентов в соответствии с контрольными вопросами и осуществлять проверку выполнения упражнений.

Далее, на практических занятиях необходимо кратко разъяснить студентам основные положения, раскрывающие тему; затем предложить студентам **откорректировать** и реализовать в среде Visual Studio **.**NET соответствующие программные продукты, представленные в Материалах …, и выполнить их анализ. Часть программ следует поручить студентам реализовать и проанализировать во *вне*аудиторное время.

Предусмотрено выполнение студентами домашнего задания. Его содержанием является компьютерное моделирование реальных или концептуальных систем средствами среды Rational Rose, среды Visual Studio **.**NET и языка C_Sharp в соответствии с парадигмой *компонентно-ориентированного* программирования. Задание (см. темы на с. 10) целесообразно определить студентам в конце второго модуля (или в начале третьего) с указанием разделов, которые необходимо выполнить и представить для контроля преподавателю в конце третьего и в конце четвертого модулей. Полностью выполненное и оформленное домашнее задание сдается студентами преподавателю в конце пятого модуля, перед экзаменом. С целью оказания помощи студентам в выполнении домашнего задания рекомендуется на практических занятиях осуществить компьютерное моделирование двух типических систем (см., например, 4. Материалы к практическим занятиям, # # 3 и 5).

В конце пятого модуля принимается экзамен (продолжительность 120 мин) по дисциплине. На экзамене студент в письменной форме дает ответ на вопрос (см. с. 10) и решает задачу на компьютере.

Указания студенту

1. Обязательно посещать лекции ведущего преподавателя; лекции – основное методическое руководство при изучении дисциплины, наиболее оптимальным образом структурированное и скорректированное на современный материал; в лекции глубоко и подробно, аргументировано и методологически строго рассмотрены главные проблемы темы; в лекциях даны необходимые разные подходы к исследуемым проблемам.

2. Готовиться и активно работать на практических занятиях; подготовка к практическим занятиям включает самостоятельную проработку материалов лекций и практических занятий, рекомендованной учебной литературы (см. Учебно-методический комплекс дисциплины «Объектно-ориентированный анализ и программирование» **[4]**).

3. Обратить особое внимание на систематическое и последовательное выполнение Домашнего задания, которое является концентрированным выражением качества усвоения дисциплины “Объектно-ориентированный анализ и программирование”. Содержанием Домашнего задания (см. темы на с. 10) является разработка в соответствии с парадигмой компонентно-ориентированного программирования компьютерной модели реальной (концептуальной) системы. Далее приводится обязательное содержание (оглавление) оформленного Домашнего задания, которое представляется преподавателю на проверку в конце пятого модуля, до экзамена.

Содержание

* См. Учебно-методический комплекс дисциплины:
Internet-ресурс – http://zei.narod.ru

.

Автор программы:  / Е.И. Забудский /