Структура междисциплинарного проекта по модулю
«Основы объектно-ориентированного программирования» (3-й семестр)

Введение

Целью выполнения проекта является закрепление теоретических знаний и приобретение практических навыков использования методов объектно-ориентированного анализа и программирования. Проект по модулю включает 4 раздела. Варианты заданий приводятся в приложении и в файле «**Class_List(T)_Variants.pdf**». Разделы 3 и 4 должны завершаться выводом по результатам выполнения.

1. **Обзор понятий, средств и методов объектно-ориентированного анализа и программирования**

Теоретический обзор темы по выбранному варианту. Примерное содержание: актуальность и назначение рассматриваемой темы или технологии, краткое описание, решаемые задачи, понятия, основные принципы, преимущества и недостатки, примеры реализации на языке C#.

2. **Построение блок-схемы выполнения операций в рассматриваемой предметной области**

- Необходимо выбрать и дать краткое описание исследуемой сферы деятельности, предметной области или определенного круга задач.
- Необходимо разработать, построить блок-схему выполнения операций или последовательности действий в рассматриваемой предметной области, дать соответствующие пояснения.

3. **Объектно-ориентированный анализ и моделирование с помощью UML**

В ходе выполнения для выбранной (или заданной) предметной области изучается процесс построения диаграмм UML с помощью средства, поддерживающего язык UML.

Общая последовательность выполнения включает определенные этапы.

1. Постановка задачи. Выбор темы для моделирования (предметной области), цель моделирования, объекты и выполняемые функции.
2. Построение диаграммы вариантов использования (диаграммы прецедентов, Use Case Diagram).
3. Построение диаграммы деятельности (Activity Diagram).
4. Построение диаграммы классов (Class Diagram).
5. Построение диаграмм взаимодействия (диаграмма последовательности действий Sequence Diagram и диаграмма кооперации Communication/Collaboration Diagram).

4. Разработка приложения

Работа с классами (создание модуля с классами). Структура проекта приложения должна включать два модуля:

- первый модуль главной формы для создания интерфейса приложения;
- второй модуль классов (без формы), содержащий определения базовых и производных классов для реализации задания.

Создание и использование коллекции List<T>. В среде Microsoft Visual Studio (C#) необходимо создать приложение (Windows Forms), в котором с помощью класса List<T> реализуются основные операции с коллекцией (или со списком), построенным из элементов различного типа, в том числе сортировка списка по разным критериям.

Класс List<T>, который в языке C# называется коллекцией, способен хранить объекты произвольного типа (или любого класса). Класс List<T> объявлен в пространстве имен System.Collections.Generic. В отличие от фиксированных статических массивов, в коллекциях List<T> размер увеличивается по мере необходимости, то есть классы List<T> поддерживают динамические массивы, расширяющиеся и сокращающиеся по мере необходимости.

Варианты заданий описаны в файле «**Class_List(T)_Variants.pdf**».

Общая последовательность выполнения включает определенные этапы. При завершении очередного этапа рекомендуется сохранять проект, выполнять компиляцию и отладку.

Средствами Microsoft Visual Studio построить диаграмму классов, дать краткое описание и поместить ее в отчет.

Заключение

В ходе выполнения работы над проектом необходимо подготовить и оформить отчет, содержащий требуемые разделы, последовательность разработки приложения, комментарии операторов, результаты выполнения, необходимые пояснения, краткий вывод в заключении.

Необходимо также предоставить проект разработанного приложения.

При сохранении файла отчета и архива проекта приложения следует использовать фамилию студента и номер варианта проекта.

Приложение

Проект по модулю. Варианты заданий

Обзор понятий, средств и методов объектно-ориентированного анализа и программирования

1. Развитие, сущность и особенности объектно-ориентированного подхода. Преимущества и недостатки объектно-ориентированного подхода.
2. Обзор и сравнительная характеристика языков объектно-ориентированного программирования.
3. Развитие методов и методологий анализа, моделирования и проектирования программных систем. Возможности CASE-средств.
4. Базовые принципы объектно-ориентированного подхода.
5. Обзор и сравнительная характеристика CASE-средств для объектно-ориентированного анализа и моделирования.
6. Объектно-ориентированное программирование. Понятие и основные принципы. Особенности и назначение абстрагирования и инкапсуляции.
7. Основные принципы объектно-ориентированного программирования. Особенности наследования. Конструкторы базовых и производных классов. Абстрактные классы и методы.
8. Основные принципы объектно-ориентированного программирования. Понятие и особенности полиморфизма. Статическое и динамическое связывание. Преимущества и недостатки полиморфизма
9. Классы и объекты. Характеристики компонентов класса. Конструктор класса, поля, свойства, методы класса. Создание и использование объектов.
10. Компоненты платформы Microsoft .NET Framework. Компилирование и выполнение .NET-приложения. Преимущества и недостатки платформы.
11. Структура и особенности программ на языке C#. Реализация основных принципов объектно-ориентированного программирования. Создание и использование классов и объектов на языке C#.
12. Разновидности массивов в языке C#. Многомерные массивы. Объявление, инициализация массивов.
13. Функции и процедуры. Объявление функции. Вызов функции. Формальные и фактические параметры. Способы передачи параметров.
14. Общая характеристика и особенности языка C#. Структура программы на языке C#. Классификация типов данных языка C#. Общая система типов CTS. Преобразования типов.
15. Динамические массивы и коллекции в языке C#.
16. Назначение, развитие и характеристика языка UML. Типы и назначение диаграмм UML.
17. Унифицированный процесс RUP. Характеристика, особенности применения. Структура жизненного цикла RUP. Назначение, содержание фаз и процессов.
18. Язык UML. Диаграммы вариантов использования. Назначение, элементы. Построение диаграммы с помощью CASE-средства. Правила построения диаграмм.
19. Язык UML. Диаграммы взаимодействия. Назначение, элементы. Построение диаграмм взаимодействий с помощью CASE-средства.
20. Язык UML. Диаграммы классов и компонентов. Назначение, элементы. Построение диаграмм и моделирование структуры приложения с помощью CASE-средства.