С.Сейфуллин атындағы Қазақ агротехникалық университетінің 60 жылдығына арналған «**Сейфуллин оқулары– 13: дәстүрлерді сақтай отырып, болашақты құру**» атты Республикалық ғылыми-теориялық конференциясының **материалдары** = **Материалы** Республиканской научно-теоретической конференции **«Сейфуллинские чтения – 13: сохраняя традиции, создавая будущее»**, посвященная 60-летию Казахского агротехнического университета имени С.Сейфуллина. - **2017. - Т.I, Ч.5.** - С.239-242

ИСПОЛЬЗОВАНИЕ ПАРАДИГМ ОБЪЕКТНО - ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ В ТЕХНОЛОГИИ СОЗДАНИЯ ИНТЕРНЕТ МАГАЗИНА ЭЛЕКТРОННЫХ УЧЕБНЫХ ИЗДАНИЙ

Турсумбаева А.

Объектно - ориентированное программирование(ООП) представляет собой парадигму программирования, которая представляет основные понятия, такие как «объекты», которые имеютполя данных(атрибуты, описывающиеобъекты) и связанные с ним процедуры, известных какметоды.Объекты, которые обычно являютсяслучаиизклассов, которые используются для взаимодействия друг с другом и последующей разработки приложений и компьютерных программ [1].

Объектно - ориентированные программы можно рассматривать как совокупность взаимодействующихобъектов, в отличие от традиционной модели, в которой программа рассматривается как список задач (подпрограмм) для выполнения.В ООП, каждый объект может получать сообщения, обработки данных и отправки сообщений на другиеобъекты.Каждый объект можно рассматривать как независимые «машины» с особой ролью и ответственностью.Действиям (или «методы») над этими объектами тесно связаны с объектом.Например, ООП,структуры данных,как правило, «несут свои собственныеоператоры» (или, по крайней мере, «наследуют» их от аналогичных объектов или классов) - кроме случаев, когда они должны быть последовательными [3].

Простая, без ООП программа представляет собой один «длинный» список инструкций (или команд).Более сложные программы будут часто сгруппированыв видефункцийилиподпрограмм, каждая из которых может выполнять определенную задачу.В конструкции такого рода, она является общей для некоторых из данных программ, чтобы быть «глобальными», т.е. доступны из любой части программы.Вследствии этого программ увеличиваются в размерах, что позволяет любую функцию, чтобы изменить любые данные означает, что ошибки могут иметь далеко идущие последствия.

В отличие от объектно-ориентированного подхода поощеряет программистов для размещения данных, где нет прямого доступа к остальной части программы.Вместо этого, доступ к данным по телефону специально написанных функций, обычно называютметоды, которые в комплекте с данными.Они действуют как посредники для получения или изменения

данных они контролируют.Программная конструкция, которая объединяет данные с набором методов для доступа и управления этими данными, называется объектом.Практика использования подпрограмм, чтобы исследовать или изменить определенные виды данных был использован также в не-ООПмодульного программирования, задолго до широкого использования объектно-ориентированного программирования.

Объектно - ориентированные программы, как правило, содержат различные типы объектов, каждый тип, соответствующий определенному виду комплексу данных, которые будут управлять или, возможно, представляют собой реальный предмет или понятие, как счет в банке, хоккеист и т.д. Программа может также содержать несколько копий каждого типа объекта, по одному для каждой из реальных объектов программа имеет дело.Например, не может быть одного банковского счета объекта для каждого реального счета в определенном банке.Каждая копия объекта банковский счет был бы так в методах, которые она предлагает для манипулирования или чтение своих данных, но данные внутри каждого бъекта будет отличаться отражая различные истории каждой учетной записи [4].

Объекты могут рассматриваться как упаковка со своими данными в наборе функций, предназначенных для того, чтобы данные использовались надлежащим образом, и для оказания помощи в их использовании.Методы объекта, как правило, включают проверку и гарантий, которые являются специфическими для типов данных, содержащихся в объекте. Объект также может предложить простую в использовании, стандартные методы для выполнения определенных операций по его данным, скрывая особенности, как эти задачи решаются.Таким образом, изменения могут быть сделаны к внутренней структуре или методах объекта, не требуя, чтобы остальная часть программы будет изменена. Этот подход может также использоваться, чтобы предложить стандартные методы для различных типов объектов.Например, несколько различных типов объектов может предложить методы печати.Каждый тип объекта может реализовать, что метод печати по-другому, отражающий различные виды данных, каждый из которых содержит, но все различные методы печати можно назвать таким же стандартным образом из других программ. Эти функции становятся особенно полезно, когда более чем один программист вносит свой вклад код проекта или когда целью является повторное использование кода между проектами.

Объектно - ориентированное программирование имеет корни, которые можно проследить в 1960-х годах.Управлять аппаратным и программным обеспечениями становится все более сложным и часто становится проблемой.Исследователи изучили способы поддержания качества программного обеспечения и разработали объектно-ориентированное программирование в частности для решения общих проблем, решительно подчеркивая дискретным, многоразовые единиц логического программирования.Технология направлена на данные, а не процессы, с программами состящие из самодостаточных модулей («классы»), каждый

экземпляр которого («объекты») содержит всю информацию, необходимую для управления своей собственной структуры данных («члены») Это в отличие от существующего модульного программирования, который был доминирующим на протяжении многих лет, который сосредоточил в себе функциимодуля, а не конкретно данных, но не менее предусмотренногоповторного использования кода, и самодостаточной многоразовой логики программирования, позволяющейсотрудничествеза счет использования связанных модулей (подпрограмм) [5].

Инкапсуляция - обеспечивает модульность. Инкапсуляция относится к созданию автономных модулей, которые связываются с функциями обработки данных.Эти пользовательские типы данных называются «классами», и один экземпляр класса «объект».Например, в систему начисления заработной платы, класс может быть Manager, и Пэт и Ян может быть в двух случаях (дваобъекта) в диспетчере класса.Инкапсуляция обеспечивает хорошую модульности кода, который держит отдельные подпрограммы и менее склонны к конфликту друг с другом.

Наследование - передает «Знание» дальше. Классы, созданные в иерархии и наследования позволяет структуры и методов в одном классе, которые передаются вниз по иерархии.Этоозначает, что менее программирования не требуется при добавлении функций сложных систем. Если шаг будет добавлен в нижней части иерархии, то только обработки данных и связанные с этим уникальным шагом должна быть добавлена. Всеостальное, о том, что шаг по наследству.Возможность повторного использования существующих объектов считается одним из основных преимуществ объектной технологии.

Полиморфизм - принимает любую форму. Объектно-ориентированное програм-мирование позволяет процедурам об объектах должны быть созданы, точный тип не известен до момента исполнения.Например, курсор может менять свою форму со стрелки на линии в зависимости от режима программирования.Рутины, чтобы переместить курсор на экране в ответ на движения мыши будут написаны для «курсор», и полиморфизм допускает, что курсор взять на себя все формы требуется во время выполнения.Она также позволяет создавать новые формы, чтобы быть легко интегрированы.

Физическое проектирование при объектном подходе включает объединение классов и других программных ресурсов в программные компоненты, а такжеразмещение этих компонентов на конкретных вычислительных устройствах.

Большинство классов можноотнести к определенному типу, который применительно к данному подходу называют стереотипам, например:

- классы-сущности (классы предметной области);
- граничные (интерфейсные) классы;
- управляющие классы;
- исключения и т. д.

Классы-сущности используют для представления сущностей реального мира или внутренних элементов системы, например структур данных. Как

правило, они не зависят от окружения, и их используют в различных приложениях. Для выявления классов-сущностей изучают описания вариантов использования, концептуальную модель и диаграммы деятельностей [6].

Полученный таким образом список классов-кандидатов фильтруют, удаляя слова, неотносящиеся к предметной области, языковые выражения и т. п. Среди оставшихся отбирают классы-кандидаты, объекты которых обладают как состоянием, так и поведением.

Граничные классы обеспечивают взаимодействие между действующими лицами и внутренними элементами системы. К этому типу относят как классы, реализующие пользовательские интерфейсы, так и классы, обеспечивающиеинтерфейс с аппаратными средствами или программными системами. Для обнаружения граничных классов изучают пары «действующее лицо - вариант использования».

Управляющие классы служат для моделирования последовательного поведения, заложенного в один или несколько вариантов использования.

Список литературы

1. Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К. Объектно-ориентированное программирование: Учеб. для вузов/ Под. Ред. Г.С. Ивановой. - М.: Изд-во МГТУ им. Н.Э.Баумана, 2001. - 320 с.
2. Грэхем И. Объектно-ориентированные методы: Принципы и практика: пер. с англ. Изд. 3-е. - М: Вильямс, 2004. - 880 с.
3. Объектно-ориентированный анализ и проектирование. http://ooad.asf.ru/
4. Гайсарян С.С. Объектно-ориентированнные технологии проектирования прикладных программных систем. http://ooad.asf.ru/books/OOTechnology/INDEX.asp
5. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. http://ooad.asf.ru/files/booch.rar
6. Бадд Т. Объектно-ориентированное программирование. http://ooad.asf.ru/files/Badd_OOP.pdf