УДК 00443:904.9 **Технические науки**

Базеева Н. А., *преподаватель факультета довузовской подготовки и среднего профессионального образования, ФГБОУ ВО «Мордовский государственный университет им. Н.П. Огарёва», г. Саранск*
Петянкин М. Ф., *преподаватель факультета довузовской подготовки и среднего профессионального образования, ФГБОУ ВО «Мордовский государственный университет им. Н.П. Огарёва», г. Саранск*
Соболев Д. С., *студент факультета довузовской подготовки и среднего профессионального образования, ФГБОУ ВО «Мордовский государственный университет им. Н.П. Огарёва», г. Саранск*
e-mail: baz_nat@mail.ru

РАЗВИТИЕ ОБЪЕКТНО - ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Аннотация: В данной статье рассматриваются вопросы, связанные объектно-ориентированным программированием.

Ключевые слова: объект, классы, парадигма, ООП.

Abstract: This article displays issues related to object-oriented programming.

Keywords: object, classes, paradigm, OOP.

Свою историю программирование начало ещё до нашей эры, когда в Древней Греции появился Антикитерский механизм. Он представлял собой калькулятор, в основе которого были различные шестеренки и конфигурации и использовался для отслеживания метонова цикла, который до сих пор используется в лунно-солнечных календарях. В те времена этот процесс много чем отличался от программирования в наши дни [1].

Но время не стоит на месте и спустя более тысячелетия механику из Турции, Аль-Джазари, удалось сконструировать машину, в которой использовались зажимы и кулачки, располагавшиеся в деревянном ящике в определённых местах. Эти механизмы приводили в действие рычаги, которые затем могли управлять ударным инструментом.

Затем, больше чем полвека спустя, во Франции случается революция в ткацком ремесле. Жозеф Мари Жаккар смог усовершенствовать ткацкий станок для создания узоров на тканях при помощи перфокарт.

Чуть позже в этом же веке Английский математик, ученый и изобретатель Чарлз Беббидж сумел разработать «Аналитическую машину», которая стала первым программируемым вычислительным устройством, но он не смог её построить.

Но первую программу для этой машины написала Ада Августа Лавлейс. Программа могла выражать закон сохранения энергии движущейся жидкости. В своей работе она рассматривала принципы работы ячеек памяти и связи формул с процессами вычисления. Другими словами, она рассматривала работу функций. Эти вопросы также имеют свой смысл и в современной трактовке программирования [2].

Затем после ряда войн 20 столетия, начался стремительный рост развития вычислительных машин и технологий для разработки под них программного обеспечения. Здесь и начинается история современного программирования.
Объектно-ориентированное программирование (далее ООП) представляет собой один из методов программирования, идея которого была основана на представление программы, как совокупность объектов. Каждый объект является экземпляром класса, с которым он связан. Классы же в свою очередь образуют наследующуюся иерархию.

Если взглянуть с идеологической точки зрения, то ООП будет больше напоминать моделирование информационных объектов, так как классы являются отображениями реальных сущностей. ООП стало заменять структурной программирования, путём бола рационального решения его

основной задачи: структурирование данных с точки зрения управления ими. При реализации крупных проектов это бывает чересчур необходимо, поскольку это улучшает процесс управления и процесс моделирования.

Предполагается, что система будет управляемой для иерархических систем и это поможет минимизировать избыточность данных и их целостность. Поэтому то, что удобно-управляемо, будет и удобно-понимаемо. Таким образом, решая задачу управляемости, мы решаем и другую важную задачу – транслирование понимания задачи программистом в более удобную для дальнейшего изучения и использования форму [3].

От предметной области и задачи будет зависеть выбор принципа структурирования данных для обеспечения оптимального управления выбранной модели. Речь идёт о следующих принципах:

1. Полиморфизм.

Этот метод направлен на определение точки, где единое управление лучше разделить или наоборот, собрать воедино разделённые управления.

2. Инкапсуляция.

Метод быстрой и надёжной организации своей иерархической управляемости. Одна команда должна выполнять какое-либо конкретное действие без уточнения о том, как оно должно быть выполнено.

3. Наследование.

Метод для быстрой организации реализации родственных механизмов. Спускаясь вниз по иерархической лестнице, нужно учитывать только изменение новых объектов. Всё что имеется на предыдущих шагах или предках не должно учитываться, это должно оставаться не тронутым с каждым последующим изменением.

4. Абстрагирование.

В этом методе учитывается все самое важное в предмете, для моделирования и решения конкретной задачи. В конечном итоге от того, как мы поняли предмет, будет зависеть его реализация в виде класса.

Говоря понятными словами, организация информации прогрессирует. В первую очередь ставятся самые важные критерии. Прогрессируя таким образом можно перейти на ещё более высокий уровень детализации. В конечном итоге процесс может замкнуться, если не будет его продолжения.

Для детального разбора, необходимо определить основные понятия.

Абстракция данных. Она представляет собой рассмотрения все информации, но выведения на передний план самой важной и на задний план малозначимой. В ООП абстракция подразумевает собой наиболее значимую информацию об объекте, которая открыта всей программе.

Инкапсуляция. В этой концепции данные и методы для их обработки объединяются в одном классе.

Наследование. Свойство системы, которое заимствует у уже имеющихся классов функционал при создании нового элемента. Эти классы могут называться по-разному. Класс, от которого происходит наследование, может называться базовым, родительским или суперклассом. Новый класс может быть потомком, наследником, дочерним классом или производным классом.

Полиморфизм. Это свойство помогает системе, без какой-либо информации о структуре системы, использовать для неё объекты с одинаковым интерфейсом.

Класс. Он является комплексным типом данных, состоящим из набора полей, функций, для работы с ними. Класс представляет собой описание сущности с внутренним и внешним интерфейсами, для работы со своими полями.

Объект. При создании экземпляра класса в адресном пространстве вычислительной системы появляется сущность, которая и называется объектом. ООП пришло к нам в результате развития процедурного программирования. В процедурном программирование не было формальной связи между данными и подпрограммами.

В ООП все основывается на классах и объектах. Объекты взаимодействуют между собой при помощи сообщений. И в дальнейшем ООП

станет агентно-ориентированным, где в роли агентов будут выступать независимые части кода при выполнении. Это взаимодействие происходит по средствам изменяемой среды.

Первым языком программирования, в котором использовались все основные понятия ООП, был «Симула». В нем были использованы новые идеи того времени: классы, объекты, виртуальные методы. Но никто не увидел в этом какого-то особого потенциала. По структуре классов в «Симуле» можно было сказать, что какой-либо сложный объект описывался множественными примитивами.

Но первым объектно-ориентированным языком программирования, получившим широкое распространение, стал «Смолток». После пересмотра Аланом Кейем и Деном Ингаллсом концепция процедурного программирование или предложили своё новое видение ООП. Понятие класса стало основной основообразующей идеей для всех конструкций языка. Здесь сам класс стал примитивом. С его помощью описывались более сложные и массивные конструкции.

В данный же момент языков программирования поддерживающих концепции ООП очень много. Они также построены на разных моделях. На объектной модели «Симулы» основаны такие известные языка как: C++, C#, Delphi, Java. Языки Objective-C, Python, Ruby основываются на модель «Смолтока».

После появления ООП его развитие не останавливалось. И следующими этапами в этом цикле являлись Компонентное программирование, прототип-ориентированное программирование и класс-ориентированное программирование. Они по-разному подходят к созданию программ и каждый из них имеет как преимущества, так и недостатки.

Компонентное программирование или компонентно-ориентированное программирование является дополнением к ООП, в котором существуют определённые правила и ограничения, для помощи, при построении крупных программных систем с долгим жизненным циклом. Изменения в структуру

вносятся при помощи создания новых компонентов в качестве дополнения или замены уже имеющихся. При этом полное наследование реализации запрещается, возможно унаследовать лишь интерфейсы базового компонента. Этим решается проблема хрупкости базового класса.

Прототип-ориентированное программирование сохранило часть ООП, но при этом отказалось от классов и наследования. Здесь имеется прототип – образец объекта, по чьему образу создаются другие объекты. При этом объекты копии могу автоматически получать изменения в прототипе [4].

Каждый новый объект не будет являться экземпляром класса. Он является экземпляром без класса. Он также может стать прототипом и использоваться для создания новых копий при помощи операции клонирования.

Класс-ориентированное программирование. Здесь данные и поведение имеют неразрывные связи между собой. Вместе они и представляют собой класс. Как и в ООП здесь остались понятия классов и экземпляров. Экземпляр в этой модели является носителем данных. Поведение, которое задано классом может менять состояние экземпляра. Получающийся экземпляр жестко заданные классом структуру и поведение.

Говоря об объектно-ориентируемом программирование, хочется сказать, что оно в наше время очень востребовано. При помощи его концепций решается большинство прикладных задач. Оно развивается и совершенствуется. Не исключено что на его основе появится более модернизированная концепция, с помощью которой поиск оптимальных решений упростится до минимума.

Библиографический список:

1. Иванова, В.В. Основы бизнес-информатики. — Санкт-Петербург: СПбГУ, 2014. — 244 с.

2. Садовская, Т.Г. Бизнес-информатика и сетецентрические системы управления. — Москва: МГТУ им. Н.Э. Баумана, 2014. — 48 с.

3. Алябьева, Е.В. Имитационное моделирование. Барнаул: АлтГПУ, 2016. — 48 с.

4. Емельянов, А.А. Имитационное моделирование экономических процессов. — Москва: Финансы и статистика, 2009. — 416 с.