УВАЖАЕМЫЕ СТУДЕНТЫ! Законспектируйте в своей рабочей тетради по дисциплине приведенную лекцию (объемом 4-5 страницы), ответьте *письменно* на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 30.01.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 33
Тема «Преимущества применения объектно-ориентированного подхода в программировании»

Считается что объектно-ориентированное программирование основано на трех столпах, которые предоставляют программисту преимущества по сравнению с процедурным подходом. Ими являются — инкапсуляция, наследование и полиморфизм.

Наследование позволяет при использовании объектно-ориентированного подхода существенно избавиться от дублирования кода. Но используя его, всегда стоит помнить про один из принципов SOLID, называющийся Liskov Substitution, который, если особо не вдаваться в подробности, гласит, что наследование должно использоваться только как реализация отношения «является». То есть класс потомок должен «являться» подвидом родителя. Например у вас может быть класс Bird (птица) и его подклассы Sparrow (воробей) и Raven (ворон), которые, например, наследуют (и могут расширять или изменять) метод fly. Однако, если же вы создадите потомка Bird, называющегося Penguin (пингвин) и его метод fly, например, будет бросать исключение (потому что пингвины не летают), это будет нарушать принцип Liskov Substitution. Говоря о наследовании, также стоит

упомянуть один важный принцип, которого советует придерживаться «Банда четырёх» (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides) в своей книге Design Patterns: Elements of Reusable Object-Oriented Software (1994), который гласит, что нужно стараться использовать композицию (когда один объект «содержит» другой объект) вместо наследования настолько, насколько возможно.

Объектно-ориентированное программирование (ООП) – это методика разработки программ, в основе которой лежит понятие класса как некоторой структуры, описывающей совокупность однотипных объект реального мира, их поведение. Задача, решаемая с использованием методики ООП, описывается в терминах классов и операций, производимыми над объектами этого класса. Программа при таком подходе представляет собой набор реализованных объектов и связей между ними. Другими словами, можно сказать, что объектно-ориентированное программирование представляет собой метод программирования, который весьма близко напоминает наше поведение.

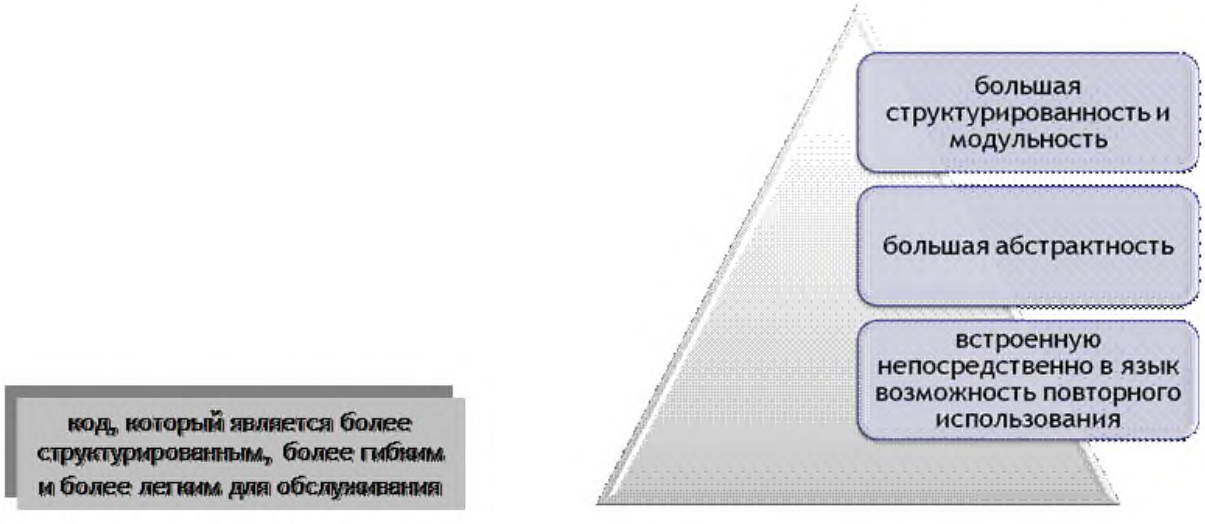

Рисунок 1 – Основные средства объектно-ориентированного языка программирования

Достоинства ООП:

От любого метода программирования мы ждем, что он поможет нам в решении наших проблем. Но одной из самых значительных проблем в программировании является сложность. Чем больше и сложнее программа, тем важнее становится разбить ее на небольшие, четко очерченные части. Чтобы побороть сложность, мы должны абстрагироваться от мелких деталей. В этом смысле классы представляют собой весьма удобный инструмент.

– Классы позволяют проводить конструирование из полезных компонент, обладающих простыми инструментами, что дает возможность абстрагироваться от деталей реализации.

– Данные и операции вместе образуют определенную сущность, и они не «размазываются» по всей программе, как это нередко бывает в случае процедурного программирования.

– Локализация кода и данных улучшает наглядность и удобство сопровождения программного обеспечения.

– Инкапсуляция информации защищает наиболее критичные данные от несанкционированного доступа.

ООП дает возможность создавать расширяемые системы (extensible systems). Это одно из самых значительных достоинств ООП и именно оно отличает данный подход от традиционных методов программирования. Расширяемость (extensibility) означает, что существующую систему можно заставить работать с новыми компонентами, причем без внесения в нее каких-либо изменений. Компоненты могут быть добавлены на этапе выполнения.

Расширение типа (type extension) и вытекающий из него полиморфизм переменных оказываются полезными преимущественно в следующих ситуациях.

– Обработка разнородных структур данных. Программы могут работать, не утруждая себя изучением вида объектов. Новые виды могут быть добавлены в любой момент.

– Изменение поведения во время выполнения. На этапе выполнения один объект может быть заменен другим. Это может привести к изменению алгоритма, в котором используется данный объект.

– Реализация родовых компонент. Алгоритмы можно обобщать до такой степени, что они уже смогут работать более, чем с одним видом объектов.

– Доведение полуфабрикатов. Компоненты нет надобности подстраивать под определенное приложение. Их можно сохранять в библиотеке в виде полуфабрикатов (semifinished products) и расширять по мере необходимости до различных законченных продуктов.

– Расширение каркаса. Независимые от приложения части предметной области могут быть реализованы в виде каркаса и в дальнейшем расширены за счет добавления частей, специфичных для конкретного приложения.

Многоразового использования программного обеспечения на практике добиться не удается из-за того, что существующие компоненты уже не отвечают новым требованиям. ООП помогает этого достичь без нарушения работы уже имеющихся клиентов, что позволяет нам извлечь максимум из многоразового использования компонент.

– Мы сокращаем время на разработку, которое с выгодой может быть отдано другим проектам.

– Компоненты многоразового использования обычно содержат гораздо меньше ошибок, чем вновь разработанные, ведь они уже не раз подвергались проверке.

– Когда некая компонента используется сразу несколькими клиентами, то улучшения, вносимые в ее код, одновременно оказывают свое положительное влияние и на множество работающих с ней программ.

– Если программа опирается на стандартные компоненты, то ее структура и пользовательский интерфейс становятся более унифицированными, что облегчает ее понимание и упрощает ее использование.