ОБЪЁКТНО-ОРИЕНТИ́РОВАННОЕ ПРОГРАММИ́РОВАНИЕ

ТРЕНЕР: АЮПОВ Р.Х.

❖**Объе́ктно-ориенти́рованное программи́ рование (ООП)** — парадигма программирования, в которой основными концепциями являются понятия объектов и классов. В случае языков с прототипированием вместо классов используются объекты-прототипы.

❖ **Абстракция**

❖ Абстрагирование — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Соответственно, абстракция — это набор всех таких характеристик.[1]

❖ **Инкапсуляция**

❖ Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя.[1]

❖ Наследование

❖ Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.

❖ **Полиморфизм**

❖ Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.[1]При использовании термина «полиморфизм» в сообществе ООП подразумевается полиморфизм подтипов; а использование параметрического полиморфизма называют обобщённым программированием.

❖ Класс

❖ Класс является описываемой на языке терминологии исходного кода моделью ещё не существующей сущности (объекта). Фактически он описывает устройство объекта, являясь своего рода чертежом. Говорят, что объект — это **экземпляр** класса. При этом в некоторых исполняющих системах класс также может представляться некоторым объектом при выполнении программы посредством динамической идентификации типа данных. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.

❖ **Объект**

❖ Сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса или копирования прототипа (например, после запуска результатов компиляции и связывания исходного кода на выполнение).

❖ **Прототип**

❖ Прототип — это объект-образец, по образу и подобию которого создаются другие объекты. Объекты-копии могут сохранять связь с родительским объектом, автоматически наследуя изменения в прототипе; эта особенность определяется в рамках конкретного языка.

- В центре ООП находится понятие *объекта*. Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. Сокрытие данных называется инкапсуляцией.
- Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

❖ Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм — возможность объектов с одинаковой спецификацией иметь различную реализацию.

ОСОБЕННОСТИ РЕАЛИЗАЦИИ

- Как уже говорилось выше, в современных объектно-ориентированных языках программирования каждый объект является значением, относящимся к определенному классу. Класс представляет собой объявленный программистом составной тип данных, имеющий в составе:
- **Поля данных**
- Параметры объекта (конечно, не все, а только необходимые в программе), задающие его состояние (свойства объекта предметной области). Иногда поля данных объекта называют свойствами объекта, из-за чего возможна путаница. Физически поля представляют собой значения (переменные, константы), объявленные как принадлежащие классу.

- **Контроль доступа**
- Поскольку методы класса могут быть как чисто внутренними, обеспечивающими логику функционирования объекта, так и внешними, с помощью которых взаимодействуют объекты, необходимо обеспечить скрытость первых при доступности извне вторых. Для этого в языки вводятся специальные синтаксические конструкции, явно задающие область видимости каждого члена класса. Традиционно это модификаторы public, protected и private, обозначающие, соответственно, открытые члены класса, члены класса, доступные только из классов-потомков, и скрытые, доступные только внутри класса. Конкретная номенклатура модификаторов и их точный смысл различаются в разных языках.

- **Методы доступа**
- Поля класса в общем случае не должны быть доступны извне, поскольку такой доступ позволил бы произвольным образом менять внутреннее состояние объектов. Поэтому поля обычно объявляются скрытыми (либо язык в принципе не позволяет обращаться к полям класса извне), а для доступа к находящимся в полях данным используются специальные методы, называемые методами доступа. Такие методы либо возвращают значение того или иного поля, либо производят запись в это поле нового значения. При записи метод доступа может проконтролировать допустимость записываемого значения и, при необходимости, произвести другие манипуляции с данными объекта, чтобы они остались корректными (внутренне согласованными). Методы доступа называют ещё аксессорами (от англ. *access* — доступ), а по отдельности — геттерами (англ. *get* — чтение) исеттерами (англ. *set* — запись)[6].

МЕТОДЫ ДОСТУПА

- Поля класса в общем случае не должны быть доступны извне, поскольку такой доступ позволил бы произвольным образом менять внутреннее состояние объектов. Поэтому поля обычно объявляются скрытыми (либо язык в принципе не позволяет обращаться к полям класса извне), а для доступа к находящимся в полях данным используются специальные методы, называемые методами доступа.

❖ Такие методы либо возвращают значение того или иного поля, либо производят запись в это поле нового значения. При записи метод доступа может проконтролировать допустимость записываемого значения и, при необходимости, произвести другие манипуляции с данными объекта, чтобы они остались корректными (внутренне согласованными).

СВОЙСТВА ОБЪЕКТА

- Псевдополя, доступные для чтения и/или записи. Свойства внешне выглядят как поля и используются аналогично доступным полям (с некоторыми исключениями), однако фактически при обращении к ним происходит вызов методов доступа. Таким образом, свойства можно рассматривать как «умные» поля данных, сопровождающие доступ к внутренним данным объекта какими-либо дополнительными действиями (например, когда изменение координаты объекта сопровождается его перерисовкой на новом месте).

❖ **Разреженные файлы** в Windows это такие файлы, которые занимают меньше дискового пространства, чем их реальный размер. При этом не используется сжатие данных. Экономия места в разреженных файлах достигается за счёт компактного представления пустых областей файла, которые содержат одни лишь нули (символы 0x00). По-английски разреженные файлы называются sparse files.

- Файл делится на логические блоки размером 64 Кб. Анализируется каждый такой блок. Если любой байт в пределах блока имеет значение, отличное от охоо, значит этот блок не пустой, он содержит данные. Блок, содержащий данные пишется на диск. Если же все байты в пределах блока имеют одинаковое значение охоо, такой блок считается пустым. Содержимое такого блока вообще не сохраняется на диск.
- При считывании файла драйверу NTFS известна общая длина файла и набор из нескольких блоков данных. Если происходит чтение из файла по смещению, которое расположено в пределах одного из имеющихся блоков данных, драйвер NTFS производит чтение с диска из соответствующего блока.

- Если происходит чтение из области, не совпадающей ни с одним из имеющихся блоков, драйвер NTFS отдаёт символы охоо. Ведь известно, что раз подходящего блока данных нет, значит он не был записан на диск, следовательно он является пустым блоком, состоящим из одних лишь нулей.
- Таким образом происходит экономия дискового пространства, ведь на диске оказываются записанными только те блоки, которые реально содержат данные. Всё остальное пустое пространство файла не хранится на диске, а генерируется драйвером на лету, если происходит чтение из этих областей.

ДВОИЧНОЕ ПРЕДСТАВЛЕНИЕ ДАННЫХ

- С помощью программы NTFS Stream Explorer 1.07 я исследовал внутреннее устройство разреженных файлов. Вообще-то эта версия не показывает, как устроены такие файлы на диске, вместо этого она показывает их в виде формата Microsoft Backup (используемого в функциях BackupRead и BackupWrite). Это особый формат представления файлов, созданный Microsoft для целей архивации и резервного копирования. Устройство разреженных файлов в этом формате может не отображать реального положения вещей, но может показать некоторые принципы хранения таких файлов.