Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) — это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Объект понимается как осязаемая реальность, проявляющая четко выделяемое поведение.

* В качестве объекта может выступать:
* осязаемый и (или) видимый предмет;
* нечто, воспринимаемое мышлением;
* нечто, на что направлена мысль или действие."

Класс— это некое множество объектов, имеющих общую структуру и общее поведение.

Состояние объекта характеризуется набором конкретных значений— полей.

Поведение определяется набором значений некоторых полей, процедур и функций, которые оперируют данными объекта. В рамках ООП эти процедуры и функции получили название методов. Например, в программной системе может быть определен класс Client, описывающий некоторого абстрактного клиента фирмы. Все экземпляры класса Client (т.е. конкретные клиенты) имеют:

* перечень свойств:— индивидуальный номер ( id ), имя ( first\_ name ), фамилия ( second\_name ), дата рождения ( birth\_date ) и др.;
* конкретные значения свойств каждого экземпляра: Aleksey (имя), Petrov (фамилия), 10.06.1975 (дата рождения);
* определенное поведение, задающееся набором действий и состояний, которые непосредственно связаны с конкретным клиентом. Например, для каждого клиента можно получить выписку его активности (метод activity ), узнать состояние обслуживания (поле service\_status может иметь значения enable — обслуживается, disable — не обслуживается, paused — приостановлено);
* идентичность — совокупность вышеперечисленных свойств, определяющих уникальность каждого клиента."

В рамках ООП выделяют несколько базовых понятий, существенных для практического программирования. К таким понятиям относятся наследование, инкапсуляция и полиморфизм.

Применение объектно-ориентированных технологий в практике программирования

Пространства имен в .NET Framework. Количество процедур и функций, реализованных в стандартном модуле PABCSystem, оказывается все же недостаточным для написания сколь-нибудь серьезных программ. Модуль содержит базовый набор операций, констант и типов, на основе которых строятся процедуры, функции, пользовательские типы данных, классы."

Основными компонентами.NET Framework являются:

1) общеязыковая исполняющая среда ( Common Language Runtime, CLR);

2) библиотека классов ( Framework Class Library, FCL).

Черпаков, И. В.

Основы программирования : учебник и практикум для среднего профессионального образования / И. В. Черпаков. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 196 с. — (Профессиональное образование). — ISBN 978-5-534-18760-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 176 — URL: https://urait.ru/bcode/545507/p.176 (дата обращения: 07.11.2024).

Инкапсуляция

Под инкапсуляцией понимается возможность объединить данные и

методы их обработки в рамках класса, а также скрыть реализацию методов от

пользователя и других классов. Инкапсуляция применяется для защиты данных и

методов от некорректного изменения.

Черпаков, И. В.

Основы программирования : учебник и практикум для среднего профессионального образования / И. В. Черпаков. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 196 с. — (Профессиональное образование). — ISBN 978-5-534-18760-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 157 — URL: https://urait.ru/bcode/545507/p.157 (дата обращения: 07.11.2024).

Инкапсуляция означает, что каждый объект полностью описывается совокупностью своих свойств и методов. Инкапсуляция позволяет скрыть внутреннюю организацию объекта, не влияющую на его внешнее поведение. Таким образом, инкапсуляция позволяет в максимальной степени изолировать объект от внешнего окружения. Основной единицей инкапсуляции в ООП является класс. Класс описывает данные, определяющие состояние объекта, и функции, определяющие поведение объекта. Инкапсуляция обеспечивает сокрытие элементов-данных и элементов-функций (в классе с управлением доступом к ним.

Гниденко, И. Г.

Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 248 с. — (Профессиональное образование). — ISBN 978-5-534-18131-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 17 — URL: https://urait.ru/bcode/539215/p.17 (дата обращения: 07.11.2024).

Языки программирования

Язык C# — объектно-ориентированный язык со строгой типизацией.

Тип в программировании — понятие первичное. Тип некоторой сущности декларирует для нее совокупность возможных состояний и набор допустимых действий. Понятие сущности мы пока не уточняем, сущностями могут быть константы, переменные, массивы, структуры и т. д.

Наиболее часто понятие тип в языках программирования используют в связи с понятием «переменная».

Примитивное (но пока достаточное для наших целей) определение: переменная это пара «обозначение переменной + значение переменной».

Для переменной тип вводит совокупность ее возможных значений и набор допустимых операций над этими значениями.

Пример определения переменной в С, С++, C# и некоторых других языках:

int spot = 16;

spot — обозначение (имя) переменной, 16 — ее значение в данный момент (после этого определения), int — название типа переменной.

В стандарте C# в связи с типами используется выражение «унифицированная система типов». Смысл унификации состоит в том, что все типы происходят от класса object, т. е. являются производными от этого класса и наследуют его члены. О механизме наследования речь пойдет позднее, но в примерах программ мы будем применять для объектов и классов некоторые возможности, унаследованные ими от класса object.

Типы C# позволяют представлять, во-первых, те «скалярные» данные, которые используются в расчетах (целые и вещественные числа, логические значения) и в обработке текстов (символы, строки). Вторая группа типов соответствует специфическим для программирования на языках высокого уровня «агрегирующим» конструкциям: массивам, структурам, объектам (классов).

Такое деление типов на две названные группы унаследовано языком C# из предшествующих ему языков программирования.

Классификация типов C#

Система типов языка С# — вещь достаточно сложная и требующая аккуратного рассмотрения при первом знакомстве. Общие отношения между типами иллюстрирует иерархическая схема, приведенная на рис. 2.1. Как уже упоминалось и как показано на схеме, все типы языка C# имеют общий базовый тип — класс object. О делении типов на типы ссылок и типы значений мы уже рассказывали. А вот с дальнейшей детализацией будем знакомиться постепенно. И знакомство начнем не с классификации, соответствующей иерархии типов, а с другого деления.

Все типы, которые могут использоваться в программах на C#, делятся на три группы:

• предопределенные в языке C# (в Стандарте они обозначены термином Built\_In, который можно перевести как «встроенные»);

• библиотечные (обычно из стандартной библиотеки .NET Framework);

• определенные программистом (пользовательские).

Предопределенные типы включены в язык C#. К ним относятся:"

Предопределенные типы включены в язык C#. К ним относятся:

1. object — тип ссылок (класс), который является первоначальным (единственным исходным) базовым для всех других типов языка C#, т. е. все другие типы являются производными от этого типа;"
2. " 2) простые (базовые или фундаментальные) типы — типы значений, для которых в языке C# введены специальные обозначения;

3) string — тип ссылок (класс) для представления строк — последовательностей символов в кодировке Unicode, и др."

Библиотечные и пользовательские типы могут быть как типами значений, так и типами ссылок. Чтобы пользоваться библиотечным типом, нужно знать его имя и возможности (поля, методы), а также название того пространства имен, которому он принадлежит.

Примечание. Как мы уже говорили, для сокращения квалифицированного

(полного) имени нужного нам класса (типа) в программу включают директивы

using название\_пространства\_имен ;

using static тип ;

Например, чтобы написать программу на C# для работы с файлами, в коде

используется директива:

using System.IO;

После этого в программе становятся доступны с помощью сокращенных

имен классов типы, необходимые для организации потокового ввода-вывода."

Подбельский, В. В.

Программирование. Базовый курс С# : учебник для среднего профессионального образования / В. В. Подбельский. — Москва : Издательство Юрайт, 2024. — 369 с. — (Профессиональное образование). — ISBN 978-5-534-11467-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 27 — URL: https://urait.ru/bcode/542233/p.27 (дата обращения: 07.11.2024).