Основные понятия ООП: объекты, классы и методы

Содержание лекции

Основные работы в сфере моделирования ООП(1)

1924 – М.Шейнфинкель (Moses Shönfinkel) разработал теорию «простых» функций

1934 – А.Черч (Alonso Church) изобрел ламбда-исчисление и применил его в исследовании теории множеств

1971 – Д.Скотт (Dana S. Scott) предложил использовать полные и непрерывные решетки для формализации семантики (типизированного) ламбда-исчисления

Основные работы в сфере моделирования ООП (2)

1980-е – Д.Скотт (Dana S. Scott) и М.Фурман (Michael P. Fourman) исследовали аппарат определенных дескрипций как средство формализации определений

1990-е – В.Э.Вольфенгаген (Vyatcheslav E. Wolfengagen) предложил схему двухуровневой концептуализации для моделирования объектов предметных областей (и языков программирования); данная модель адекватна как для объектов данных (ОД), так и для объектов метаданных (ОМД)

Интуитивные определения основных понятий

Объектом называется математическое представление сущности реального мира (или предметной области), которое используется для моделирования.

Классом называется весьма общая сущность, которая может быть определена как совокупность элементов.

Свойством (или атрибутом) называется пропозициональная функция, определенная на произвольном типе (данных).

Методом (или функцией) называется операция, определенная над объектами некоторого класса.

Соотношение понятий объекта и класса

Понятие класса является более общим, чем понятие объекта.

Объект является экземпляром класса.

Класс может рассматриваться как совокупность объектов (подобно тому, как множество есть совокупность элементов).

Класс может быть элементарным или подразделяться на подклассы (подобно тому как множество подразделяется на подмножества).

Например, класс **PERSON** содержит подкласс **STUDENT**, который, в свою очередь, содержит объект **John_Smith**.

Концептуализация как модель объекта

1. Модель основана на типизированном λ-исчислении, семантика которого моделируется посредством полных и непрерывных решеток Д.Скотта.

2. Вводятся аппликативные структуры с приписыванием типов.

3. Для формализации определений используются определенные дескрипции Скотта-Фурмана вида:

IxФ, что означает “тот единственный x, для которого Ф истинно”.

4. Произвольный объект моделируется тройкой

<концепт, индивид, состояние>,

составляющие которой соединены соотнесениями.

Принцип концептуализации

$$||\mathbf{I}x\Phi(x)||i = d \Leftrightarrow \{d\} = \{d \in D \mid ||\Phi(d)||i = true\}$$

Значением индивидного концепта является функция из соотнесений в индивиды

Произвольный объект d предметной области D может быть единственным образом индивидуализирован посредством функции Ф (где i означает соотнесение).

D можно также интерпретировать как класс, а d – как объект языка программирования.

Классы в C#

- Ссылочный тип, определенный пользователем (аналогично языкам C++ и Java)
- Единичное наследование классов
- Множественное наследование интерфейсов
- Члены (элементы) класса:

  - константа, поле, метод, оператор, конструктор, деструктор;
  - свойство, индексатор, событие;
  - статические и инициализированные члены.

- Доступ к членам класса (`public, protected, private`(по умолч.),`internal,protected internal`)
- Инициализация – посредством оператора `new`

Описание и использование классов в C# (1)

```
class C {
  …
  int value = 0;
  …
  }
```

Инициализация поля (значения) факультативна и не должна давать доступа к полям и методам того же типа

Доступ внутри класса C: `... value ...`

Доступ из других классов:

```
C c = new C();
... c.value ...
```

Описание и использование классов в C# (2)

Описание:

```
class Rectangle {
  Point origin;
  public int width, height;
  public Rectangle()
  {origin = new Point(0,0); width=height=0;}
     public Rectangle (Point p, int w, int h)
       {origin = p; width = w; height = h;}
     public void MoveTo (Point p) {origin = p;}
  }
```

Применение:

```
Rectangle r=new Rectangle(new Point(10,20),5,5);
int area = r.width * r.height;
r.MoveTo(new Point(3, 3));
```

Статические поля объекта в языке C#

Принадлежат классу, а не объекту:

```
class Rectangle {
   static Color defaultColor;// для каждого класса
   static readonly int scale; // для каждого класса
                  // статические константы недопустимы
   int x, y, width,height;   // для каждого объекта
      ...
   }
```

Доступ изнутри класса:

```
... defaultColor ... scale ...
```

Доступ из других классов:

```
... Rectangle.defaultColor
... Rectangle.scale ...
```

Пример использования метода в языке C#

```
class C {
    int sum = 0, n = 0;
    public void Add (int x)
{   sum = sum + x; n++; // процедура}
    public float Mean(){
     return(float)sum/n; //ф-ция(должна возвратить знач.)
    }
}
```

Доступ изнутри класса:

```
this.Add(3);
float x = Mean();
```

Доступ из других классов:

```
C c = new C();
c.Add(3);
float x = c.Mean();
```

Пример статического метода в языке C#

Операции над данными классов (статическими полями):

```
class Rectangle {
  static Color defaultColor;

  public static void ResetColor() {
        defaultColor = Color.white;
  }
}
```

Доступ изнутри класса:

```
ResetColor();
```

Доступ из других классов:

```
Rectangle.ResetColor();
```

Обработка и наследование классов и объектов в C#

```
class Stack {
  int[] values;
  int top = 0;
  public Stack(int size) { ... }
  public void Push(int x) {...}
  public int Pop() {...}
}
```

Объекты хранятся в куче (для классов и ссылочных типов)

- Объекты необходимо инициализировать оператором `new`:

```
Stack s = new Stack(100);
```

- Классы могут наследовать свойства других классов (единичное наследование кода)
- Классы могут реализовывать множественные интерфейсы (множественное наследование типов)

Преимущества и недостатки объектных теорий

Преимущества:

- интуитивная близость произвольной предметной области;
- возможность моделирования сколь угодно сложной предметной области;
- событийно-ориентированный подход;
- высокий уровень абстракции;
- возможность повторного использования описаний;
- параметризация методов обработки объектов

Недостатки:

- сложность тестирования и верификации программ

Библиография (1)

1. Schönfinkel M. ‘Über die Bausteine der matematischen Logik, Math. Annalen 92, pp. 305-316, 1924. Translation printed as ‘On the building blocks of mathematical logic’, in van Heijenoort, J. (ed.), From Frege to Gödel, Harvard University Press, 1967
2. Church A. The calculi of lambda-conversion.- Princeton, 1941, ed. 2, 1951
3. Barendregt H.P. The lambda calculus (revised edition), Studies in Logic, 103, North Holland, Amsterdam, 1984
4. Scott D.S. The lattice of flow diagrams.- Lecture Notes in Mathematics, 188, Symposium on Mathematics of Algorithmic Languages.- Springer-Verlag, 1971, p.p. 311-372

Библиография (2)

5. Scott D.S. Identity and existence in intuitionistic logic.- In: Application of Sheaves.- Berlin: Springer, 1979, p.p. 600-696
6. Fourman M. The logic of topoi. In: Handbook of Mathematical Logic, J.Barwise et al., eds. North-Holland, 1977
7. Wolfengagen V.E. Event-driven objects. In: Proc. CSIT'1999, Moscow, Russia, 1999, Vol.1, p.p. 88-96
8. Wolfengagen V.E. Fuctional notation for indexed concepts. In: Proc. WFLP'2000, Benicassim, Spain, Sept. 2000. http://www.dsic.upv.es/~wflp2000/
9. Scott D.S. Domains for denotational semantics. ICALP 1982, 577-613