Объектно–ориентированное программирование

Информатика
10-11 классы

26 февраля 2012 г.

Что такое ООП?

- Динамическое и функциональное виды программирования, как известно, решают весьма важную задачу разделения *бизнес–логики* приложения от низкоуровневых алгоритмов.
- Когда мы используем автомобиль, мы не задумываемся о его устройстве, а просто используем различные способы управления.
- При этом даже те инженеры, которые разрабатывают автомобили, имеют свои специализации: часть занимается двигателем, часть — дизайном, часть — безопасностью, а кто-то — и концепцией в целом.

Что такое ООП?

- Концепция *объектно–ориентированного программирования* (ООП) предлагает оперировать в программе не переменными и функциями, а *объектами*.
- Всё в программе является объектами.
- У объекта имеются свойства и методы.
- Свойства представляют собой переменные, принадлежащие объекту.
- Методы — функции, позволяющие получить / изменить информацию об объекте.

Объект Кот

Какие свойства есть у кота?

Свойства кота

- Порода
- Цвет
- Рост
- Возраст
- Дата последнего кормления
- Дата последнего поглаживания
- Дата последнего мяукания
- ...

А методы?

Методы кота

- Мяукнуть
- Поесть
- Потребовать погладить
- Погулять
- ...

А что с другими животными?

Упс, не то

Собака

Сравнение свойств Кота и Собаки

- Порода
- Цвет
- Рост
- Возраст
- Дата последнего кормления
- Дата последнего поглаживания
- **Дата последнего мяукания**

- Порода
- Цвет
- Рост
- Возраст
- Дата последнего кормления
- Дата последнего поглаживания
- **Дата последнего гавкания**

Сравнение методов Кота и Собаки

- **Мяукнуть**
- Поесть
- Потребовать погладить
- Погулять

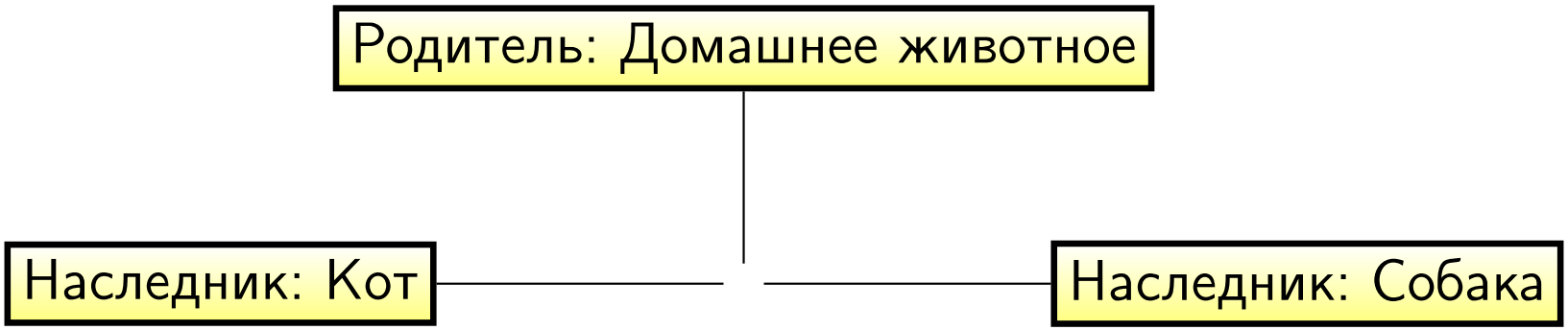
- **Гавкнуть**
- Поесть
- Потребовать погладить
- Погулять
- **Выгуляться**

Домашние животные

Принцип наследования

Общие свойства и методы объектов можно вынести в класс–**родитель**. Все "дети"–наследники автоматически получают их.

Схема наследования



Несколько нудных терминов

- Одинаковые объекты являются **экземплярами класса**.
- Кот — это, на самом деле, класс.
- А вот, например, кот Вася — это объект, то есть, представитель класса.
- Класс — это программная структура.
- В программе мы сначала создаём класс, а потом уже создаём (**инстанцируем**) объекты.
- В ruby всё что угодно является объектом. Даже число 5, строка “мама мыла раму” и пр.

Класс Многоугольник

- Создадим класс Многоугольник.
- Базовые свойства фигуры: стороны фигуры, углы, периметр, площадь и др.
- Методы: посчитать площадь, посчитать периметр, найти радиус описанной окружности и др.
- Фигуры бывают разные: треугольник, четырёхугольник (среди которых тоже есть квадрат, ромб и пр.)
- У каких-то фигур мы знаем, как считать площадь и пр., а у каких-то — нет.
- Напишем программу.

Класс Многоугольник

Listing 1: Класс Многоугольник

```
class Polygon
  attr_accessor :sides, :corners,
                :perimeter, :square
  def perimeter
    @perimeter = @sides.inject(0){|res, elem|
                                  res + elem}
  end
  def num_points
    @sides.size
  end
end
```

Пояснения к классу

- Методы класса определяются точно так же, как и обычные функции. Отличий нет.
- Свойства класса мы будем определять через специальную конструкцию **attr_accessor**. Не вдаваясь в детали, просто перечислим все свойства–переменные. Обратите внимание, что они начинаются со знака “двоеточие”.
- Чтобы внутри метода обратиться к свойству, нужно перед его (свойства) названием поставить знак @.
- В данном классе мы определяем методы perimeter и num_points (количество вершин). Мы специально метод perimeter назвали одинаково со свойством, чтобы при вызове obj.perimeter происходило автоматическое вычисление.

Используем класс Polygon

Listing 2: Использование Polygon

```
fig = Polygon.new
fig.sides = [2,4,2,4]
fig.corners = [90, 90, 90, 90]

puts fig.perimeter
```

- Для создания экземпляра класса используется конструкция CLASS.new. Аналог — ручное создание массивов и хэшей.
- Некоторые свойства мы задаём вручную.
- Также, как и массивами, для вызова методов и свойств используем разделитель–точку.

Класс Triangle

Listing 3: Класс Triangle

```
class Triangle < Polygon
  def square
    pp = self.perimeter/2
    (pp*(pp - sides[0])*(pp-sides[1])*
    (pp-sides[2]))**0.5
  end
end
tr = Triangle.new
tr.sides = [3,4,5]
puts tr.square
puts tr.perimeter
```

Разбор класса Triangle

- Аналогично создаём класс Triangle, являющийся наследником класса Polygon.
- Для наследования используем конструкцию: Наследник < Родитель.
- Все методы и свойства класса Polygon автоматически появились в классе Triangle.
- Отдельно определяем по формуле Герона площадь треугольника.
- Итого, теперь в треугольнике мы можем посчитать и площадь, и периметр.

Разбор класса Triangle

- Разберёмся в конструкции self.perimeter.
- **self** означает текущий объект, то есть тот объект, для которого вызывается метод или свойство.
- **self.sides** — замена **@sides**.
- Однако вызвать метод со знаком @ не получится. Для этого и используем конструкцию self.
- self.perimeter вызывает метод perimeter для текущего объекта.
- **Задание**. В чём отличие записи @perimeter от self.perimeter? Одинаков ли будет результат. Если нет, приведите пример.

Задания

- Написать класс Прямоугольник — наследник Polygon. Определить в нём метод подсчёта площади. Проверить корректность его работы.
- Написать в классе Прямоугольник метод, определяющий, является ли прямоугольник квадратом. Метод должен возвращать булевский ответ. Проверить корректность работы метода.
- Создать в классе Треугольник метод, проверяющий, является ли данный треугольник прямоугольным. Проверить корректность работы метода.

References

- При подготовке данного материала использовались сайты: http://ru.wikibooks.org/wiki/Ruby, http://rubydev.ru, http://en.wikipedia.org, http://ruby-lang.org, http://de.trinixy.ru/, http://www.krassotkam.ru/, http://gen.su/.
- Все презентации доступны на http://school.smirik.ru!
- Вопросы, предложения, д/з: smirik@gmail.com