Объектно-ориентированное программирование в Java

Лекция 2

План лекции

- Введение
- Принципы ООП
- Этапы разработки программ
- ООП в Java

Введение

Объектно-ориентированное программирование (ООП) - это технология создания программного обеспечения, основанная на использовании совокупности программных объектов, каждый из которых является экземпляром определенного класса, при этом классы могут образовывать иерархию.

Класс можно определить как тип данных, который включает описание полей данных, а также методов (подпрограмм), работающих с этими полями данных.

Взаимодействие программных объектов в такой системе осуществляется с помощью передачи сообщений, реализуемой с помощью вызова процедур и функций.

Введение

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.

Введение

Преимущества объектно-ориентированного подхода:

- полное использование возможностей объектно-ориентированных языков программирования,
- унификация разработки и пригодность программ для повторного использования,
- простота внесения изменений,
- уменьшение риска неудачи при разработке сложных систем,
- ориентированность на человеческое восприятие мира.

Принципы ООП

- абстрагирование
- инкапсуляция
- модульность
- иерархичность
- типизация
- параллелизм
- сохраняемость

Принципы ООП

- ***Абстрагирование*** **–** принцип объектно-ориентированного программирования, который определяет правила выделения классов и объектов и построения их объектных моделей.
- Абстракция включает существенные для данной задачи характеристики некоторого объекта.
- В объектно-ориентированном программировании все свойства абстракции, определяющие состояние и поведение анализируемого объекта, объединяются в единую программную единицу - класс.
- Выделяют две части описания абстракции: интерфейс (доступные извне элементы абстракции - характеристики состояния и поведения) и реализация (недоступные извне элементы абстракции - внутренняя организация абстракции и механизмы осуществления ее поведения).

Принципы ООП

- ***Инкапсуляция*** **–** принцип объектно-ориентированного программирования, который определяет объект как единое целое и определяет интерфейс пользователя для использования объекта.

  Инкапсуляция **-** объединение интерфейса и реализации в один класс, а также скрытие деталей реализации.
- ***Модульность*** **-** принцип разработки программной системы, предполагающий ее реализацию в виде отдельных частей (модулей). Модули играют роль физических контейнеров, в которые помещаются описания классов и объектов.

Принципы ООП

- ***Иерархичность*** предполагает использование иерархий (упорядоченных систем абстракций) при разработке программных систем.
- В ООП используют два вида иерархии:
- «целое-часть»
- «общее-частное»

«целое-часть»- некоторые абстракции включены в другую абстракцию как ее части. Этот вид иерархии используется в процессе разбиения системы на части на разных этапах проектирования

Принципы ООП

«общее-частное» - некоторая абстракция является частным случаем другой абстракции. Данный вид иерархии используется при разработке структуры классов, когда сложные классы строятся на основе простых с помощью добавления к ним новых характеристик и изменения имеющихся. Этот механизм, являющийся в ООП одним из важнейших, называется ***наследованием***.

Принципы ООП

Наследование - принцип объектно-ориентированного программирования, который реализует отношение обобщения между классами.

Класс-потомок, наследующий характеристики другого класса, обладает теми же возможностями, что и класс-предок, от которого он порожден, при этом класс-предок остается без изменения, а классу-потомку можно добавлять новые элементы или изменять унаследованные. Благодаря этому потомок обладает большими возможностями, чем предок.

Все классы, используемые в Java, имеют общего предка - класс *java.lang.Object*.

Принципы ООП

- ***Типизация*** **-** ограничения, накладываемые на объект и препятствующее взаимозаменяемости объектов различных типов или уменьшающее возможность такой замены.

Принципы ООП

Тип может связываться с объектом во время компиляции (статическое или раннее связывание) или во время выполнения программы (динамическое или позднее связывание). Динамическое связывание, а также наличие наследования позволяют реализовать важный механизм ООП - *полиморфизм* (одно и то же имя может означать объекты разных типов, но, имея общего предка, все они имеют и общее подмножество операций, которые можно над ними выполнять).

Принципы ООП

- ***Полиморфизм*** – принцип объектно-ориентированого программирования, который позволяет за одним именем метода закреплять несколько различных алгоритмов и реализаций.

- ***Параллелизм*** - возможность нескольким абстракциям одновременно находиться в активном состоянии, выполняя некоторые операции.

Реальный параллелизм возможен только на многопроцессорных (многоядерных) системах, когда каждый процесс может выполняться отдельным процессором (ядром). На однопроцессорных (одноядерных) компьютерах параллелизм имитируется за счет механизма разделения времени между процессами (псевдопараллелизм).

Принципы ООП

- ***Сохраняемость*** - способность объекта существовать во времени независимо от породившего его процесса и/или в пространстве, возможно перемещаясь из своего первоначального адресного пространства.

На уровне программы по степени сохраняемости различают промежуточные результаты вычисления выражений, локальные переменные подпрограмм и глобальные объекты.

На уровне данных по степени сохраняемости выделяют данные, сохраняющиеся между сеансами выполнения программы, данные, сохраняемые при переходе на новую версию программы, а также данные, которые переживают программу.

Этапы разработки программы

- анализ
- проектирование
- реализация
- модификация

Этапы разработки программы

- Цель **первого** этапа - максимально полное описание задачи. В этот момент выполняют анализ предметной области, объектную декомпозицию и описывают абстракции. Результатом этапа является разработка диаграммы объектов, на которой показывают основные абстракции (объекты) предметной области и взаимодействие между ними.

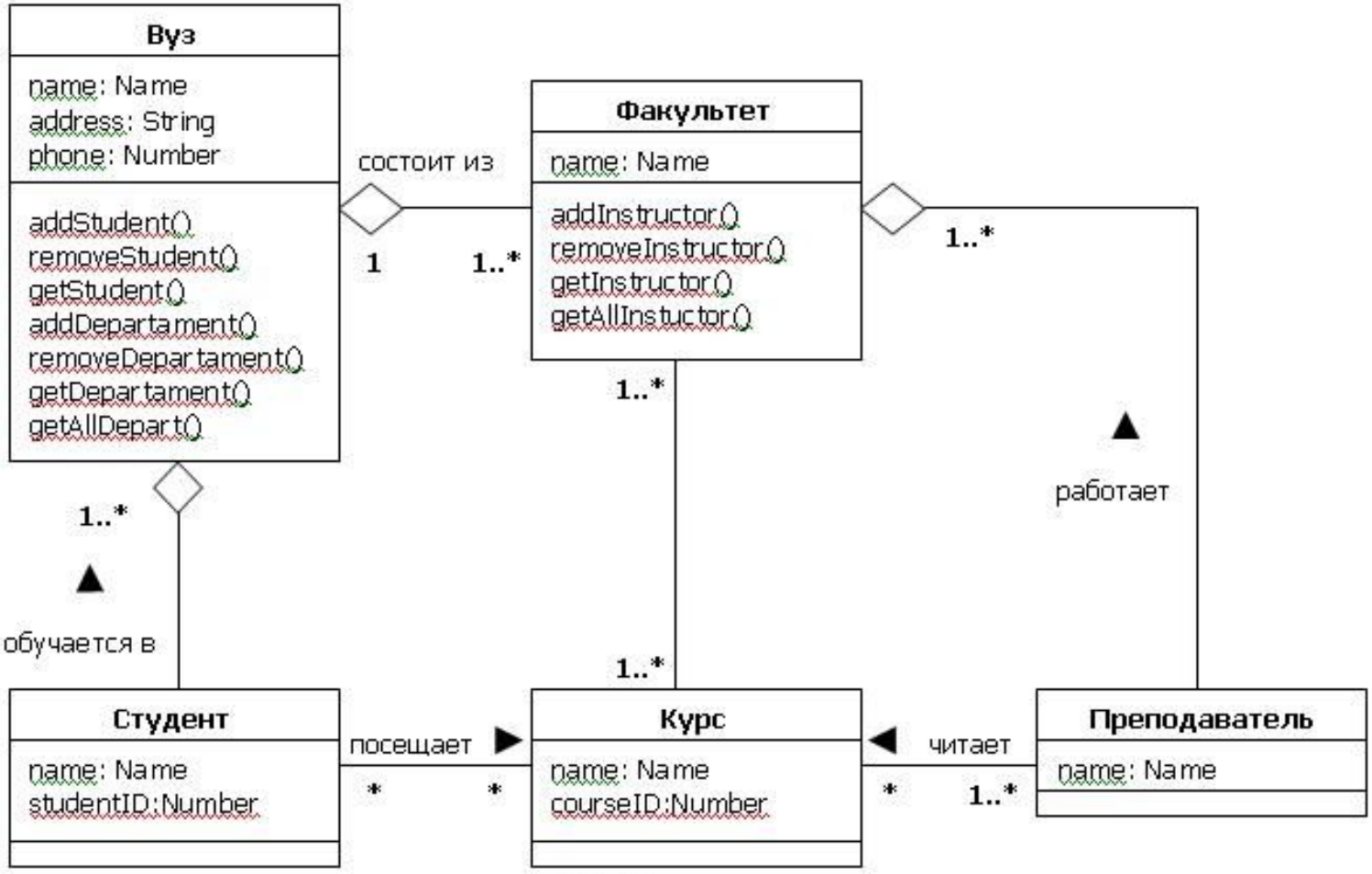
Этапы разработки программы

- **Проектирование** – это построение формализованного описания задачи, готового для реализации.

При этом выполняют логическое (разработка структуры классов) и физическое (объединение реализаций классов в модули, определение способов взаимодействия с операционной системой, син хронизация процессов при параллельной обработке и т. д.) проектирование.

Результат этапа проектирования - это диаграмма классов (иерархия и состав классов), а также другие диаграммы, описывающие задачу.

Этапы разработки программы



Этапы разработки программы

Цель этапа **реализации** - получение программной системы, выполняющей заданные функции.
Этап включает последовательную реализацию и подключение классов к проекту, создание работающего прототипа, а также постоянное тестирование и отладку.
Результатом этапа реализации является готовая программная система.

Цель этапа **модификации** - изменение существующей системы в соответствии с новыми требованиями.
Этот этап включает добавление новых элементов и модификацию существующих, при этом, как правило, меняется реализация, а не интерфейс.
Результатом модификации является измененная система.

ООП в Java

Объект — это мыслимая или реальная сущность, обладающая характерным поведением и отличительными характеристиками и являющаяся важной в предметной области (Гради Буч)

Объектно-ориентированное программирование — парадигма программирования, в которой программа строится из взаимодействующих объектов

ООП в Java

Свойства объектов

- Объект является экземпляром класса
- Объект имеет внутреннее состояние
- Объект может принимать сообщения (в большинстве языков сообщение = вызов метода)
- Объект — это «умные данные»

ООП в Java

ООП в Java:

- Поддержка ООП заложена в Java изначально (инкапсуляция, наследование, полиморфизм)
- В Java все является объектом, кроме примитивных типов
- Исполняемый код может находиться только в классе
- Стандартная библиотека предоставляет огромное количество классов, но можно свободно создавать свои

ООП в Java

Возможности ООП:

- Инкапсуляция

Сокрытие деталей реализации за внешним интерфейсом

- Наследование

Создание производных классов, наследующих свойства базового

- Полиморфизм

Разная обработка сообщений в разных классах

ООП в Java

Модификаторы можно разделить на две группы:
- модификаторы доступа,
- модификаторы свойств.

Модификаторы доступа:

- public - доступ для всех
- protected - доступ в пределах пакета и дочерних классов
- private - доступ в пределах класса по умолчанию (нет ключевого слова) доступ в пределах пакета

ООП в Java

Все составляющие класса называются его элементами или членами класса.

Обращение к отдельному элементу класса имеет синтаксис:

имя_объекта.имя_элемента.

ООП в Java

Вложенные классы:

- Можно объявить класс внутри другого класса
- Такие классы имеют доступ к private-членам друг друга
- Экземпляр вложенного класса связан с экземпляром внешнего класса
- Если связь не нужна, вложенный класс объявляют с модификатором static

ООП в Java

Методы:

```
class Example {
      private int number ;
      /* modifiers */ int getNumber () {
            return number ;
      }
}
```

ООП в Java

Имя метода в классе может быть неуникальным.

Существование в классе методов с одинаковым именем называется перегрузкой, а сами одноименные методы называются **перегруженными.**

Перегрузка методов полезна, когда требуется решать однотипные задачи с разным набором параметров.

Перегруженные методы, имея одинаковое имя, должны обязательно отличаться либо числом параметров, либо их типами.

ООП в Java

Например, один класс может содержать такие перегруженные методы, позволяющие определять максимум:

//максимум двух целых чисел int max(int a, int b);

//максимум трех целых чисел int max(int a, int b, int c);

//максимум целого числа и длины строки int max(int a, string b);

//максимум чисел в строковом представлении int max(string a, string b);

ООП в Java

Существует несколько типов специальных методов:

- конструкторы,
- деструкторы,
- методы-свойства,
- индексаторы,
- метод Main().

ООП в Java

Конструктор - обязательный элемент любого класса, предназначенный для инициализации объекта, он вызывается автоматически при создании объекта класса с помощью операции *new.*
Имя конструктора всегда совпадает с именем класса.
Хотя конструктор и является методом, он не возвращает значения (даже *void)*
Если в классе явно не задан ни один конструктор, то к классу автоматически добавляется конструктор по умолчанию - конструктор, вызываемый без параметров.
Класс может иметь несколько конструкторов с разными параметрами (это называется перегрузкой конструкторов).

ООП в Java

Деструктор:

- В Java нет деструкторов, сбор мусора автоматический
- Есть метод void finalize(), но пользоваться им не рекомендуется
- При необходимости освободить ресурсы заводят обычный метод void close() или void dispose()

ООП в Java

Методы-свойства (Getters и Setters):

- Для переменных класса модификатор доступа **public** нежелателен т.к. он нарушает безопасность объекта.
- Для защиты был придуман механизм доступа к переменным класса с помощью **get**и **set** (геттер и сеттер).
- **Get** позволяет получить значения (читать значения)
- **Set** - записать значения в переменную.
- В коде они представляют собой обычные методы. Но имя метода всегда начинается с префикса **get** или **set**.

Спасибо за внимание!