Принципы ООП

Абстрагирование полиморфизм интерфейсы

Абстрагирование, Абстракция

- **Абстрагирование** – это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые.
- **абстракция** – это набор всех таких характеристик.
- Абстракция является основой ООП и позволяет работать с объектами, не вдаваясь в особенности их реализации

Абстрагирование

- ***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).

наследование

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.*

полиморфизм

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

interface

- Интерфейсы определяют некоторый функционал, не имеющий конкретной реализации, который затем реализуют классы, применяющие эти интерфейсы.
- И один класс может применить множество интерфейсов.
- Чтобы определить интерфейс, используется ключевое слово interface.

интерфейс

- Интерфейс определяет возможные сообщения, но не их реализацию

```
interface Interface1 {
        int getNumber ();
}
```

- Класс может реализовывать несколько интерфейсов

```
class Example implements Interface1 , Interface2 {
        int getNumber () {
                // implementation
        }
        // other methods
}
```

методы интерфейса

- Все методы интерфейса не имеют модификаторов доступа, но фактически по умолчанию доступ public, так как цель интерфейса - определение функционала для реализации его классом.
- Поэтому весь функционал должен быть открыт для реализации.
- Чтобы класс применил интерфейс, надо использовать ключевое слово implements:

параллелизм

- ***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).

перечисления

Перечисления:

- Класс с фиксированным количеством экземпляров
- Может иметь поля и методы

```
enum Direction {
        LEFT ,
        RIGHT ,
        UP ,
        DOWN
}
```

Этапы разработки программ

- анализ
- проектирование
- реализация
- модификация

Этап: анализ

- 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.

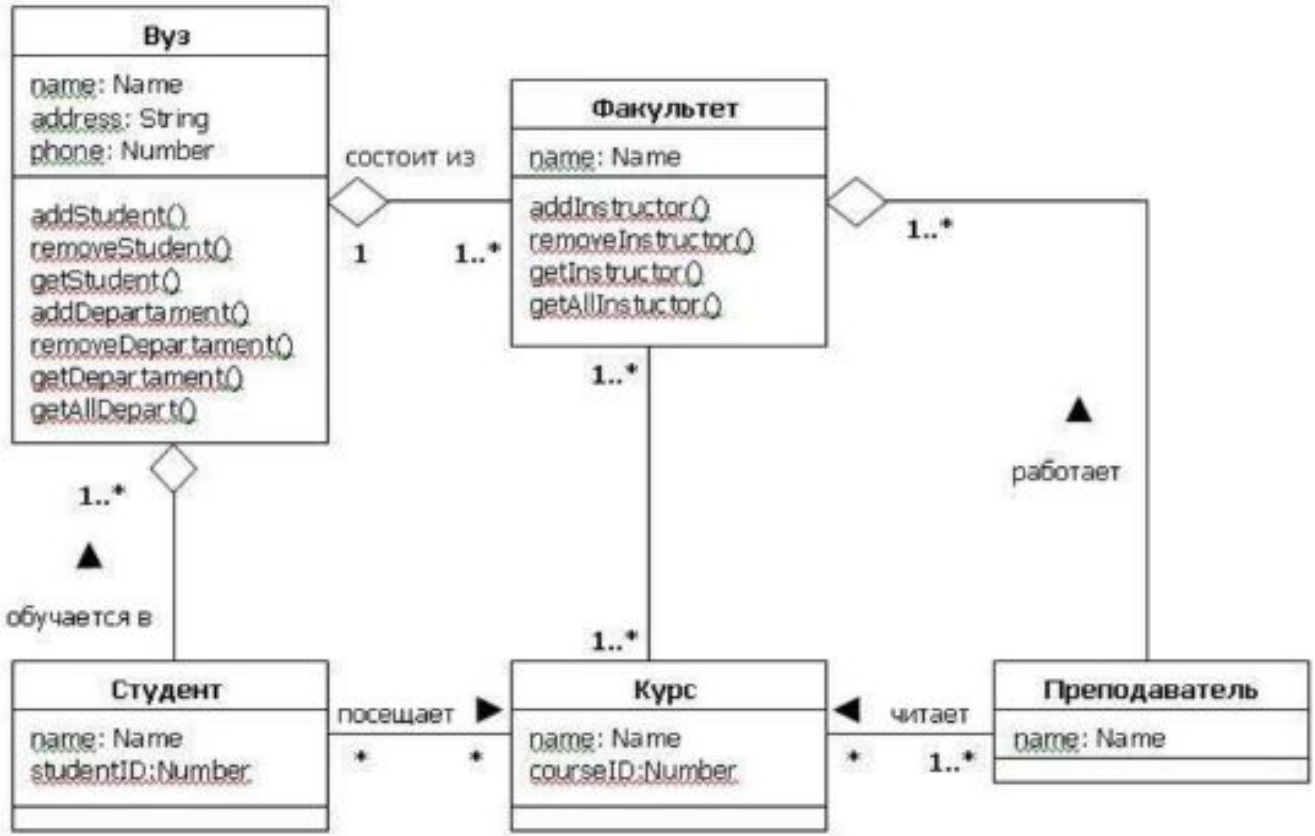
Целью данного этапа является разработка диаграммы объектов, на которой показывают оснавные абстракции(объекты) и взаимодействие между ними

Этап: проектирование

- **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, синхронизация процессов при параллельной обработке и т. д.) проектирование.

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


Вуз
name: Name
address: String
phone: Number
addStudent()
removeStudent()
getStudent()
addDepartament()
removeDepartament()
getDepartament()
getAllDepart()
состоит из
1
1..*
Факультет
name: Name
addInstructor()
removeInstructor()
getInstructor()
getAllInstuctor()
1..*
1..*
работает
1..*
обучается в
Студент
name: Name
studentID:Number
посещает
*
*
1..*
Курс
name: Name
courseID:Number
читает
*
1..*
Преподаватель
name: Name

Этап:Реализация

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.

Структура курсовой работы

Введение

- анализ
- проектирование
- реализация
- модификация

заключение

Список использованных источников

Приложение А
Приложение Б

Требования к оформлению

- Объем не менее 15 страниц
- Без воды
- Поля: 3/2/2/2
- Межстрочный интервал: одинарный
- Рисунки, схемы, диаграммы, скриншоты только собственного производства