Изучаемый курс

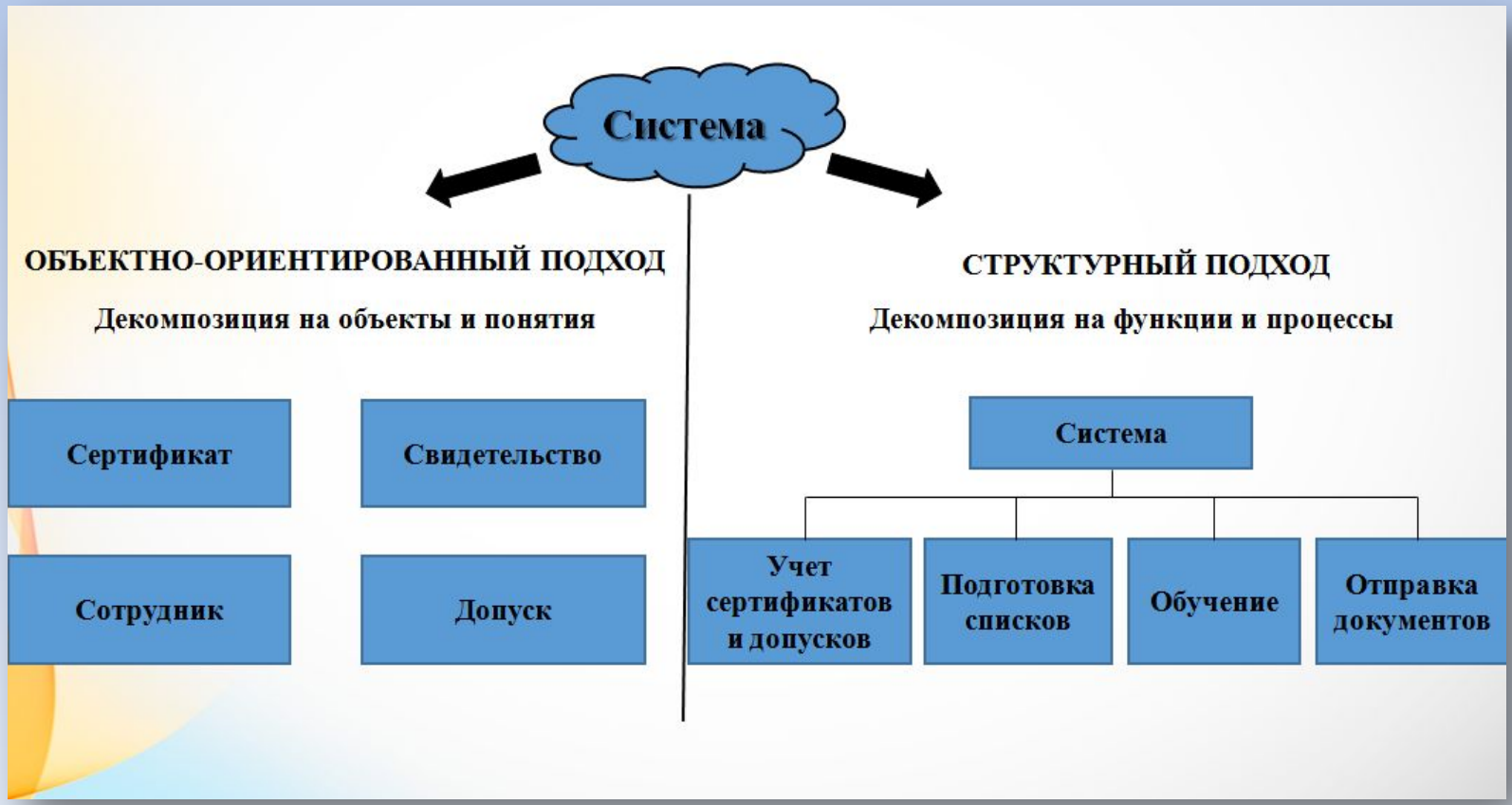
«ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Преподаватель дисциплины:
Нефедова Людмила Петровна

Специальность:
09.02.07 информационные системы и программирование

Занятие 8.

Основные принципы построения объектной модели.



Объектно-ориентированные методы анализа и проектирования ПО

- **Различие моделей** в декомпозиции
- В основе объектно-ориентированного подхода ООП лежит **объектная декомпозиция**, при этом структура системы описывается в терминах ***объектов*** и ***связей*** между ними, а поведение системы описывается в терминах обмена сообщениями между ними.
- Каждый объект системы обладает **собственным поведением**, моделирующим **поведение объекта реального мира**

* Определение ООП и его основные концепции

- В центре ООП находится понятие ***объекта***. **Объект** — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. **Объект** — это **экземпляр класса**. Данные объекта скрыты от остальной программы. **Инкапсуляция** включает в себя **сокрытие** (Но им не является!).
- Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие **наследования**.
- Но даже наличие **инкапсуляции и наследования** не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован **полиморфизм подтипов** — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

* Парадигма объектно-ориентированного программирования

Объе́ктно-ориенти́рованное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности взаимодействующих **объектов**, каждый из которых является экземпляром определённого **класса**, а классы образуют **иерархию наследования.**

Основные принципы структурирования предметной задачи в случае применения ООП для оптимального управления соответствующей моделью:

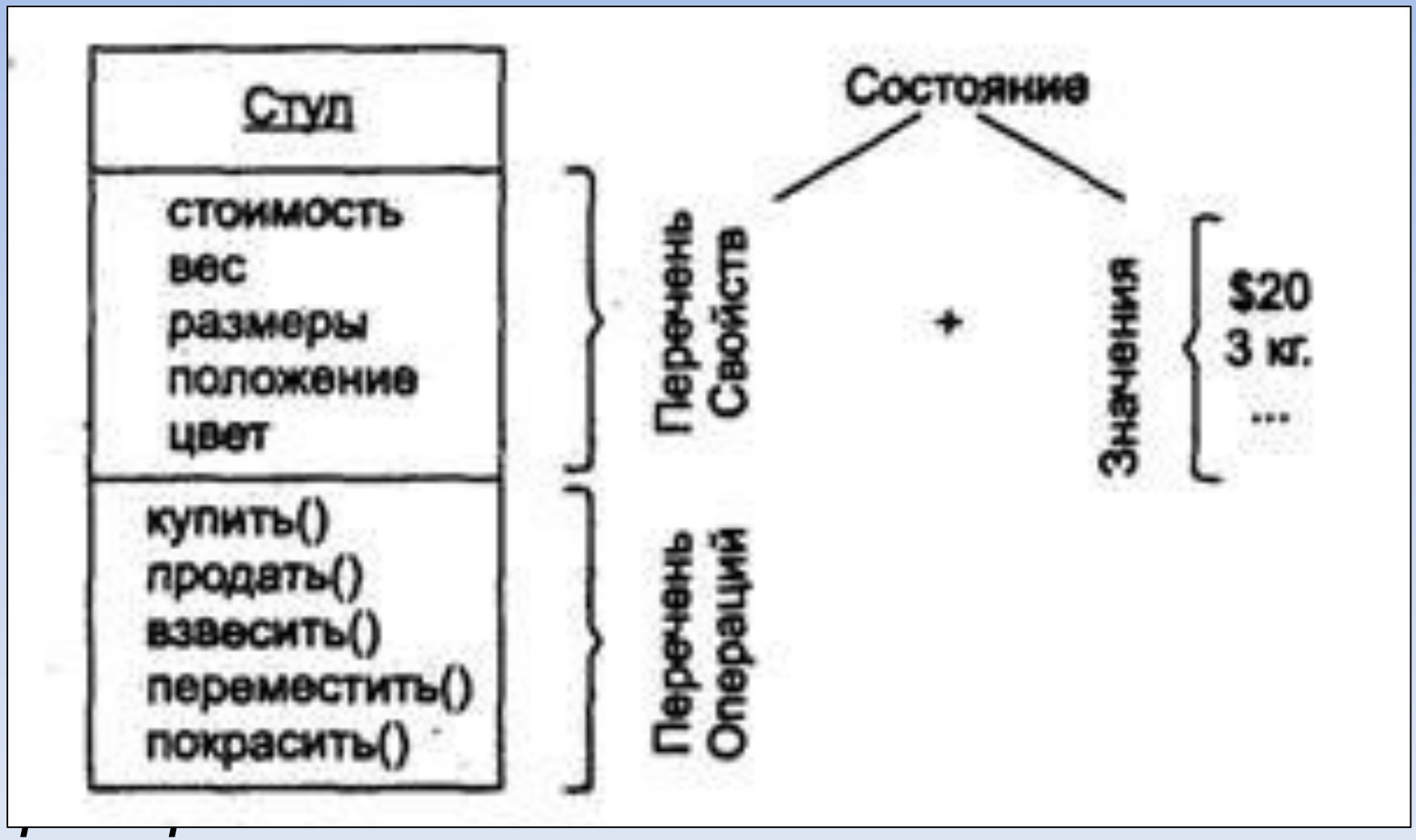
- **Абстрагирование** (упрощает представление физического объекта);
- **Инкапсуляция** (закрывает детали внутреннего представления абстракций)
- **Наследование** (позволяет многократно использовать общие группы в других абстракциях)
- **Полиморфизм** (возможность работы с объектами разных типов, каждый из которых поддерживает данный набор интерфейсов, но реализует их по-разному)

* 1. Абстрагирование

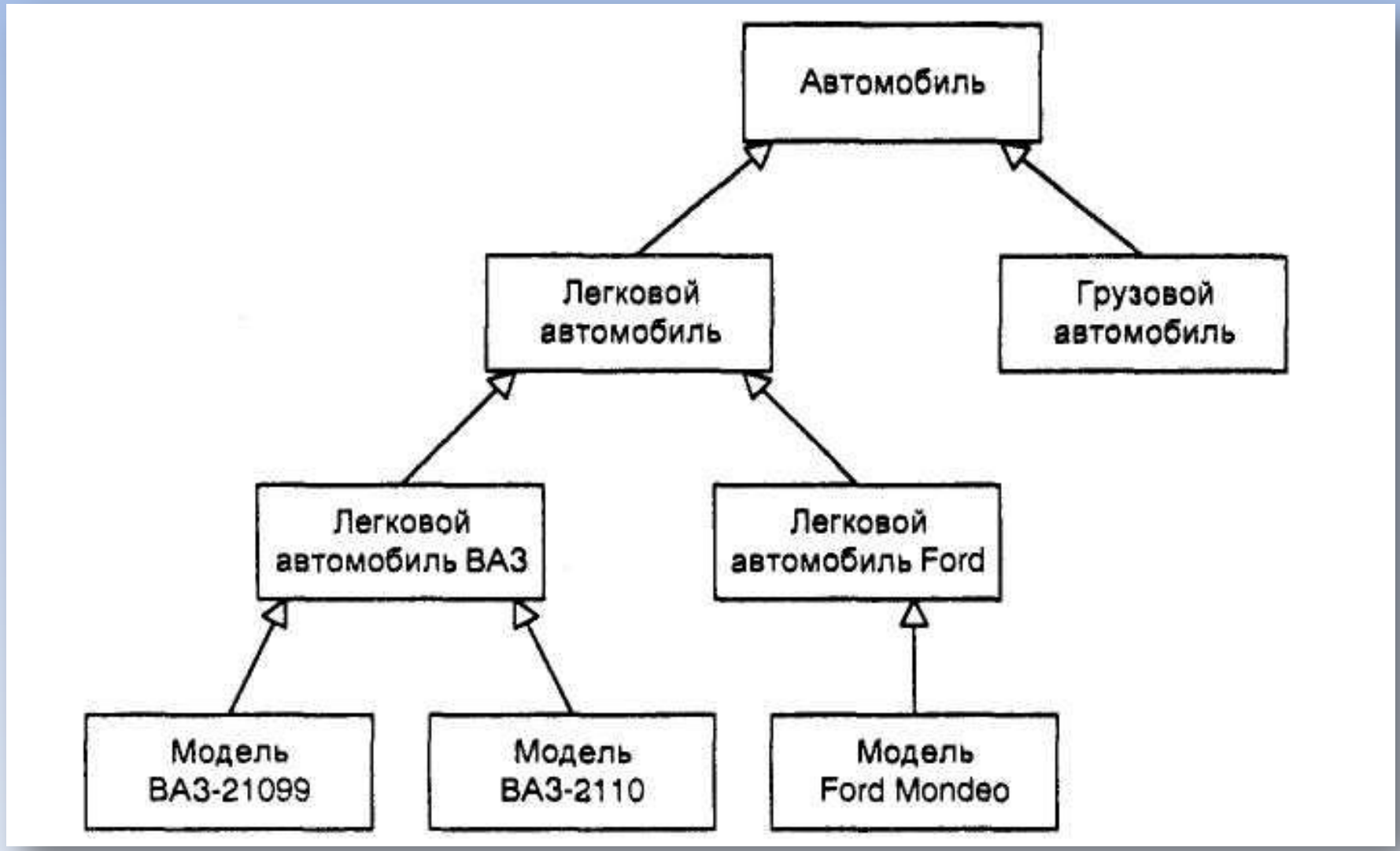
Пример:

Водитель едет в автомобиле по оживлённому участку движения и не задумывается о химическом составе краски автомобиля, особенностях взаимодействия шестерён в коробке передач или влияния формы кузова на скорость. Однако, руль, педали, указатель поворота он будет использовать регулярно.

* 2. Инкапсуляция



* 3. Наследование



* 4. Полиморфизм

Пример:

Основные элементы управления автомобиля имеют одну и ту же конструкцию и принцип действия. Если человеку надо доехать с работы домой, то он сядет за руль автомобиля и будет выполнять одни и те же действия, независимо от того, какой именно тип автомобиля он использует. Можно сказать, что все автомобили имеют один и тот же интерфейс.

5. Класс. + 6. Объект

- **Класс** — универсальный, комплексный тип данных, состоящий из тематически единого набора «**полей**» (переменных более элементарных типов) и «**методов**» (функций для работы с этими полями), то есть он является моделью информационной сущности с внутренним и внешним интерфейсами для оперирования своим содержимым (значениями полей).
- **Объект** это эквивалент класса. Сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса (например, после запуска результатов компиляции и связывания исходного кода на выполнение).
- Переменная-объект, относящаяся к заданному классом типу, называется **экземпляром** этого

Термины ООАП

- **Объект** – некоторая сущность, обладающая состоянием и поведением(свойства и методы)
- **Класс** – совокупность объектов, связанных общностью структуры и поведением. Поведение – воздействие на другие объекты, воздействия на него со стороны других и передачи сообщений.
- **Операция** – воздействие одного объекта на другой в целях вызвать реакцию(методы)
- **Наследование**
- **Инкапсуляция**
- **Полиморфизм**

Основные принципы объектной модели

- *Объектная модель* – концептуальная основа ОО подхода
- Основные принципы:
  - ✔ **Абстрагирование** – выделение наиболее важных, существенных характеристик объекта, отличающих от других
  - ✔ **Инкапсуляция** – локализация свойств и поведения в рамках единственной абстракции
  - ✔ **Модульность** – возможность декомпозиции на внутренние сильно сцепленные, слабо связанные подсистемы (модули)
  - ✔ **Иерархия** - упорядоченная система абстракций, расположение по уровням.

Язык моделирования

Большинство существующих методов объектно-ориентированного анализа и проектирования (ООАП) включают **язык моделирования** и **описание процесса моделирования**.

- **Процесс моделирования** – описание шагов, выполняемых при моделировании.
- **Язык моделирования** — это язык графического описания, нотация (в основном графическая), которая используется для описания проектов.
- **Нотация** представляет собой совокупность графических объектов, которые используются в моделях; она является *синтаксисом языка моделирования.*
- **Процесс** — это описание шагов, которые необходимо выполнить при разработке проекта

Универсальный язык моделирования UML – открытый стандарт

- ***UML*** – ***Unified Modeling Language*** – **унифицированный язык моделирования**, предназначен для **визуализации и документирования объектно-ориентированных систем и бизнес-процессов** для последующей реализации в виде программного обеспечения.
- Содержит **стандартный набор диаграмм моделирования**
- ***UML*** - единый универсальный стандарт для объектно-ориентированного моделирования (1996-2005 гг)

Диаграммы

Диаграммы визуализируют структурные компоненты системы, их взаимосвязи, а также поведение системы в пространстве и с течением времени. Можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

Язык UML включает **13 видов диаграмм**, среди которых на первом месте в списке — **диаграмма классов**.

Общие принципы языка UML

- Абстрагирование
- Многомодельность
- Иерархия

Словарь UML включает три вида строительных блоков:

- **Диаграммы.**
- **Сущности.**
- **Связи(отношения)**.

- **Сущности** – это абстракции, которые являются основными элементами модели, связи соединяют их между собой, а диаграммы группируют представляющие интерес наборы сущностей.
- **Диаграмма** – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей).
- **Отношения(связи)** связывают различные сущности


Основные диаграммы UML (нотация 2.0):
Статические диаграммы:
Диаграмма классов;
Диаграмма объектов;
Диаграмма компонентов
И др.
Динамические диаграммы
Поведенческие диаграммы:
Диаграмма прецедентов;
Диаграмма последовательностей
Диаграмма состояний;
Диаграмма активности;
Диаграмма развертывания и т.д.
Физические диаграммы:
Диаграмма реализации.

Модели UML . Виды

- Структурные(статические)
- МоСтруктурные(статические)

Модели UML . Уровни

- Концептуальные (верхний уровень, обобщенныйю. Например: Диаграмма вариантов)
- Логические модели(второй уровень, не имеют физического воплощения, отражают логиеские аспекты структуры, поведенческие. Например: Диаграммы классов, состояний, деятельности, последовательности, кооперации)
- Физические (нижний уровень описания, конкретные материальные сущности. Например: Диаграммы компонентов, развертывания)

Статические диаграммы представляют либо постоянно присутствующие в *системе сущности и связи между ними*, либо *сущности и связи*, существующие в какой-то определенный момент времени. Они не показывают способов поведения этих сущностей.

К этому типу относятся *диаграммы классов, объектов, компонентов и диаграммы развертывания*.

Динамические диаграммы описывают *происходящие в системе процессы*.

К ним относятся *диаграмма прецедентов; диаграмма последовательностей; диаграмма состояний; диаграмма активности; диаграмма развертывания* и т.д.

Логическое представление системы, описывает, как *система должна быть построена;*

Представление реализации, описывает *зависимость между программными компонентами*;

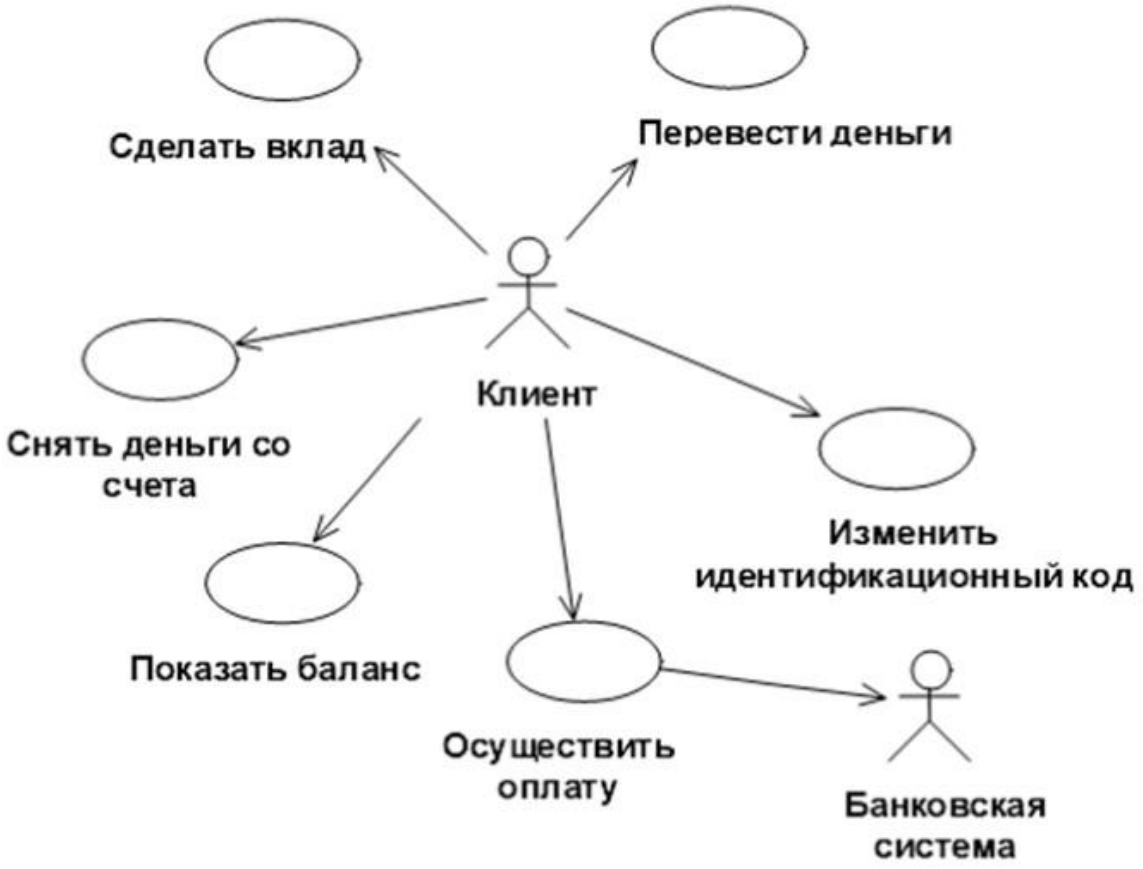
Представление развертывания, описывает *аппаратные элементы, устройства и программные компоненты*

Визуальное моделирование в UML представляет собой *процесс* ***последовательного поуровневого спуска*** *от наиболее общей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели разрабатываемой программной системы*.

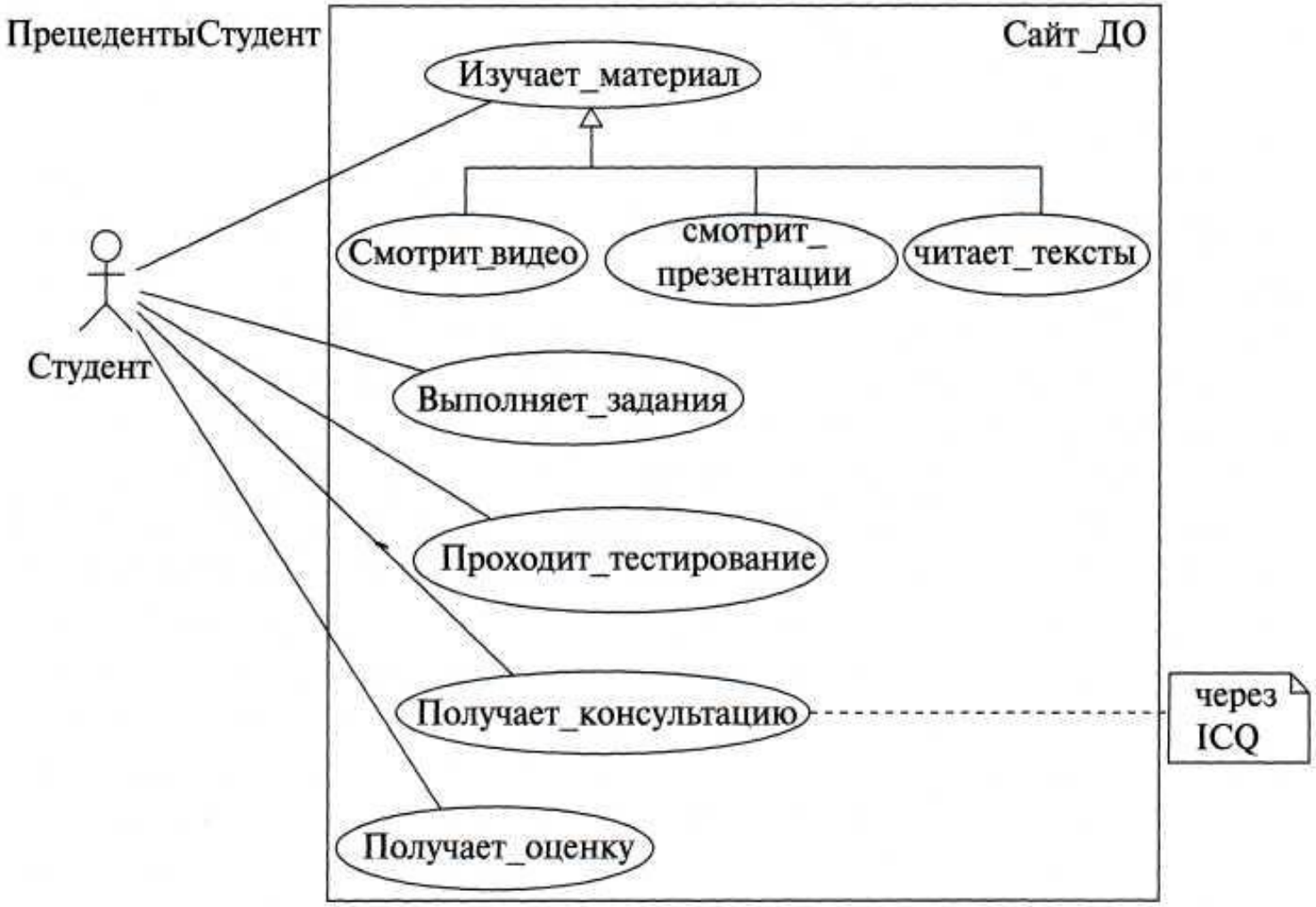
Первой строится модель ***Диаграммы вариантов использования*** *(use case diagram*), которая описывает функциональное назначение системы или то, что система будет делать в процессе своего функционирования.

Диаграмма вариантов использования является *базой при разработке функциональных требований.* Разработку начинают с анализа требований к функциональности, согласно ТЗ. Выявляются внешние пользователи, аспекты поведения во взаимодействии с пользователями. Аспекты поведения называются **вариантами**

Диаграмма вариантов использования (use case diagram)

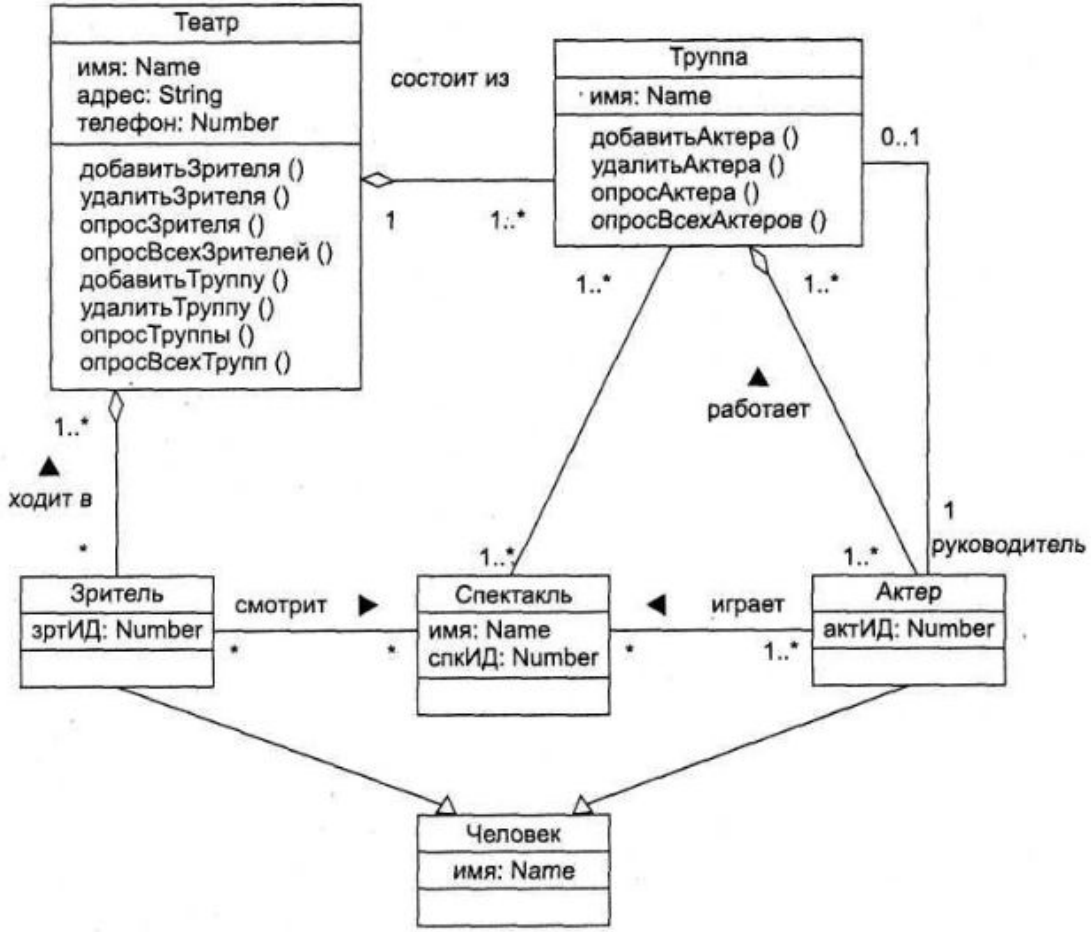


Пример


ПрецедентыСтудент
Сайт_ДО
Изучает_материал
Смотрит_видео
смотрит_
презентации
читает_тексты
Студент
Выполняет_задания
Проходит_тестирование
Получает_консультацию
через
ICQ
Получает_оценку

1. Диаграмма классов (Class Diagram)

- **показывает набор классов и их отношений, обеспечивая статическое представление ПС.**



2. Диаграмма объектов (Object Diagram)-

- **набор объектов и их отношений, показывает статический «моментальный снимок» с экземпляров предметов.**

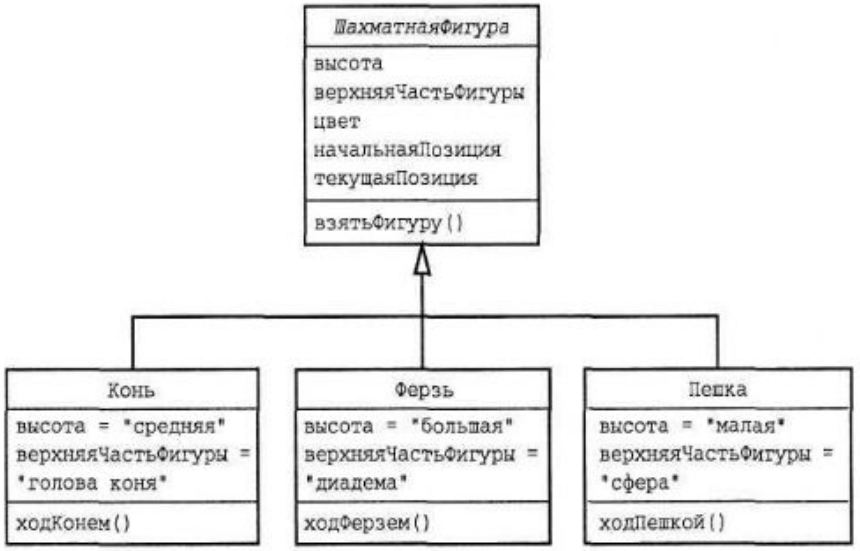


Рис. 4.18. Диаграмма классов для фрагмента игры в шахматы

3. Диаграмма прецедентов или вариантов использования (Use Case Diagram)

- **показывает набор актёров, вариантов использования (Use Case) системы и их отношения.**

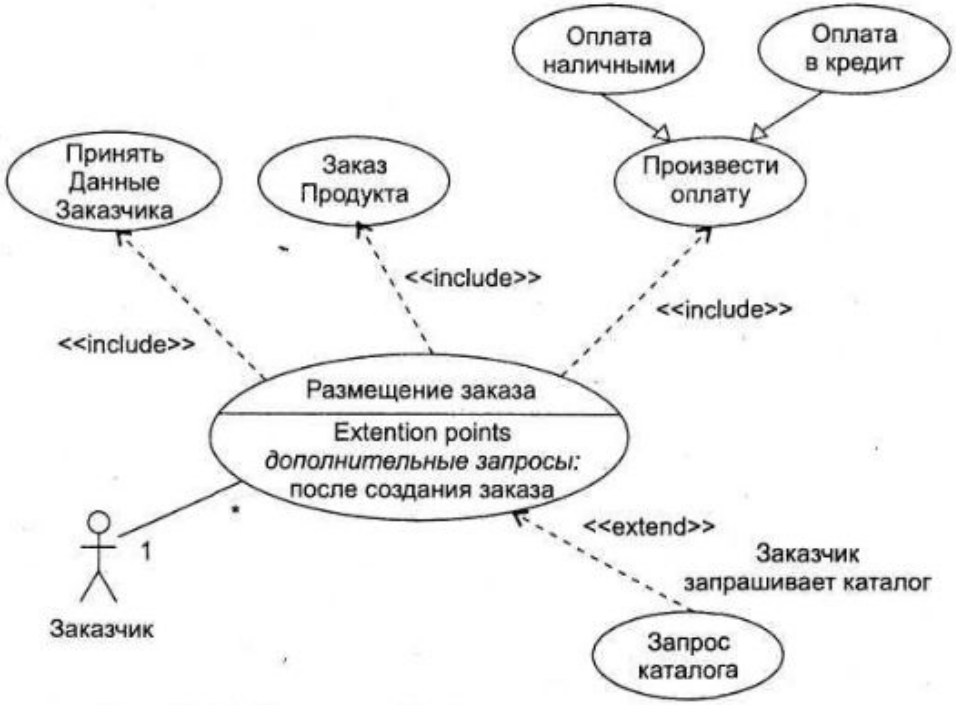
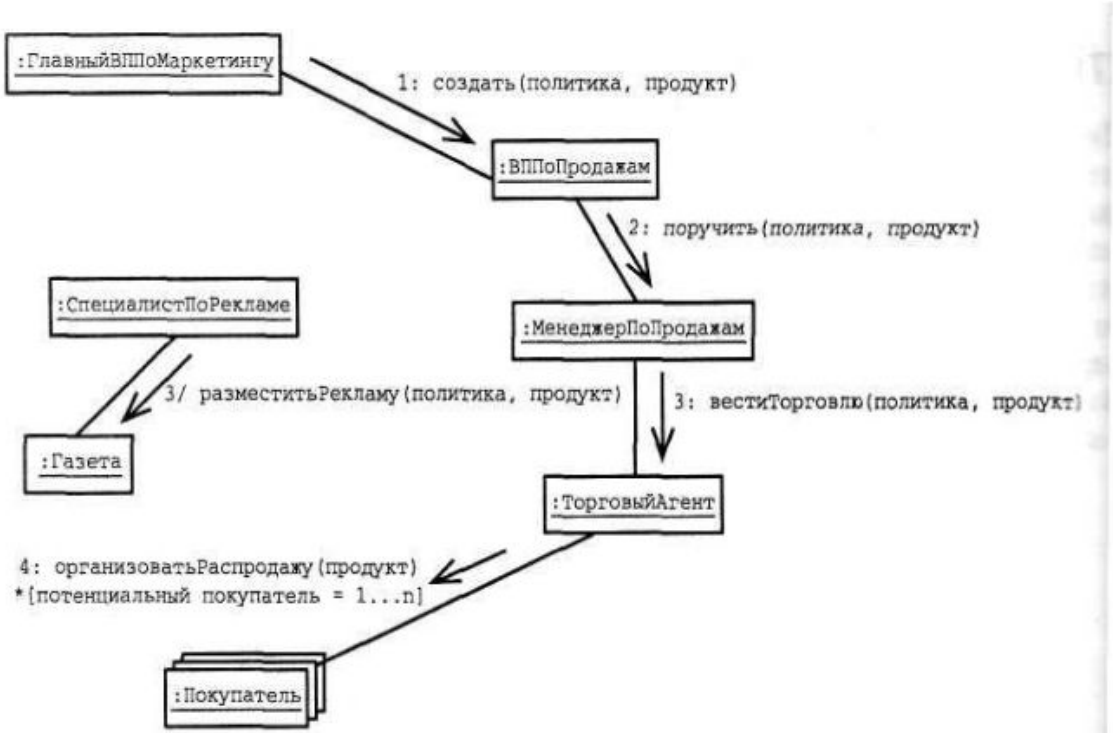


Рис. 12.34. Диаграмма Use Case для обслуживания заказчика

- ***Диаграмма Use Case строится для отражения функциональных требований к ПС.***

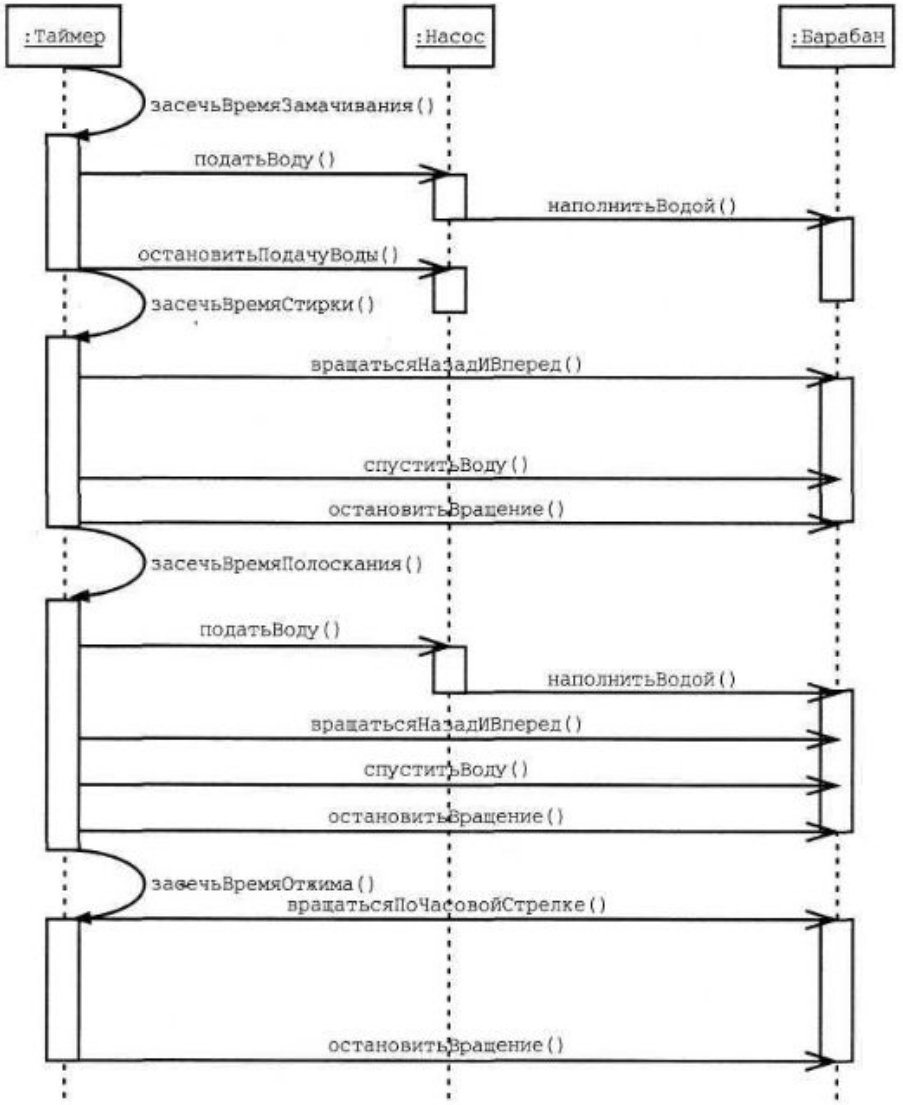
4. Диаграмма взаимодействия (Communication Diagram)

- **показывает взаимодействия, включающие набор объектов, их отношений, а также пересылаемые между ними сообщения.**



5. Диаграмма последовательности (Sequence Diagram) -

- **диаграмма взаимодействия, которая выделяет упорядочение сообщений во времени.**



6. Диаграмма деятельности (Activity Diagram)

- **показывает поток от действия к действию внутри системы.**

Источники

- http://www.kgau.ru/istiki/umk/mbp/pt04.html
- https://prog-cpp.ru/uml-classes/