Дисциплина: «**МДК 01.01. Технология разработки программного обеспечения**»

Лабораторная работа «Построение диаграммы состояний и диаграммы классов»

Преподаватель спец. дисциплин Радунцева Александра Антоновна

Цели лабораторной работы

- Изучить теорию построения диаграмм состояний
- Научиться анализировать и описывать диаграммы состояний
- Научиться строить диаграммы состояний

Диаграммы состояний

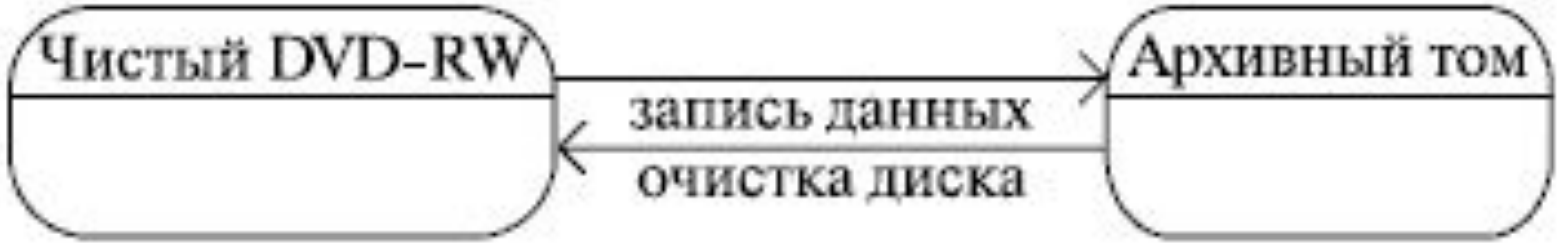
- Объекты характеризуются поведением и состоянием, в котором находятся. Например, человек может быть новорожденным, младенцем, ребенком, подростком или взрослым. Другими словами, объекты что-то делают и что-то "знают". Диаграммы состояний применяются для того, чтобы объяснить, каким образом работают сложные объекты.
- Состояние (state) - ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, выполняет определенную деятельность или ожидает какого-то события. Состояние объекта определяется значениями некоторых его атрибутов и присутствием или отсутствием связей с другими объектами.

Диаграммы состояний

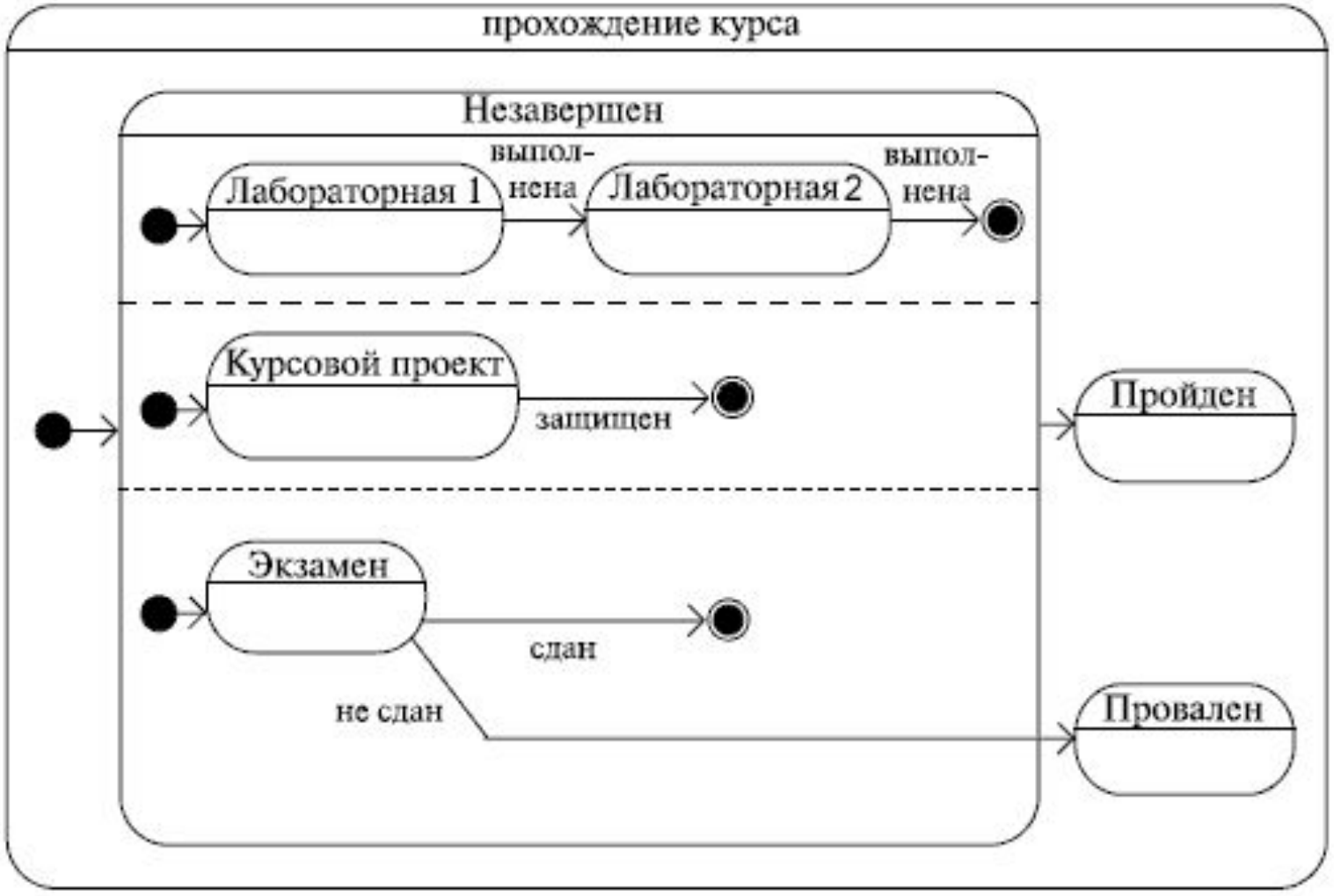
- Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Очевидно, что диаграммы состояний служат для моделирования динамических аспектов системы (как и диаграммы последовательностей, кооперации, прецедентов, деятельности). Диаграмма состояний полезна при моделировании жизненного цикла объекта (как и ее частная разновидность - диаграмма деятельности).
- От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события. Понятие жизненного цикла применимо как раз к реактивным объектам, настоящее состояние (и поведение) которых обусловлено их прошлым состоянием. Но диаграммы состояний важны не только для описания динамики отдельного объекта. Они могут использоваться для конструирования исполняемых систем путем прямого и обратного проектирования.

Диаграммы состояний

- Скругленные прямоугольники представляют состояния, через которые проходит объект в течение своего жизненного цикла. Стрелками показываются переходы между состояниями, которые вызваны выполнением методов описываемого диаграммой объекта. Существует также два вида псевдосостояний: начальное, в котором находится объект сразу после его создания (обозначается сплошным кружком), и конечное, которое объект не может покинуть, если перешел в него (обозначается кружком, обведенным окружностью).
- Пример простейшей диаграммы состояний.

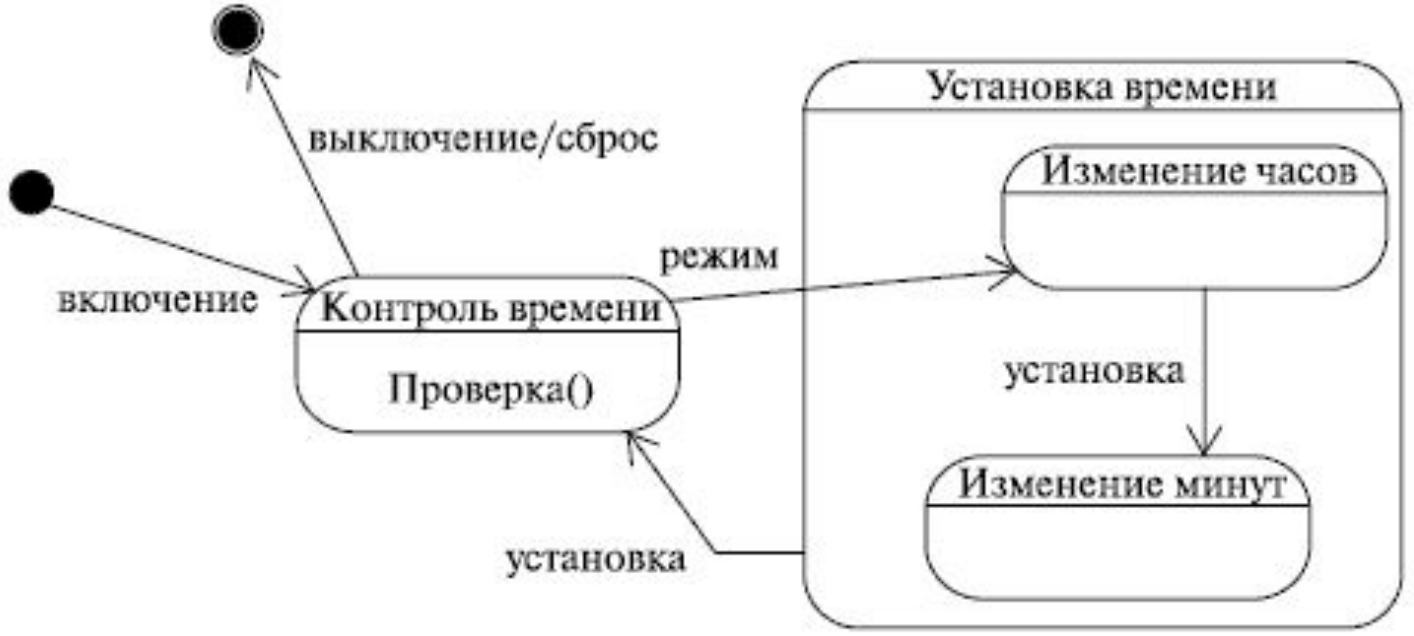


Диаграммы состояний



- Здесь составное состояние, включающее другие состояния, одно из которых содержит также параллельные подсостояния.
- Это диаграмма прохождения академического курса студентом.
- Для того чтобы пройти курс, студент должен выполнить лабораторные работы, защитить курсовой проект и сдать экзамен.

Диаграммы состояний



- Устройство – таймер. Такой прибор может применяться в составе различных реле, например, для отключения телевизора по истечении указанного промежутка времени.
- Основное его назначение - контроль времени (проверка, не истек ли указанный промежуток), но у него есть еще один режим работы - установка. По истечении указанного промежутка времени или при "сбросе" устройство отключается.

Диаграмма классов

- Диаграммы классов показывают набор классов, интерфейсов, а также их связи. Диаграммы этого вида чаще всего используются для моделирования объектно-ориентированных систем. Они предназначены для статического представления системы.
- Большинство элементов UML имеют уникальную и прямую графическую нотацию, которая дает визуальное представление наиболее важных аспектов элемента.

Сущности

- Диаграммы классов оперируют тремя видами сущностей UML:
- Структурные.
- Поведенческие.
- Аннотирующие.
- **Структурные сущности** – это «имена существительные» в модели UML. В основном, статические части модели, представляющие либо концептуальные, либо физические элементы. Основным видом структурной сущности в диаграммах классов является класс.

Диаграмма классов

- **Поведенческие сущности** – динамические части моделей UML. Это «глаголы» моделей, представляющие поведение модели во времени и пространстве. Основной из них является взаимодействие – поведение, которое заключается в обмене сообщениями между наборами объектов или ролей в определенном контексте для достижения некоторой цели. Сообщение изображается в виде линии со стрелкой, почти всегда сопровождаемой именем операции.

- **Аннотирующие сущности** – это поясняющие части UML-моделей, иными словами, комментарии, которые можно применить для описания, выделения и пояснения любого элемента модели. Главная из аннотирующих сущностей – примечание. Это символ, служащий для описания ограничений и комментариев, относящихся к элементу либо набору элементов. Графически представлен прямоугольником с загнутым углом; внутри помещается текстовый или графический комментарий.

Диаграмма классов

- **Класс** – это описание набора объектов с одинаковыми атрибутами, операциями, связями и семантикой.
- Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями:
  - имя класса
  - атрибуты (свойства) класса
  - операции (методы) класса.
- Для атрибутов и операций может быть указан один из трех типов видимости:
  - — private (частный)
  - # protected (защищенный)
  - + public (общий)
- Видимость для полей и методов указывается в виде левого символа в строке с именем соответствующего элемента.

Диаграмма классов

- Каждый класс должен обладать именем, отличающим его от других классов. **Имя** – это текстовая строка. Имя класса может состоять из любого числа букв, цифр и знаков препинания (за исключением двоеточия и точки) и может записываться в несколько строк.
- На практике обычно используются краткие имена классов, взятые из словаря моделируемой системы. Каждое слово в имени класса традиционно пишут с заглавной буквы (верблюжья конвенция), например Sensor (Датчик) или TemperatureSensor (ДатчикТемпературы).
- Для абстрактного класса имя класса записывается курсивом.

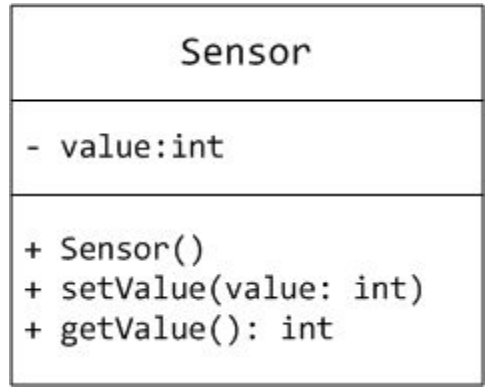


Диаграмма классов

- **Атрибут** (свойство) – это именованное свойство класса, описывающее диапазон значений, которые может принимать экземпляр атрибута. Класс может иметь любое число атрибутов или не иметь ни одного. В последнем случае блок атрибутов оставляют пустым.
- Атрибут представляет некоторое свойство моделируемой сущности, которым обладают все объекты данного класса. Имя атрибута, как и имя класса, может представлять собой текст. На практике для именования атрибута используются одно или несколько коротких существительных, выражающих некое свойство класса, к которому относится атрибут.
- Можно уточнить спецификацию атрибута, указав его тип, кратность (если атрибут представляет собой массив некоторых значений) и начальное значение по умолчанию.
- Статические атрибуты класса обозначаются подчеркиванием.

Диаграмма классов

- **Операция** (метод) – это реализация метода класса. Класс может иметь любое число операций либо не иметь ни одной. Часто вызов операции объекта изменяет его атрибуты.
- Графически операции представлены в нижнем блоке описания класса.
- Допускается указание только имен операций. Имя операции, как и имя класса, должно представлять собой текст. На практике для именования операции используются короткие глагольные конструкции, описывающие некое поведение класса, которому принадлежит операция. Обычно каждое слово в имени операции пишется с заглавной буквы, за исключением первого, например `move` (переместить) или `isEmpty` (проверка на пустоту).
- Можно специфицировать операцию, устанавливая ее сигнатуру, включающую имя, тип и значение по умолчанию всех параметров, а применительно к функциям – тип возвращаемого значения.
- Абстрактные методы класса обозначаются курсивным шрифтом.
- Статические методы класса обозначаются подчеркиванием.

Диаграмма классов

- Изображая класс, не обязательно показывать сразу все его атрибуты и операции. Для конкретного представления, как правило, существенна только часть атрибутов и операций класса. В силу этих причин допускается упрощенное представление класса, то есть для графического представления выбираются только некоторые из его атрибутов. Если помимо указанных существуют другие атрибуты и операции, вы даете это понять, завершая каждый список многоточием.

- Чтобы легче воспринимать длинные списки атрибутов и операций, желательно снабдить префиксом (именем стереотипа) каждую категорию в них. В данном случае **стереотип** – это слово, заключенное в угловые кавычки, которое указывает то, что за ним следует.

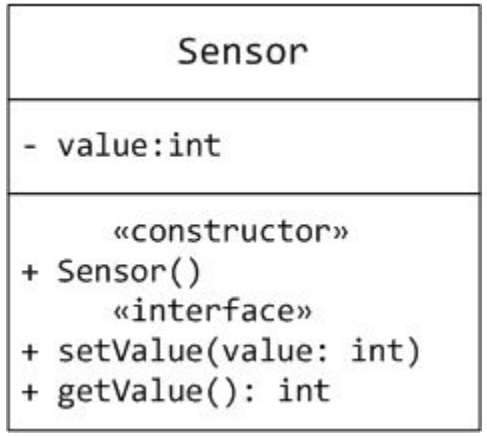


Диаграмма классов

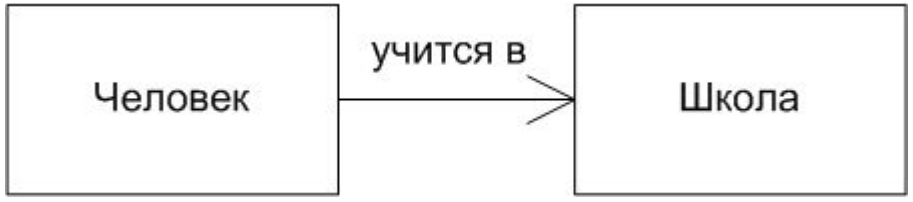
Отношения между классами

- Существует четыре типа связей в UML:
  - Зависимость
  - Ассоциация
  - Обобщение
  - Реализация
- Эти связи представляют собой базовые строительные блоки для описания отношений в UML, используемые для разработки хорошо согласованных моделей.
- Первая из них – **зависимость** – семантически представляет собой связь между двумя элементами модели, в которой изменение одного элемента (независимого) может привести к изменению семантики другого элемента (зависимого). Графически представлена пунктирной линией, иногда со стрелкой, направленной к той сущности, от которой зависит еще одна; может быть снабжена меткой.

- Зависимость – это связь **использования**, указывающая, что изменение спецификаций одной сущности может повлиять на другие сущности, которые используют ее.

Диаграмма классов

- **Ассоциация** – это структурная связь между элементами модели, которая описывает набор связей, существующих между объектами.
- Ассоциация показывает, что объекты одной сущности (класса) связаны с объектами другой сущности таким образом, что можно перемещаться от объектов одного класса к другому.
- Например, класс Человек и класс Школа имеют ассоциацию, так как человек может учиться в школе. Ассоциации можно присвоить имя «учится в». В представлении однонаправленной ассоциации добавляется стрелка, указывающая на направление ассоциации.



- Двойные ассоциации представляются линией без стрелок на концах, соединяющей два классовых блока.
- Ассоциация может быть именованной, и тогда на концах представляющей её линии будут подписаны роли, принадлежности, индикаторы, мультипликаторы, видимости или другие свойства.

Диаграмма классов

- **Множественность ассоциации** представляет собой диапазон целых чисел, указывающий возможное количество связанных объектов. Он записывается в виде выражения с минимальным и максимальным значением; для их разделения используются две точки. Устанавливая множественность дальнего конца ассоциации, вы указываете, сколько объектов может существовать на дальнем конце ассоциации для каждого объекта класса, находящегося на ближнем ее конце. Количество объектов должно находиться в пределах заданного диапазона. Множественность может быть определена как единица 1, ноль или один 0..1, любое значение 0..* или *, один или несколько 1..*. Можно также задавать диапазон целых значений, например 2..5, или устанавливать точное число, например 3.

Диаграмма классов

- **Агрегация** – особая разновидность ассоциации, представляющая структурную связь целого с его частями. Как тип ассоциации, агрегация может быть именованной. Одно отношение агрегации не может включать более двух классов (контейнер и содержимое).
- Агрегация встречается, когда один класс является коллекцией или контейнером других. Причём, по умолчанию агрегацией называют агрегацию по ссылке, то есть когда время существования содержащихся классов не зависит от времени существования содержащего их класса. Если контейнер будет уничтожен, то его содержимое — нет.
- Графически агрегация представляется пустым ромбом на блоке класса «целое», и линией, идущей от этого ромба к классу «часть».

Диаграмма классов

- **Композиция** — более строгий вариант агрегации. Известна также как агрегация по значению.
- Композиция – это форма агрегации с четко выраженными отношениями владения и совпадением времени жизни частей и целого. Композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет также уничтожено.
- Графически представляется как и агрегация, но с закрашенным ромбиком.

Диаграмма классов

- Третья связь – **обобщение** – выражает специализацию или наследование, в котором специализированный элемент (потомок) строится по спецификациям обобщенного элемента (родителя). Потомок разделяет структуру и поведение родителя. Графически обобщение представлено в виде сплошной линии с пустой стрелкой, указывающей на родителя.

- Четвертая – **реализация** – это семантическая связь между классами, когда один из них (поставщик) определяет соглашение, которого второй (клиент) обязан придерживаться. Это связи между интерфейсами и классами, которые реализуют эти интерфейсы. Это, своего рода, отношение «целое-часть». Поставщик, как правило, представлен абстрактным классом. В графическом исполнении связь реализации – это гибрид связей обобщения и зависимости: треугольник указывает на поставщика, а второй конец пунктирной линии – на клиента.

Диаграмма классов

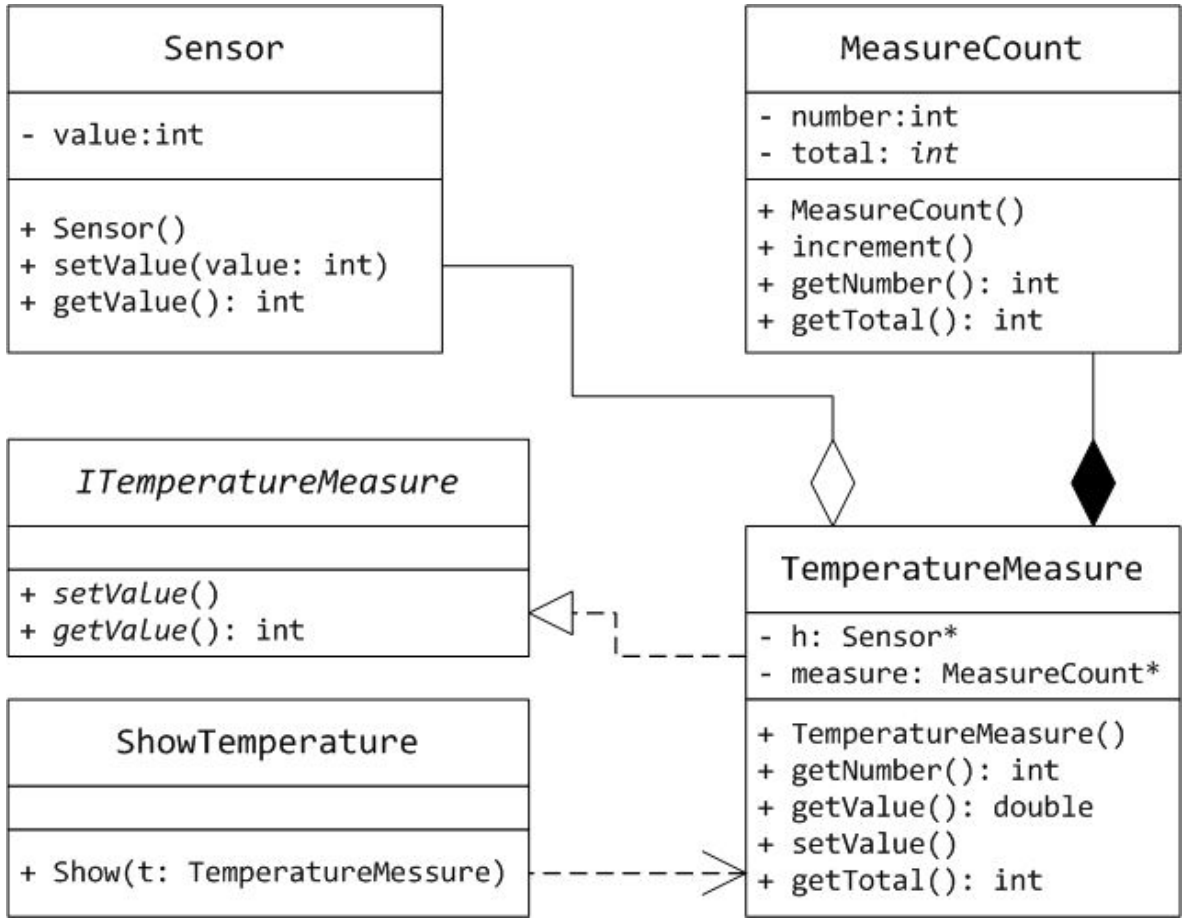


Диаграмма классов

- На диаграмме классов основным классом является класс TemperatureMeasure, который и является измерителем температуры. В качестве измеренного значения формируется среднее арифметическое всех измерений - сумма всех измерений, деленная на их количество.
- Для получения измерений и их суммирования используется класс Sensor (в качестве датчика температуры). В консольной задаче сами измерения передаются в этот класс для суммирования. Класс состоит в отношении агрегации с основным классом TemperatureMeasure: мы сначала создаем объект класса Sensor, а потом передаем его в качестве параметра конструктора классу TemperatureMeasure, чтобы использовать его в качестве части класса.
- Количество измерений формируется классом MeasureCount, который содержит статическое свойство total для подсчета общего измерений, а также свойство count для подсчета количества измерителей конкретного объекта TemperatureMeasure. Класс MeasureCount находится в отношении композиции с классом TemperatureMeasure: объект MeasureCount создается непосредственно при создании объекта TemperatureMeasure (в его конструкторе).
- Класс ITemperatureMeasure представляет собой интерфейс класса TemperatureMeasure и является своего рода поставщиком в отношении реализации.
- Наконец, класс ShowTemperature находится в отношении зависимости с классом TemperatureMeasure, поскольку реализация единственного метода Show класса ShowTemperature зависит от структуры класса TemperatureMeasure.

Задание

1. Найти пример диаграммы состояний и описать ее.
2. Построить свою диаграмму состояний в соответствии с темой курсовой работы.
3. Найти пример диаграммы классов и описать ее.
4. Построить свою диаграмму классов в соответствии с темой курсовой работы.

Работу выполнить в электроном виде в виде отчета.

В отчете обозначить:

- Титульный лист
- Практическая работа
- Тема практической работы
- Цели
- Ход выполнения работы

Дисциплина: «**МДК 01.01. Технология разработки программного обеспечения**»

Лабораторная работа «Построение диаграммы состояний и диаграммы классов»

Преподаватель спец. дисциплин Радунцева Александра Антоновна