Содержание:

Введение.

Объектно – ориентированные языки программирования пользуются в последнее время очень большой популярностью среди программистов, так как они позволяют использовать преимущества объектно- ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

Современное общество уже трудно представить без использования разного рода программ и приложений, которые люди ежесекундно используют в работе, учебе или в попросту быту. За последнее десятилетие разработки новых программных продуктов значительно облегчили нам жизнь, и внесли существенные коррективы в отношение к технике.

Приложения используются в производстве техники и оборудования.
С помощью приложения происходит автоматизация производства, а также он используются для оптимизации бизнес процессов. Часть работы уже роботизирована, и компании уже могут экономить, не нанимая дополнительных сотрудников.

Все это является преимуществом появления разработки и усовершенствования объектно-ориентированного программирования.

В данной курсовой работе предоставлен теоретический материал по объектно – ориентированному программированию, а также составлена схема классов Аппаратной платформы персонального компьютера.

1.

Теоретическая часть.

Объектно –ориентированное программирование.

Объект – это абстрактная сущность, наделенная характеристиками объектов окружающего нас реального мира.

Создание объектов и манипулирование ими – это результат методологии программирования, воплощающей в кодовых конструкциях описания объектов и операции над ними. Каждый объект программы, как и любой реальный объект, отличается собственными атрибутами и характерным поведением. Объекты можно классифицировать по разным категориям: например, мои цифровые наручные часы "Cassio" принадлежат к классу часов. Программная реализация часов входит, как стандартное приложение, в состав операционной системы вашего компьютера. [5]

Каждый класс занимает определенное место в иерархии классов, например, все часы принадлежат классу приборов измерения времени (более высокому в иерархии), а класс часов сам включает множество производных вариаций на ту же тему. Таким образом, любой класс определяет некоторую категорию объектов, а всякий объект есть экземпляр некоторого класса. [4]

Объектно-ориентированное программирование (далее – ООП) – это методика, которая концентрирует основное внимание программиста на связях между объектами, а не на деталях их реализации.

Основные принципы ООП:

- инкапсуляция,
- полиморфизм,
- наследование,
- создание классов и объектов[1]

Инкапсуляция

Инкапсуляция – это механизм, который связывает вместе код и данные и который хранит их от внешнего воздействия и от неправильного использования. Более того, именно инкапсуляция позволяет создавать объект.

Объект – это логическое целое, которое включает в себя данные и код для работы с этими данными. Мы можем определить часть кода и данных как собственность объекта, которая недоступна извне. На этом пути объект обеспечивает существенную защиту против случайной модификации

или некорректного использования таких частных (private) членов объекта. [4]

Во всех случаях объект представляет собой переменную, тип которой определяется пользователем. Вначале кажется странным, что объект, который соединяет вместе и код и данные, как переменную, но в объектном программировании именно так. Если объект определяется неявным образом создается новый тип переменной

Инкапсуляция позволяет в максимальной степени изолировать объект от внешнего окружения. Она существенно повышает надежность разрабатываемых программ, т.к. локализованные в объекте функции обмениваются с программой сравнительно небольшими объемами данных, причем количество и тип этих данных обычно тщательно контролируются. В результате замена или модификация функций и данных, как правило, не влечет за собой плохо прослеживаемых последствий для программы в целом [6]

Полиморфизм

Объектно-ориентированные языки программирования поддерживают полиморфизм, который характеризуется следующей фразой: «один интерфейс — множество методов».

Полиморфизм – это атрибут, который позволяет использовать один и тот же интерфейс при реализации целого класса различных действий. Выбор того, какое именно действие будет совершено, определяется конкретной ситуацией. [1]

В качестве пример полиморфизма можно взять регулятор температуры. Неважно каким типом обогревательного прибора отапливается помещение (с использованием газа, дров, электричества) во всех случаях регулятор температуры работает одинаково. Он является интерфейсом, не важно какой нагревательный прибор и метод отопления используется. Если необходимо иметь температуру 22 градуса по Цельсию, на регуляторе температуры задается эта величина и не важно какой нагревательный прибор используется.

Такой же принцип используется в программировании. Например, в программе, которая определяет три различных типа списков: один используется для целых чисел, другой – для символов и третий для значений с плавающей запятой. Благодаря полиморфизму можно создать два набора функций, имеющих

одинаковое имя push() (поместить) и pop() (извлечь) – по одной для каждого типа данных. Общая концепция (интерфейс) заключается в том, чтобы вставлять и извлекать данные в список
и из списка. Функцию определяют специфические методы, с помощью которых эти операции выполняются для каждого типа данных. Если данные вставляются в список, то автоматически вызывается та версия функции push(), которая соответсятвует типу обрабатываемых данных. [6]

Полиморфизм помогает уменьшить сложность программы, позволяя использовать один и тот же интерфейс для задания целого класса действий. Задача выбора специфического метода в зависимости от конкретной ситуации возлагается на компилятор. Программисту нет необходимости делать такой выбор вручную. Требуется только запомнить и использовать общий интерфейс.

Наследование

Наследование представляет – это процесс, благодаря которому один объект может наследовать, приобретать свойства от другого объекта. Это свойство поддерживает концепцию классификации, чем и обусловливается его важность. Например, красное яблоко представляет собой часть класса яблоко, который, в свою очередь, представляет собой часть класса фрукт, который в свою очередь входит в больший класс продукты питания. Без использования классификации каждый объект должен был бы определять все свои характеристики явным образом. На основе классификации объект нуждается только в определении таких качеств, которые отличают его от других объектов этого класса. Благодаря механизму наследования объект может характеризоваться в рамках классификации общего и частного. [3]

Последовательное проведение в жизнь принципа наследования, особенно при разработке крупных программных проектов, хорошо согласуется с техникой нисходящего структурного программирования (от общего к частному), и во многом стимулирует такой подход. При этом сложность кода программы в целом существенно сокращается. Производный класс (потомок) наследует все свойства, методы и события своего базового класса (родителя) и всех его предшественников в иерархии классов.

При наследовании базовый класс наделяется новыми атрибутами
и операциями. В производном классе обычно появляются новые члены данных,

свойства и методы. При работе с объектами обычно подбирается наиболее подходящий класс для решения конкретной задачи и создания одного или нескольких потомков от него, которые приобретают способность делать не только то,
что заложено в родителе. Дружественные функции позволяют производному классу получить доступ ко всем членам данных внешних классов. [7]

Кроме того, производный класс может перегружать (overload) наследуемые методы в том случае, когда их работа в базовом классе не подходит потомку. Использование перегрузки в ООП всячески поощряется, хотя в прямом понимании значения этого слова перегрузок обычно избегают. Говорят, что метод перегружен, если он ассоциируется с более чем одной одноименной функцией. Обратите внимание, что механизм вызовов перегруженных методов в иерархии классов полностью отличается от вызовов переопределенных функций. Перегрузка и переопределение - это разные понятия. Виртуальные методы используются для переопределения функций базового класса. [1]

Создание классов и объектов

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). В частности, в классах широко используются специальные блоки из одного или чаще двух спаренных методов, отвечающих
за элементарные операции с определенным полем (интерфейс присваивания и считывания значения), которые имитируют непосредственный доступ к полю. Эти блоки называются «свойствами» и почти совпадают по конкретному имени со своим полем (например, имя поля может начинаться со строчной, а имя свойства — с заглавной буквы). Другим проявлением интерфейсной природы класса является то, что при копировании соответствующей переменной через присваивание, копируется только интерфейс, но не сами данные, то есть класс — ссылочный тип данных. [4]

Переменная-объект, относящаяся к заданному классом типу, называется экземпляром этого класса. При этом в некоторых исполняющих системах класс

также может представляться некоторым объектом при выполнении программы посредством динамической идентификации типа данных. Обычно классы разрабатывают таким образом, чтобы обеспечить отвечающие природе объекта и решаемой задаче целостность данных объекта, а также удобный и простой интерфейс. В свою очередь, целостность предметной области объектов и их интерфейсов, а также удобство
их проектирования, обеспечивается наследованием. Объект Сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса (например, после запуска результатов компиляции и связывания исходного кода на выполнение). [6]

Практическая часть.

Задание

Вариант №9: Аппаратная платформа персонального компьютера

- Проанализировать выбранную сущность, выделить наиболее значимые ее составляющие;
- Установить характер связей между составляющими;
- Построить диаграмму классов, отражающую установленные взаимосвязи.

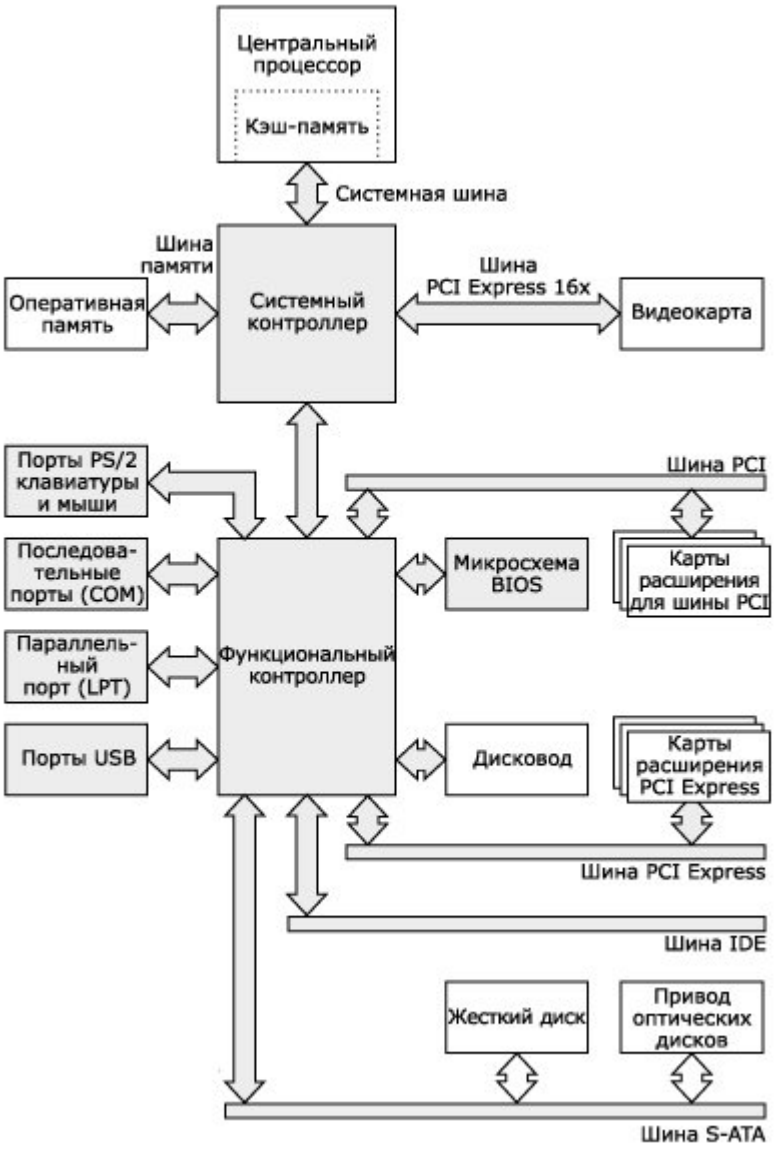
.

Анализ выбранной сущности

Персональный компьютер обладает следующими компонентами:

1. процессор;
2. блок питания;
3. материнская плата;
4. оперативная память;
5. жесткий диск;
6. системы охлаждения;
7. устройства ввода-вывода;
8. звуковая, сетевая карта;
9. функциональные компоненты;
10. дополнительные устройства.

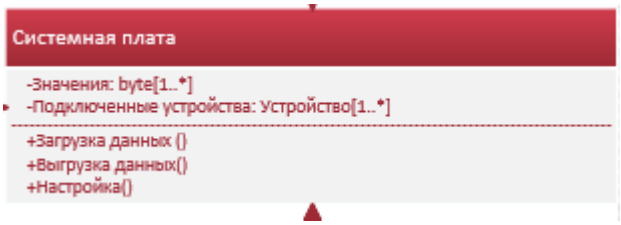
Рисунок 1. Обобщенная схема функциональных аппаратных компонентов персонального компьютера и их взаимосвязей.



Ход проектирования

Персональный компьютер содержит в себе материнскую плату, как главный компонент вычислительной схемы. Материнская, или системная, плата является главной сущностью, которая будет включать в себя все остальные компоненты, притом таковые компоненты не могут функционировать без связующей системной шины, которая находится на материнской плате, и потому данная сущность будет иметь отношение “Композиция”.

Рисунок 2. Сущность “Материнская плата”.

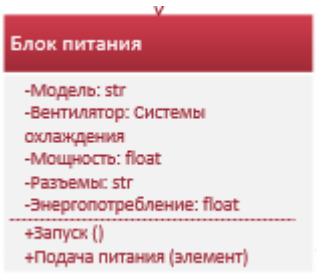


Неотъемлемой частью материнской платы является гнездо для процессора. Именно такую сущность необходимо создать, притом стоит учесть, что процесс использует такие компоненты, как: кэш, ядра и может иметь встроенный графический процессор. ЦПУ также включает в себя такие компоненты, как АЛУ и гнездо чипсета – таковые характеристики предполагаются, раз имеется процессор. Таким образом, данная сущность будет объединять неотъемлемые составные части сущности, являясь отношением “Композиция”.

Рисунок 3. Сущность “Процессор”.

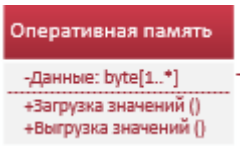
Для запуска компьютера необходим блок питания. Таковой элемент подключается напрямую к материнской плате, питая электрическом ее компоненты. Блок питания требует системы охлаждения, так как является источником энергии, в следствии чего температура, как и энергопотребление – главные характеристики сущности. Блок питания учувствует только в подключении к материнской плате (и другим энергозависимым элементам), и не взаимосвязан с другими компонентами, за исключением систем охлаждения.

Рисунок 4. Сущность “Блок питания”.



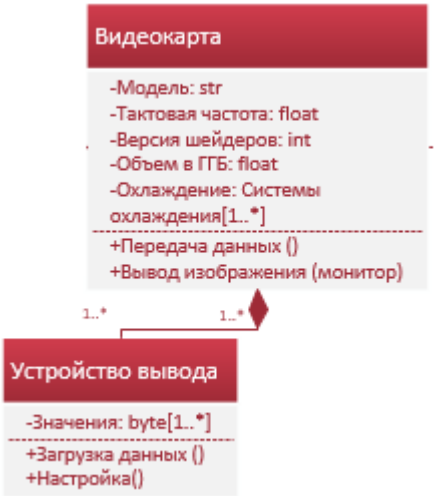
Быстродействие компьютера определяется во многом оперативной памятью, притом количество плашек может быть любым, но как минимум одна должна быть установлена.

Рисунок 5. Сущность “Оперативная память”.



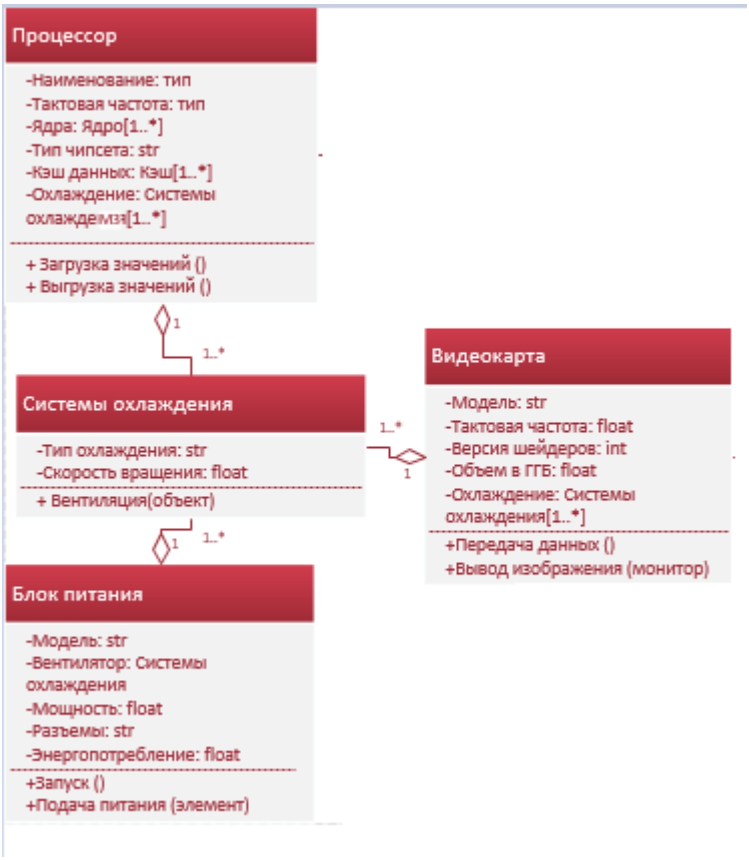
Видеокарта также определяет быстродействие компьютера, на она ответственна за графическую составляющую. Видеокарта нуждается в охлаждении из-за высоких нагрузок. Видеокарта применяется как в северном (наиболее быстром) мосту, так и на южном (являясь посредником-буфером). Устройство вывода напрямую соотносится с видеокартой. Видеокарт в компьютере может быть несколько. Отметим, что таковая конфигурация подходит только для пользовательского ПК – иногда, как к примеру, на серверах, графический вывод информации не нужен.

Рисунок 6. Сущность “Видеокарта”.



Системы охлаждения необходимы для всех элементов, которые могут испытывать термическую коррозию, следовательно – фактических для всех элементов необходимы процедуры сброса высокой температуры. Так как без вентиляторов система не сможет просуществовать длительное время – как минимум один из источников охлаждения должен присутствовать в аппаратной платформе, притом таковой источник влияет на все части системы. Наиболее часто кулеры (системы охлаждения) ставятся на процессор, видеокарту, блок питания и видеокарту – отобразим таковое отношение “Аггрегация”.

Рисунок 7. Сущность “Системы охлаждения”.



Совокупность таких элементов, как процессор, видеокарта, оперативная память, представляет собой северный мост на материнской плате.

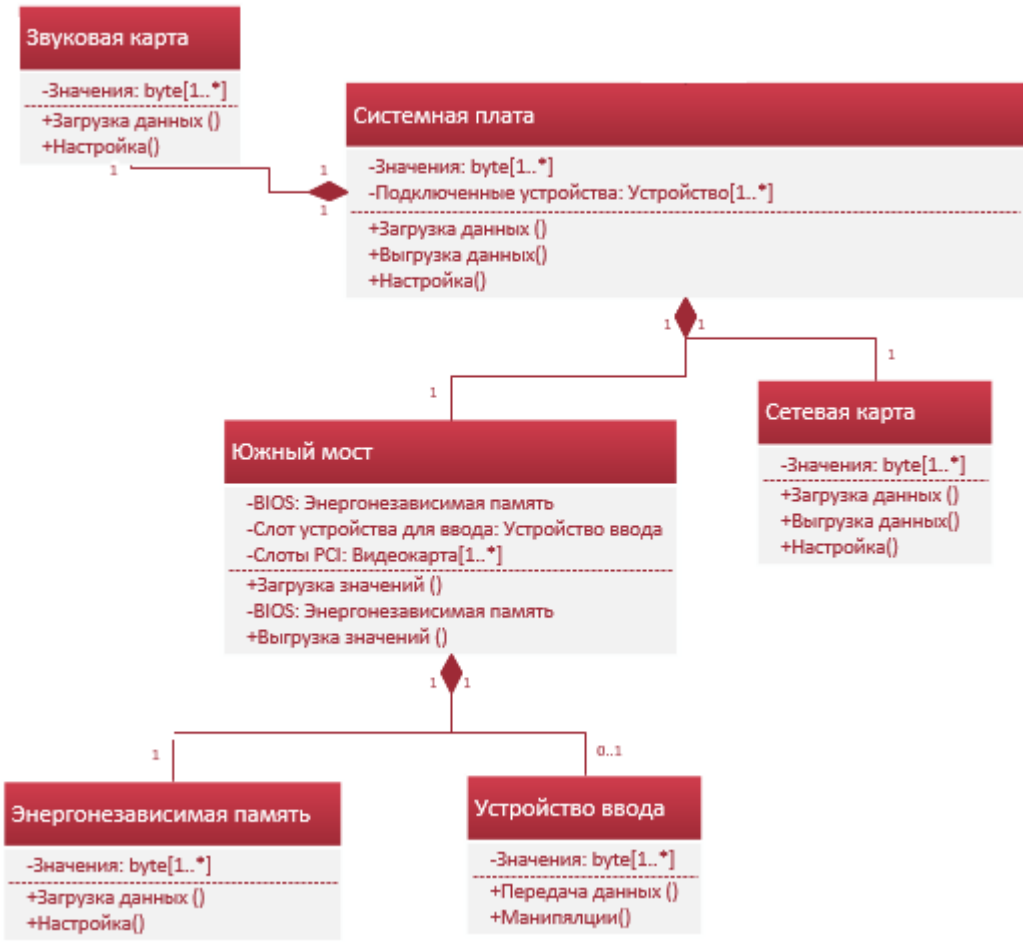
Рисунок 8. Сущность “Северный мост”.

В компьютер зачастую подключаются дополнительные устройства, такие как TV-тюнер, внешние носители информации, профессиональное оборудование и т.п. Отобразим это на схеме:

Рисунок 9. Сущность “Дополнительные устройства”.

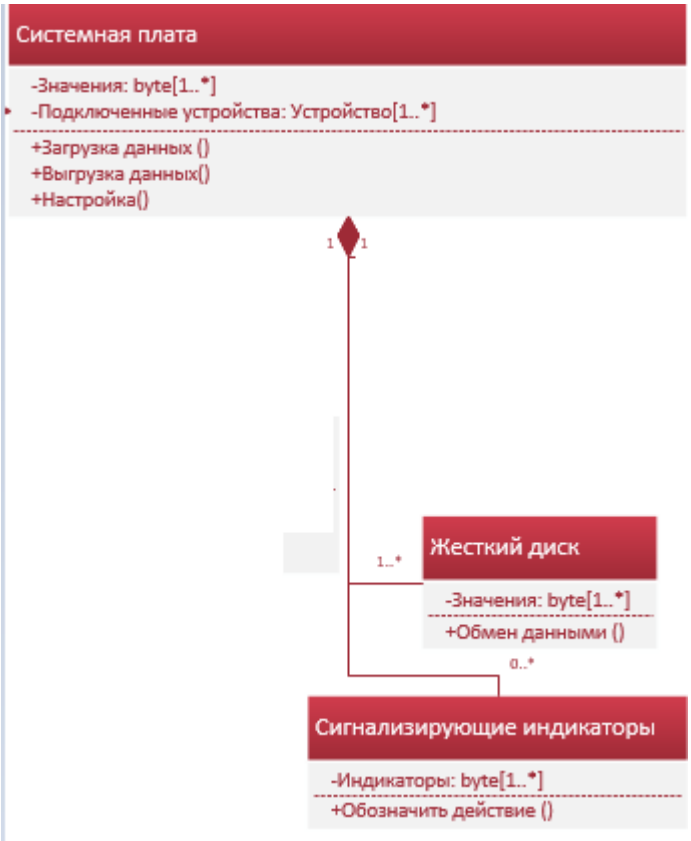
Звуковая карта, как и сетевая, как и BIOS – стандартные компоненты материнской платы, они влияют на нее напрямую. BIOS в совокупности с вводом-выводом составляет собой южный мост платы.

Рисунок 10. Сущность “Южный мост”.



Компьютеру необходим съемный носитель информации, а также индикаторы, которые бы оповещали пользователя о происходящих изменениях в системе. Таковые элементы, как правило, присутствует на системных платах.

Рисунок 11. Сущность “Жесткий диск” и “Сигнализирующие индикаторы”.

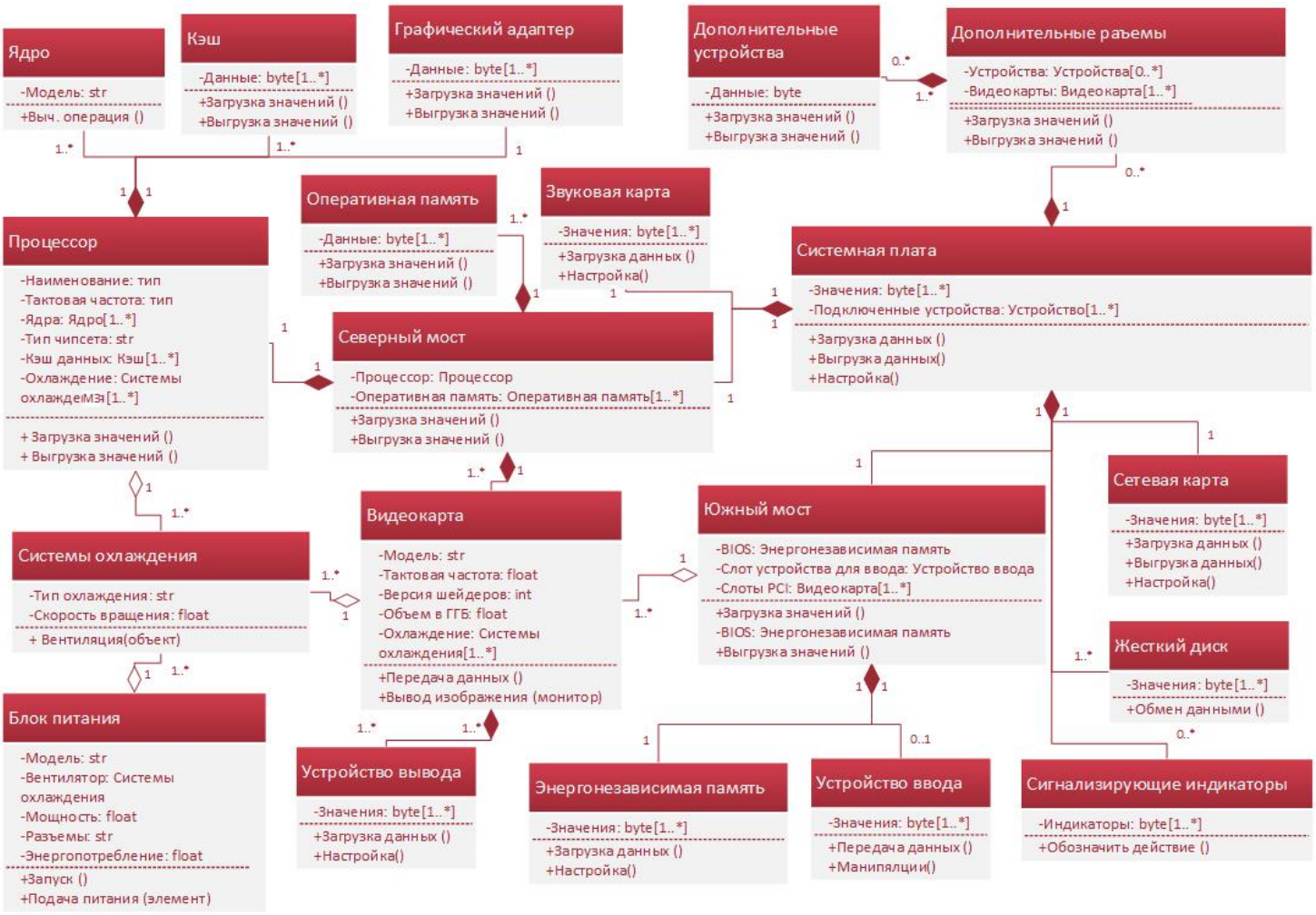


Совокупностью описываемых элементов будет являться полноценная диаграмма:

Результирующая схема

Соединяя и образуя взаимосвязи между разработанными сущностями – результирующим итогом получим диаграмму классов.

Рисунок 12. Диаграмма классов “Аппаратная платформа персонального компьютера” и “Сигнализирующие индикаторы”.



Заключение.

В курсовой были в начале используя специализированные средства для построенных диаграмм и схем классов – Microsoft Visio – удалось решить требуемые задачи,
и в свою очередь сделать это качественно, визуально, наглядно и эффективно.
По созданной схеме можно проследить работу выбранного объекта,
при детализации – можно понять, как функционирует анализируемая сущность в объекте. Рассмотрев, проанализировав и составив схему классов можно повысить свой навык в ООП-тематике.

Список используемой литературы.

1. Онлайн справочник «Программирование на C++ - http://www.c-cpp.ru/books/nasledovanie

2. Электронная книга «Объектно-ориентированное программирование и C++» - http://bourabai.ru/C-Builder/3.htm#resume
3. Учебное пособие «Программирование на языке СИ»
   Страуструп Б. Язык программирования C++. – Москва: Бином, 2006, 1104 с.
4. Объектно-ориентированное программирование (Учебное пособие). Автор: Шахгельдян К.И., редактор: Александрова Л.И. - https://abc.vvsu.ru/Books/u_programm/
5. Гради Буч «Объектно-ориентированный анализ и проектирование с примерами приложений»
6. Зыков С.В. Введение в теорию программирования. Объектно-ориентированный подход электронный ресурс - http://www.intuit.ru/department/se/tppobj/
7. Бертран Мейер. Основы объектно-ориентированного программирования электронный ресурс - http://www.intuit.ru/department/se/oopbases/