**Ministry of education and science of the Kyrgyz Republic**

**Kyrgyz State Technical University named after I.Razzakov**

**Faculty of Information Technologies**

**Department of Software of Computer Systems**

**Major: 710400 «Software Engineering»**

Report

Discipline: «**Object-Oriented Design**»

Software requirements

Done by**:** student of the SE(eng)- 1- 21

Shumkarbekov Sultan

Checked by: Musabaev E.B.

Bishkek 2024

**Functional requirements**

**Task1:**

1. Input Employee Information
2. Display Employee Information
3. Input Manager Information
4. Display Manager Information

**Task2:**

1. **Display Class**:
   * The system should allow the creation of a display object with specified parameters including display type, number of colors, and resolution.
2. **MotherBoard Class**:
   * The system should allow the creation of a motherboard object with specified parameters including processor type, processor speed, and RAM size.
3. **Computer Class**:
   * The system should allow the creation of a computer object based on the parent classes **Display** and **MotherBoard**.
4. **Input Validation**:
   * The system should validate input parameters to ensure they are within acceptable ranges.
5. **User Interface**:
   * The system should provide a user-friendly interface for users to input parameters and view information about computers, displays, and motherboards.

**Nonfunctional requirements**

1. Response Time: The program should quickly calculate results,
2. Availability-is the system available only when connected to the internet or does it also work offline
3. Usability-The interface must be simple and understandable
4. OS Windows 7 and higher
5. Processor intel core i5 and higher

**Answers to questions:**

1. **В чем заключается выигрыш для программиста при использовании им наследования?**

Выигрыш от него состоит в том, что наследование позволяет использовать существующий код несколько раз. Имея написанный и отлаженный базовый класс, мы можем его больше не модифицировать, при этом механизм наследования позволит нам приспособить его для работы в различных ситуациях. Используя уже написанный код, мы экономим время и деньги, а также увеличиваем надежность программы.

1. **Какие члены базового класса не наследуются?**

При наследовании классов производный класс создается не на пустом месте, а на основе базового класса, получая в наследство от своего родителя поля и методы. Тем не менее из того, что производный класс наследует свойства базового класса, еще не следует, что он наследует *все* свойства.

1. **От чего зависит доступ к членам базового класса из произвольного класса?**

Какие поля и методы наследуются, а какие нет, определяется доступностью соответствующих членов базового класса и *механизмом наследования*.

1. Чем отличается идентификатор доступа от механизма доступа?

Идентификатор доступа определяет уровень доступа к членам класса (например, переменным и методам) извне класса. Он может быть public, protected или private и контролирует, какие части программы могут получить доступ к членам класса.

Механизм доступа - это способ, которым можно получить доступ к членам класса с использованием идентификаторов доступа. Он определяет, как именно члены класса будут доступны из других частей программы в соответствии с установленными идентификаторами доступа.

1. **Назовите основные правила наследования (используйте таблицу наследования).**

|  |  |  |  |
| --- | --- | --- | --- |
| **Механизм** | public- наследование | private- наследование | protected - наследование |
| **Доступность** |  |  |  |
| рublic - член | public | private | protected |
| рrivate- член | нет доступа | нет доступа | нет доступа |
| рrotected - член | public | private | protected |

1. **Синтаксис объявления производного класса, как наследника базового класса?**

Class B: public A

1. Что такое простое, множественное и многократное наследование?

**Простое наследование -** это когда у производного класса имеется один базовый класс.

**Множественное наследование -** когда у производного класса имеется несколько базовых классов.

**Многократное наследование –** это когда производный класс может иметь и базовый и производный классы.

1. **Особенности использования идентификатор доступа protected при наследовании классов?**

Члены с идентификатором доступа protected доступны в производном классе и в его классах-наследниках.

Они не доступны извне класса, но доступны в методах производного класса и его наследников.

Этот идентификатор позволяет реализовать концепцию инкапсуляции и защищает данные от прямого доступа извне, но при этом предоставляет доступ к ним в классах-наследниках.

1. **Какие существуют виды конструкторов при наследовании?**

Конструктор по умолчанию: Создает объект с базовым классом с использованием его конструктора по умолчанию.

Конструктор с аргументами: Создает объект с базовым классом, используя его конструктор с аргументами. Может передавать аргументы конструктору базового класса через инициализацию членов в списке инициализации.

Копирующий конструктор: Создает объект, копируя данные из другого объекта того же класса. При наследовании, он также может вызывать копирующий конструктор базового класса.

1. **Каков синтаксис объявления конструктора производного класса?**

Manager() : Employee(), annual\_bonus(0.0), company\_car(""), stock\_options(0) {}

1. **Каковы особенности конструктора при простом наследовании**?

Конструктор производного класса должен вызвать конструктор базового класса явно через список инициализации.

Если конструктор производного класса не вызывает конструктор базового класса, то вызывается конструктор по умолчанию базового класса.

1. **Каковы особенности конструктора при многократном наследовании?**

Конструкторы базовых классов вызываются в порядке, в котором они указаны в списке наследования в объявлении производного класса.

При множественном наследовании конструкторы каждого базового класса вызываются независимо друг от друга.

1. **Какова последовательность вызова конструкторов и деструкторов при наследовании?**

При создании объекта производного класса сначала вызываются конструкторы базовых классов, начиная с наиболее базового класса, затем конструктор производного класса.

При удалении объекта вызываются деструкторы в обратной последовательности: сначала деструктор производного класса, затем деструкторы базовых классов.