**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

**Лабораторная работа №4**

по дисциплине: Метрология, стандартизация и сертификация программного обеспечения

тема: «Метрики объектно-ориентированных программных систем»

Выполнил: ст. группы ПВ-223

Пахомов Владислав Андреевич

Проверили:

ст. пр. Осипов Олег Васильевич

Белгород 2025 г.

**Лабораторная работа №4  
Метрики объектно-ориентированных программных систем  
Вариант 6**

**Цель работы:** изучить теоретические сведения и получить практические навыки оценки иерархии классов объектно-ориентированных программных систем.

**Задания для выполнения к работе:**

1. Реализовать диаграмму классов собственной объектно-ориентированной программной системы.
2. Для каждого класса указать все его свойства и методы, кратко охарактеризовать их назначение и смысл.
3. Определить значения метрик из набора метрик Чидамбера и Кемерера.
4. Сформулировать рекомендации по модификации составленной иерархии классов на основании вычисленных значений метрик Чидамбера и Кемерера.
5. Определить значения метрик из набора метрик Лоренца и Кидда.
6. Сформулировать рекомендации по модификации составленной иерархии классов на основании вычисленных значений метрик Лоренца и Кидда

**Программа «CBattle»**

**Диаграмма классов:**

****

**Suspicion**

*Описание:*

Data-класс, содержащий позицию «попадания»

*Свойства:*

(x и y) и текущее предполагаемое направление, куда нужно бить в следующий раз.

*Методы:*

* Suspicion – дефолтный конструктор
* Suspicion – конструктор с указанием позиции и направления

**BotGameGlient**

*Описание:*

Класс, содержащий логику для игры против компьютера.

*Свойства:*

Содержит игру game, флаг prev\_step\_mine – был ли его ход прошлым, prev\_x, prev\_y – то, куда он походил в прошлый раз, prev\_suspicion – предыдущее предполагаемое место нахождения нужной клетки с кораблём и suspicions – список подозреваемых клеток.

*Методы:*

* onUpdate – метод, вызываемый при обновлении игры
* onError – метод, вызываемый при ошибке
* init – метод-инициализатор, вызываемый при начале игры, отправляет «рукопожатие» серверу
* BotGameClient – конструктор
* ~BotGameClient – деструктор

**GameClient**

*Описание:*

Абстрактный класс для создания клиента способного играть в морской бой.

*Свойства:*

Содержит conn – коннектор к игре, через который клиент может взаимодействовать с игрой.

*Методы:*

* onUpdate – метод, вызываемый при обновлении игры
* onError – метод, вызываемый при ошибке
* init – метод-инициализатор, вызываемый при начале игры, отправляет «рукопожатие» серверу
* onStep – метод, вызываемый если был выполнен шаг в игре
* onHandshake – клиент установил связь с сервером и выполнил рукопожатие
* onSurrender – метод, вызываемый при сдаче

**ClientConnection**

*Описание:*

Коннектор, через который клиент и общается с сервером. Реализует паттерн «стратегия»

*Свойства:*

Содержит strategy – стратегию, по которой будет взаимодействовать коннектор

*Методы:*

* ClientConnection – конструктор, принимающий стратегию
* ~ClientConnection – деструктор, освобождающий память
* sendHandshake – отправляет «рукопожатие» серверу с позицией кораблей
* sendStep – отправялет серверу шаг
* onUpdate – метод, вызываемый при обновлении игры
* onError – метод, вызываемый при ошибке
* onSurrender – метод, вызываемый при сдаче

**ClientConnectionStrategy**

*Описание:*

Абстрактная стратегия для коннектора. Реализует паттерн «стратегия»

*Свойства:*

Содержит указатель на игровой клиент client

*Методы:*

* ClientConnectionStrategy – конструктор, принимающий клиент
* ~ClientConnectionStrategy – деструктор, освобождающий ресурсы
* sendHandshake – отправляет «рукопожатие» серверу с позицией кораблей
* sendStep – отправялет серверу шаг
* onUpdate – метод, вызываемый при обновлении игры
* onError – метод, вызываемый при ошибке
* onSurrender – метод, вызываемый при сдаче

**LocalClientConnectionStrategy**

*Описание:*

Стратегия коннектора, имеющая непосредственный доступ по указателю к коннектору сервера.

*Свойства:*

Содержит указатель на серверное соединение conn а также на клиент client

*Методы:*

* LocalClientConnectionStrategy– конструктор, принимающий клиент и серверный коннектор
* ~ LocalClientConnectionStrategy– деструктор, освобождающий ресурсы
* sendHandshake – отправляет «рукопожатие» серверу с позицией кораблей
* sendStep – отправялет серверу шаг
* onUpdate – метод, вызываемый при обновлении игры
* onError – метод, вызываемый при ошибке
* onSurrender – метод, вызываемый при сдаче

**TCPClientConnectionStrategy**

*Описание:*

Стратегия коннектора, имеющая доступ к серверу посредством протоколов TCP/IP

*Свойства:*

Содержит указатель на серверное соединение по TCP conn, клиент client и массив временных данных temp

*Методы:*

* TCPClientConnectionStrategy– конструктор, инициализирующий объект QT, устанавливающий соединение
* ~ TCPClientConnectionStrategy– деструктор, освобождающий ресурсы и закрывающий TCP-соединение
* onReadyRead – обработчик событий TCP, парсящий входные данные и вызывающий соответствующий метод.
* sendHandshake – отправляет «рукопожатие» серверу с позицией кораблей
* sendStep – отправялет серверу шаг
* onUpdate – метод, вызываемый при обновлении игры
* onError – метод, вызываемый при ошибке
* onSurrender – метод, вызываемый при сдаче

**ServerConnection**

*Описание:*

Абстрактный класс, выполняющий соединение от сервера к клиенту

*Свойства:*

Содержит саму игру game

*Методы:*

* ServerConnection– конструктор, принимающий игру
* ~ ServerConnection– деструктор, освобождающий ресурсы
* sendUpdate – отправляет клиенту обновление игры
* sendError– отправляет клиенту ошибку игры
* onStep – принимает от клиента «шаг»
* onHandshake – метод, вызываемый при рукопожатии от клиента
* onSurrender – метод, вызываемый при сдаче

**LocalServerConnection**

*Описание:*

Реализация абстрактного класса ServerConnection для локальной игры (с непосредственным доступом к коннектору клиента по указателю)

*Свойства:*

Содержит саму игру game, клиентское соединение conn

*Методы:*

* LocalServerConnection – конструктор, принимающий игру
* ~LocalServerConnection – деструктор, освобождающий ресурсы
* sendUpdate – отправляет клиенту обновление игры
* sendError– отправляет клиенту ошибку игры
* onStep – принимает от клиента «шаг»
* onHandshake – метод, вызываемый при рукопожатии от клиента
* onSurrender – метод, вызываемый при сдаче
* onReadyRead – обработчик событий TCP, парсящий входные данные и вызывающий соответствующий метод.

**TCPServerConnection**

*Описание:*

Реализация абстрактного класса ServerConnection для сетевой игры по TCP.

*Свойства:*

Содержит указатель на клиентское соединение по TCP conn, игру game и массив временных данных temp

*Методы:*

* TCPServerConnection – конструктор, инициализирующий объект QT, устанавливающий соединение
* ~TCPServerConnection – деструктор, освобождающий ресурсы и закрывающий TCP-соединение
* sendUpdate – отправляет клиенту обновление игры
* sendError– отправляет клиенту ошибку игры
* onStep – принимает от клиента «шаг»
* onHandshake – метод, вызываемый при рукопожатии от клиента
* onSurrender – метод, вызываемый при сдаче

**GameServer**

*Описание:*

Класс, реализующий основную бизнес-логику игры, поддерживающий целостность игрового процесса, оповещающий клиента об изменениях и принимающий от них сообщения.

*Свойства:*

Содержит going\_one – флаг, ходит ли первый игрок, game\_over – окончена ли игра, player\_one\_won – выиграл ли первый победитель, winning\_reason – причина победы (противник сдался или честный выигрыш), player\_one\_field – поле первого игрока, player\_one\_validated – готов ли первый игрок, player\_two\_field – поле второго игрока, player\_one – коннектор к первому игроку, player\_two – коннектор ко второму игроку.

*Методы:*

* GameServer – конструктор, инициализирующий сервер игры
* ~GameServer– деструктор, освобождающий ресурсы и закрывающий TCP-соединение
* checkGameOver – проверяет, закончена ли игра
* isCovered – проверяет, «убит» ли корабль
* onHandshake – метод, вызываемый при рукопожатии от клиента, инициализирует поля игрока и проверяет валидность полей
* onSurrender – метод, вызываемый при сдаче, назначает оставшегося игрока победителем и прекращает игру
* onStep – метод, вызываемый при совершении игроком шага, проверяет, можно ли совершить шаг и совершает его, проверяет конец ли игры и оповещает пользователей об изменении
* sendUpdate – оповещает пользователей о состоянии игры

**AbstractGameBuilder**

*Описание:*

Абстрактный класс, реализующий паттерн «билдер», позволяет установить игру

*Свойства:*

Содержит server, если пользователь является «хостом», то оно не null, playing\_client – первый игрок, vs\_client – игрок противник, playing\_setup – установлен ли первый игрок, vs\_setup – установлен ли второй игрок.

*Методы:*

* AbstractGameBuilder – конструктор, инициализирующий билдер игры
* ~AbstractGameBuilder– деструктор, освобождающий ресурсы и закрывающий TCP-соединение
* playing – устанавливает первого игрока как локального игрока так или через TCP-сокет, оборачивая оба способа в соответствующие стратегии и коннекторы
* playing – устанавливает второго игрока как локального игрока так или через TCP-сокет, оборачивая оба способа в соответствующие стратегии и коннекторы
* begin – проверяет готовность игроков и начинает игру.

**GameBuilder**

*Описание:*

Реализация абстрактного класса, AbstractGameBuilder

*Свойства:*

Содержит server, если пользователь является «хостом», то оно не null, playing\_client – первый игрок, vs\_client – игрок противник, playing\_setup – установлен ли первый игрок, vs\_setup – установлен ли второй игрок.

*Методы:*

* AbstractGameBuilder – конструктор, инициализирующий билдер игры
* ~AbstractGameBuilder– деструктор, освобождающий ресурсы и закрывающий TCP-соединение
* playing – устанавливает первого игрока как локального игрока так или через TCP-сокет, оборачивая оба способа в соответствующие стратегии и коннекторы
* playing – устанавливает второго игрока как локального игрока так или через TCP-сокет, оборачивая оба способа в соответствующие стратегии и коннекторы
* begin – проверяет готовность игроков и начинает игру.
* establishLocalConnection – утилитный метод для установления локальной связи между сервером и клиентом

**Вывод:** в ходе лабораторной работы изучили теоретические сведения и получили практические навыки оценки иерархии классов объектно-ориентированных программных систем.