**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
* y - 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:*

* game - игра
* 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
* 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:*

* 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:*

* strategy – стратегия, 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:*

* 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:*

* указатель на серверное соединение conn
* указатель 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:*

* Указатель на серверное соединение по TCP conn
* Клиент client
* 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:*

* Сама игра 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:*

* Сама 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:*

* Указатель на клиентское соединение по TCP conn
* Игра game
* 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:*

* 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:*

* 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:*

* 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

**Field**

*Описание:*

Dataкласс, содержащее поле с некоторыми утилитными методами.

*Свойства:*

* field – матрица с отметками игрока
* ships – массив кораблей на поле.

*Методы:*

* areInBounds – проверяет, находится ли координата в пределах поля
* generateRandom – конструирует поле со случайно расположенными кораблями
* validateShips – валидирует положение кораблей на поле

**Game**

*Описание:*

Dataкласс, содержащее упрощённое состояние игры для отображения

*Свойства:*

* game\_over – флаг, конец ли игры
* youre\_winner – является ли текущий игрок победителем
* youre\_going – ход игрока
* reason – причина победу
* own\_field – состояние собственного поля

enemy\_field – состояние вражеского поля

**Ship**

*Описание:*

Dataкласс, содержащий информацию о корабле

*Свойства:*

* x
* y – позиция корабля
* length – длина корабля
* is\_horizontal – вертикальный ли корабль

*Методы:*

* Ship - конструктор

**MoveableObjectAdapter**

*Описание:*

Абстрактный класс, предназначенный для Drag-N-Drop поведения View внутри родителя

*Свойства:*

* parent – родительский View
* object – модель объекта
* graph\_object – View для отображения
* offset – отступ для мышки
* style – стиль поля

*Методы:*

* construct\_object – создаёт View для отображения
* onUpdateScenePos – на основе положения курсора обновляет положение graph\_object
* MoveableObjectAdapter – конструктор
* ~MoveableObjectAdapter – деструктор, удаляющий View из сцены и освобождающий ресурсы
* init – создаёт объект для отображения и добавляет его в родительский компонент
* getObject – геттер для модели объекта
* getOffset – геттер для получения offset

**MoveableObjectShip**

*Описание:*

Реализация MoveableObjectAdapter для Ship

*Свойства:*

* parent – родительский View
* object – модель объекта
* graph\_object – View для отображения
* offset – отступ для мышки
* style – стиль поля

*Методы:*

* construct\_object – создаёт View для отображения
* onUpdateScenePos – на основе положения курсора обновляет положение graph\_object
* MoveableObjectAdapter – конструктор
* ~MoveableObjectAdapter – деструктор, удаляющий View из сцены и освобождающий ресурсы
* init – создаёт объект для отображения и добавляет его в родительский компонент
* getObject – геттер для модели объекта
* getOffset – геттер для получения offset

**FieldStyle**

*Описание:*

Класс, который задаёт стили для поля

*Свойства:*

* padding\_percent – отступ для кораблей
* pen\_grid – кисть для отрисовки сетки
* pen\_ship – кисть для отрисовки корабля
* brush\_ship – кисть для закрашивания корабля
* cross\_img – изображение для отметки «не попал»
* explosion\_img – изображение для отметки «попал»
* DEFAULT\_PADDING\_PERCENT – стандартный отступ
* DEFAULT\_PEN\_GRID – стандартная кисть для отрисовки сетки
* DEFAULT\_PEN\_SHIP – стандартная кисть для отрисовки корабля
* DEFAULT\_BRUSH\_SHIP – стандартная кисть для закрашивания корабля
* DEFAULT\_CROSS – стандартный путь для отметки «не попал»
* DEFAULT\_EXPLOSION – стандартный путь для отметки «попал»
* STYLE\_EDIT – стиль для поля в режиме редактирования
* STYLE\_PLAY\_SELF – стиль для собственного поля
* STYLE\_PLAY\_ENEMY – стиль для поля врага

*Методы:*

* FieldStyle – конструктор стиля поля
* FieldStyle – конструктор стиля поля со стандартными полями
* ~ FieldStyle – деструктор стиля поля
* initStyles – инициализирует стандартные стили
* getPaddingPercent – геттер для отступа для кораблей
* getPenGrid – геттер для кисти для отрисовки сетки
* getPenShip – геттер для отрисовки корабля
* getBrushShip – геттер для кисти для закрашивания корабля
* getCrossImg – геттер для отметки «не попал»
* getExplosionImg – геттер для отметки «попал»

**FieldWidget**

*Описание:*

Виджет для отрисовки поля

*Свойства:*

* field – игровое поле
* style – стиль поля

*Методы:*

* FieldWidget – конструктор виджета
* setField – устанавливает игровое поле
* getField – геттер игрового поля
* redraw – перерисовывает поле
* getShipScenePos – возвращает точки прямоугольника корабля на поле
* getCollidedShipIndex – возвращает индекс корабля в указанной точке
* getCellCoordinate – возвращает позицию клетки на основе координаты курсора

**FieldWidgetEdit**

*Описание:*

Виджет для редактируемого поля

*Свойства:*

* field – игровое поле
* style – стиль поля
* moveable\_object – перемещаемый объект

*Методы:*

* FieldWidgetEdit – конструктор виджета
* setField – устанавливает игровое поле
* getField – геттер игрового поля
* redraw – перерисовывает поле
* getShipScenePos – возвращает точки прямоугольника корабля на поле
* getCollidedShipIndex – возвращает индекс корабля в указанной точке
* getCellCoordinate – возвращает позицию клетки на основе координаты курсора
* mouseMoveEvent – событие перемещение мыши, на его основе выбранный корабль меняет свою позицию на поле
* mousePressEvent – выбирает корабль на поле, а также меняет его вращение, если было нажато ПКМ
* mouseReleaseEvent – устанавливает позицию корабля при отпускании ЛКМ

**FieldWidgetPlayEnemy**

*Описание:*

Виджет для вражеского поля

*Свойства:*

* field – игровое поле
* style – стиль поля
* client – клиент игры

*Методы:*

* FieldWidgetEdit – конструктор виджета
* setField – устанавливает игровое поле
* getField – геттер игрового поля
* redraw – перерисовывает поле
* getShipScenePos – возвращает точки прямоугольника корабля на поле
* getCollidedShipIndex – возвращает индекс корабля в указанной точке
* getCellCoordinate – возвращает позицию клетки на основе координаты курсора
* mousePressEvent – выбирает корабль на поле и отправляет по коннектору событие удара по кораблю

**GameWindow**

*Описание:*

Виджет для отображения игры, наследуется от игрового клиента

*Свойства:*

* ui – объект со всеми элементами виджета
* enemy\_field – виджет вражеского поля
* own\_field – виджет собственного поля
* handshake\_ok – успешно ли рукопожатие
* game – игра
* ships - корабли

*Методы:*

* GameWindow – конструктор виджета
* ~GameWindow – деструктор виджета
* onUpdate – метод, вызываемый при обновлении игрового состояния, на его основе отображает или прячет элементы, перерисовывает поля
* onError– обрабатывает ошибки с сервера
* init – отсылает рукопожатие серверу
* showEvent – при инициализации окна инициализирует виджеты поля игроков
* surrender – сдаётся
* closeEvent – при закрытии игры сообщает серверу о том, что игрок сдаётся

**InitGame**

*Описание:*

Виджет для начала игры

*Свойства:*

* ui – объект со всеми элементами окна
* game\_window – окно редактирования поля
* server – TCP сервер
* game\_going – проходит ли игра

*Методы:*

* setupServer– инициализирует компоненты для инициализации TCP сервера
* downServer – выключает сервер
* beginBot – создаёт игру с компьютером
* beginCreate – создаёт или выключает TCP сервер
* beginJoin – присоединяется к игре по TCP
* InitGame – конструктор
* ~InitGame – деструктор
* showEvent – при показе окна создаёт поле с кораблями для редактирования
* newConnection – метод при новом подключении к TCP серверу

**Метрики Чидамбера и Кемерера**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **WMC** | **DIT** | **NOC** | **CBO** | **RFC** | **НЕ СВЯЗАНЫ** | **СВЯЗАНЫ** | **LOCM** |
| Suspicion | 2 | 0 | 0 | 0 | 2 | 1 | 0 | 1 |
| BotGameGlient | 5 | 1 | 0 | 2 | 23 | 6 | 2 | 4 |
| GameClient | 6 | 0 | 2 | 3 | 22 | 0 | 6 | 0 |
| ClientConnection | 7 | 0 | 0 | 3 | 21 | 0 | 21 | 0 |
| ClientConnectionStrategy | 7 | 0 | 2 | 3 | 21 | 1 | 0 | 1 |
| LocalClientConnectionStrategy | 7 | 1 | 0 | 5 | 21 | 12 | 6 | 6 |
| TCPClientConnectionStrategy | 8 | 1 | 0 | 7 | 14 | 17 | 10 | 7 |
| ServerConnection | 7 | 0 | 2 | 3 | 7 | 1 | 0 | 1 |
| LocalServerConnection | 8 | 1 | 0 | 5 | 20 | 10 | 1 | 9 |
| TCPServerConnection | 7 | 1 | 0 | 6 | 11 | 15 | 6 | 9 |
| GameServer | 8 | 0 | 0 | 4 | 20 | 18 | 15 | 3 |
| AbstractGameBuilder | 5 | 0 | 1 | 3 | 14 | 0 | 0 | 0 |
| GameBuilder | 6 | 1 | 0 | 7 | 17 | 0 | 0 | 0 |
| Field | 3 | 0 | 0 | 1 | 4 | 0 | 0 | 0 |
| Game | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Ship | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| MoveableObjectAdapter | 7 | 0 | 1 | 5 | 11 | 6 | 7 | 0 |
| MoveableObjectShip | 7 | 1 | 0 | 6 | 12 | 0 | 0 | 0 |
| FieldStyle | 10 | 0 | 0 | 5 | 14 | 28 | 0 | 28 |
| FieldWidget | 7 | 1 | 2 | 6 | 22 | 11 | 10 | 1 |
| FieldWidgetEdit | 10 | 2 | 0 | 7 | 25 | 0 | 3 | 0 |
| FieldWidgetPlayEnemy | 8 | 2 | 0 | 7 | 23 | 0 | 1 | 0 |
| GameWindow | 8 | 1 | 0 | 10 | 34 | 17 | 7 | 10 |
| InitGame | 9 | 1 | 0 | 8 | 51 | 3 | 33 | 0 |
|  | **153** | **2** | **10** | **107** | **410** |  |  | **80** |

Дерево зависимости имеет высоту максимум в два элемента, ширина также составляет два элемента. Проект показывает довольно высокую связность RFC, что может привести к усложнению в тестировании и исполнении кода, а также к рекурсивному вызову. Код необходимо упростить.

**Набор метрик Лоренца и Кидда**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CS** | | | **NOO** | | **NOA** | | **SI** | **OSavg** | | **OCavg** | | | **NPavg** | **NSS** | **NKC** | **NSUB** |
| Suspicion | 0 | | | 0 | | 0 | | 0 | 0 | |  | | | 1,5 |  |  |  |
| BotGameGlient | 6 | | | 3 | | 0 | | 0,125 | 1 | |  | | | 1 |  |  |  |
| GameClient | 0 | | | 0 | | 0 | | 0 | 0,5 | |  | | | 0,833333333 |  |  |  |
| ClientConnection | 1 | | | 0 | | 0 | | 0 | 0,714285714 | |  | | | 0,857142857 |  |  |  |
| ClientConnectionStrategy | 1 | | | 0 | | 0 | | 0 | 0 | |  | | | 0,857142857 |  |  |  |
| LocalClientConnectionStrategy | 2 | | | 5 | | 0 | | 0,208333333 | 0,714285714 | |  | | | 1 |  |  |  |
| TCPClientConnectionStrategy | 4 | | | 6 | | 1 | | 0,25 | 1,125 | |  | | | 1 |  |  |  |
| ServerConnection | 1 | | | 0 | | 0 | | 0 | 0 | |  | | | 0,857142857 |  |  |  |
| LocalServerConnection | 1 | | | 5 | | 0 | | 0,208333333 | 0,625 | |  | | | 0,75 |  |  |  |
| TCPServerConnection | 4 | | | 6 | | 1 | | 0,25 | 1,714285714 | |  | | | 1 |  |  |  |
| GameServer | 11 | | | 0 | | 0 | | 0 | 4,125 | |  | | | 1,25 |  |  |  |
| AbstractGameBuilder | 5 | | | 0 | | 0 | | 0 | 0 | |  | | | 0,8 |  |  |  |
| GameBuilder | 6 | | | 5 | | 1 | | 0,208333333 | 1,166666667 | |  | | | 0,833333333 |  |  |  |
| Field | 0 | | | 0 | | 0 | | 0 | 0 | |  | | | 0,333333333 |  |  |  |
| Game | 0 | | | 0 | | 0 | | 0 | #ДЕЛ/0! | |  | | | #ДЕЛ/0! |  |  |  |
| Ship | 0 | | | 0 | | 0 | | 0 | 0 | |  | | | 4 |  |  |  |
| MoveableObjectAdapter | 6 | | | 0 | | 0 | | 0 | 0,142857143 | |  | | | 0,714285714 |  |  |  |
| MoveableObjectShip | 7 | | | 1 | | 0 | | 0,041666667 | 0,142857143 | |  | | | 0,571428571 |  |  |  |
| FieldStyle | 9 | | | 0 | | 0 | | 0 | 0 | |  | | | 0,7 |  |  |  |
| FieldWidget | 4 | | | 0 | | 6 | | 0 | 1,571428571 | |  | | | 0,857142857 |  |  |  |
| FieldWidgetEdit | 8 | | | 3 | | 0 | | 0,25 | 0,8 | |  | | | 0,5 |  |  |  |
| FieldWidgetPlayEnemy | 6 | | | 1 | | 0 | | 0,083333333 | 0,125 | |  | | | 0,5 |  |  |  |
| GameWindow | 8 | | | 5 | | 1 | | 0,208333333 | 2,125 | |  | | | 0,75 |  |  |  |
| InitGame | 11 | | | 1 | | 5 | | 0,041666667 | 2,777777778 | |  | | | 0,222222222 |  |  |  |
|  | |  |  | |  | |  | | |  | | 228 |  | | 1 | 0,333333 | 4 |

Метрика CS соответствует рекомендуемым, NOO иногда превышает значение – необходимо упростить классы. Параметр NOA лишь в единственном классе выходит за пределы. SI в целом находится в пределах допустимого, однако иногда превышает. OSavg в пределах рекомендуемого. NPavg превышает норму в некоторых случаях – необходимо уменьшить количество параметров. NSS – одна запись (курсовая работа). NKS соответствует норме, NSUB также соответствует норме.

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