**Российский химико-технологический университет**

**имени Д.И.Менделеева**

**Факультет информационных технологий и управления**

**Отдел дополнительной профессиональной подготовки**

Название предмета: \_\_\_Программирование на языке Java\_\_\_\_\_\_

Преподаватели: \_\_Женса А. В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_Сыромятников А. И.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| № Задания | 1 | 2 | Оценка |
| Балл |  |  |  |

(подпись преподавателя)

**Проект по курсу «Основы программирования Java»**

**Сыромятникова Алексея, группа КС-40**

Описание.

Проект представляет собой игру под названием «Жизнь» («Conway’s Game of Life»), которая имитирует клеточный автомат, придуманный английским математиком Джоном Конвеем в 1970 году.

Игровое поле реализовано с помощью класса JTable и имеет размерность 70х70 клеток. Каждая клетка представляет ячейку, которая может быть «жива» или «мертва» (соответствует белому или черному цвету клетки). В приложении имеются кнопки JButton «Start game», «Stop game», «One step», «Clean cells», а также переключатели JCheckBox «Unlimited borders» и «Drawing mode».

* Start game – запускает игру
* Stop game – останавливает игру
* One step – производит одну итерацию игры (одна перезапись всех ячеек и их перерисовка)
* Clean cells – очищает поле (устанавливает все ячейки в состояние «мертва»)
* Unlimited borders – включает режим «бесконечных границ». Превращает квадратное поле в эмуляцию поверхности тора посредством соединения левой границы с правой и верхней с нижней, но отображается все равно в виде квадрата (по умолчанию отключён).
* Drawing mode – режим позволяющий «рисовать» курсором мышки вместо точечного заполнения клеток (по умолчанию отключён).

В процессе игры все кнопки и переключатели блокируются, кроме кнопки «Stop game». Соответственно при окончании игры по условию или при искусственной остановке кнопки и переключатели разблокируются обратно, а кнопка «Stop game» наоборот блокируется.

Игра заканчивается по условию в тот момент, когда все ячейки мертвы или если на очередной итерации ячейки никак не изменились, то есть образовалась стабильная конфигурация.

Правила игры:

1. Если текущая ячейка жива и имеет две или три живые прилегающие ячейки, то на следующей итерации ячейка остаётся живой. В ином случае ячейка умирает на следующей итерации от переполнения или недостатка соседей.
2. Если текущая ячейка мертва и имеет три живые прилегающие ячейки, то эта ячейка «оживает» на следующей итерации, иначе ячейка остаётся мёртвой.
3. В процессе игры игрок является наблюдателем и не может влиять на её ход. Он может только задать начальную конфигурацию игрового поля и задать режим игры.

Структура и связь классов.

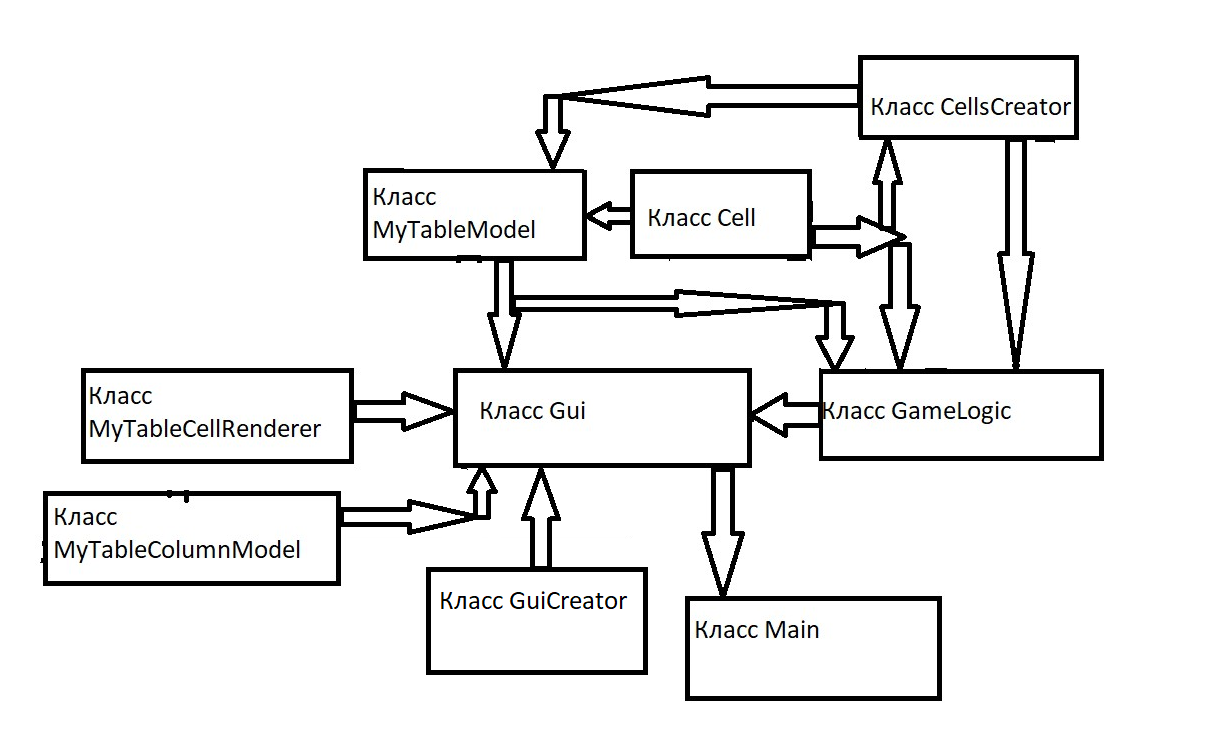


Рисунок 1. Связь классов.

* Класс Cell – имитирует ячейку
* CellsCreator – имеет статические методы, которые инициализируют матрицу ячеек типа Cell. Используется в GameLogic и MyTableModel.
* MyTableCellRenderer – является классом-рисовальщиком. Перерисовывает ячейки при их изменении в MyTableModel, в зависимости от их состояния.
* MyTableColumnModel – содержит модель столбцов игрового поля. Необходим для корректного отображение игровой сетки.
* MyTableModel – содержит в себе модель данных, которую использует игровое поле для отрисовки ячеек, имеет в себе двумерный массив ячеек типа Cell.
* Класс GameLogic – содержит логику игры, т.е. правила, алгоритмы и все данные и манипуляции с ними, необходимые для работы игры.
* Класс Gui – интерфейс игры. Содержит в себе модели для отображения сетки, рендер, логику игры, а также все необходимые кнопки и переключатели с обработкой событий.
* GuiCreator – инициализирует необходимые элементы интерфейса
* Main – инициализирует экземпляр класса Gui и все.

Здесь приведён листинг основных классов, в которых создаются экземпляры всех остальных классов.

Весь листинг можно просмотреть по адресу - <https://github.com/exalibyr/Java/tree/master/GameOfLife>

public class Main {

public static void main(String[] args) {

new Gui();

}

}

public class Gui extends JFrame{

private static final long serialVersionUID = 1L;

Gui(){

super("Game of life");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

MyTableModel gameFieldData = new MyTableModel();

GuiCreator guiCreator = new GuiCreator();

JTable gameFieldTable = guiCreator.initGameFieldTable(gameFieldData);

GameLogic gameLogic = new GameLogic(gameFieldTable, gameFieldData);

JButton stopGameButton = guiCreator.initStopGameButton();

JButton oneStepButton = guiCreator.initOneStepButton();

JButton startGameButton = guiCreator.initStartGameButton();

JButton cleanCellsButton = guiCreator.initCleanCellsButton();

JLabel infoLabel = guiCreator.initInfoLabel();

JCheckBox unlimitedBordersCheckBox = new JCheckBox("Unlimited borders", false);

JCheckBox drawingModeCheckBox = new JCheckBox("Drawing mode", false);

JPanel panel = new JPanel();

JPanel checkBoxesPanel = guiCreator.initCheckBoxesPanel();

checkBoxesPanel.add(unlimitedBordersCheckBox);

checkBoxesPanel.add(drawingModeCheckBox);

panel.add(gameFieldTable);

panel.add(cleanCellsButton);

panel.add(startGameButton);

panel.add(stopGameButton);

panel.add(oneStepButton);

panel.add(infoLabel);

panel.setBackground(Color.ORANGE);

getContentPane().add(BorderLayout.WEST, panel);

getContentPane().add(checkBoxesPanel);

setPreferredSize(new Dimension(1920, 1080));

pack();

setVisible(true);

drawingModeCheckBox.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

if(gameLogic.isDrawingMode()){

gameLogic.setDrawingMode(false);

drawingModeCheckBox.setBorderPaintedFlat(false);

}

else{

gameLogic.setDrawingMode(true);

drawingModeCheckBox.setBorderPaintedFlat(true);

}

}

});

unlimitedBordersCheckBox.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

if(gameLogic.isUnlimitedBorders()){

gameLogic.setUnlimitedBorders(false);

unlimitedBordersCheckBox.setBorderPaintedFlat(false);

}

else{

gameLogic.setUnlimitedBorders(true);

unlimitedBordersCheckBox.setBorderPaintedFlat(true);

}

}

});

gameFieldTable.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {

gameLogic.setMouseReleased();

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

if(gameLogic.isGameRunning()){

infoLabel.setText("Stop the game first!");

}

else{

if(gameLogic.isDrawingMode()){

gameLogic.press();

}

else{

int row = e.getY() / gameFieldTable.getRowHeight();

int column = e.getX() / gameFieldTable.getColumnModel().getTotalColumnWidth();

gameLogic.changeCellState(row, column);

}

}

}

@Override

public void mouseExited(MouseEvent e) { }

@Override

public void mouseEntered(MouseEvent e) { }

@Override

public void mouseClicked(MouseEvent e) { }

});

stopGameButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

gameLogic.stopGame();

startGameButton.setEnabled(true);

cleanCellsButton.setEnabled(true);

oneStepButton.setEnabled(true);

unlimitedBordersCheckBox.setEnabled(true);

drawingModeCheckBox.setEnabled(true);

stopGameButton.setEnabled(false);

infoLabel.setText("Stopped!");

}

});

startGameButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

if(gameLogic.isGameFieldEmpty()){

infoLabel.setText("Cells are empty!");

}

else {

infoLabel.setText("");

gameLogic.startGame(infoLabel, startGameButton, stopGameButton, oneStepButton,

cleanCellsButton, unlimitedBordersCheckBox, drawingModeCheckBox);

}

}

});

cleanCellsButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if(!gameLogic.isGameFieldEmpty()){

gameLogic.cleanCells();

infoLabel.setText("Cleaned!");

}

}

});

oneStepButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if(gameLogic.isGameFieldEmpty()){

infoLabel.setText("Cells are empty!");

}

else {

gameLogic.performOneStep();

infoLabel.setText("Performed!");

}

}

});

}

}