

Java Development Kit

Урок 1

Графические интерфейсы



GUI: графический интерфейс пользователя





На предыдущем курсе





- 📌 процедурное программирование;
- 📌 ООП;
- 📌 исключения;
- 📌 устройство языка Java;
- 📌 устройство платформы для создания и запуска приложений на JVM-языках;
- 📌 базовые средства ввода-вывода;
- 📌 базовые терминальные приложения;
- 📌 алгоритмические задачи, не требующие сложных программных решений.





Что будет на уроке сегодня

Интерфейс — это чрезвычайно важно для простых пользователей

-  создание окна;
-  менеджеры размещений;
-  элементы графического интерфейса;
-  обработчики событий.





Почему именно Swing?

- ✓ Поможет лучше понять ООП;
- ✓ Работа композиции из объектов;
- ✓ Обмен информацией между объектами;
- ✓ Ссылочная природа данных в Java;
- ✓ Запоминание базовых взаимосвязей объектов;
- ✓ Без искусственных примеров.
- Разработчики Swing платят мне за рекламу;
- Это модный и современный фреймворк;
- Он пригодится любому программисту на джава;

Почему не JavaFX? **Перестал быть стандартным начиная с Java 9**

Почему не LibGDX? **Потому что под капотом всё равно Swing.**

IntelliJ IDEA — Написана на Swing



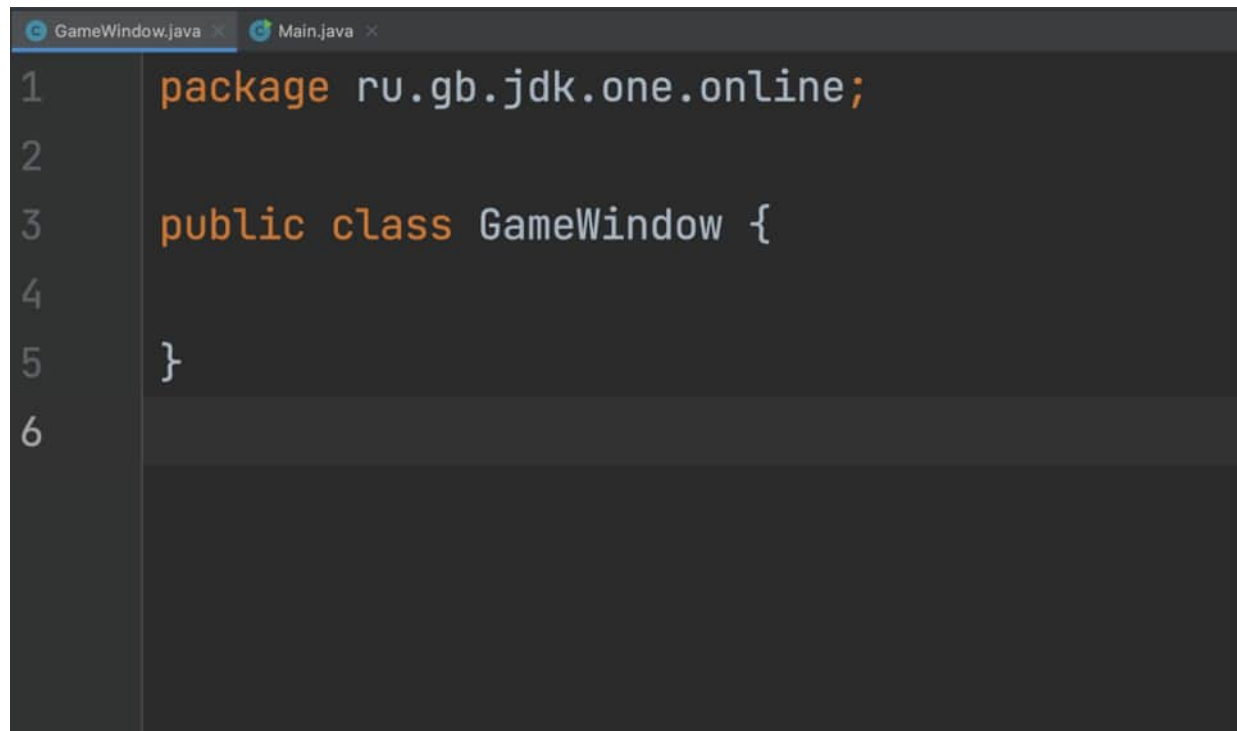


Окно JFrame





Подготовка класса главного окна



```
1 package ru.gb.jdk.one.online;
2
3 public class GameWindow {
4
5 }
6
```



Подготовка класса главного окна

```
1 package ru.gb.jdk.one.online;
2
3 import javax.swing.*;
4
5 public class GameWindow extends JFrame {
6
7 }
8 |
```



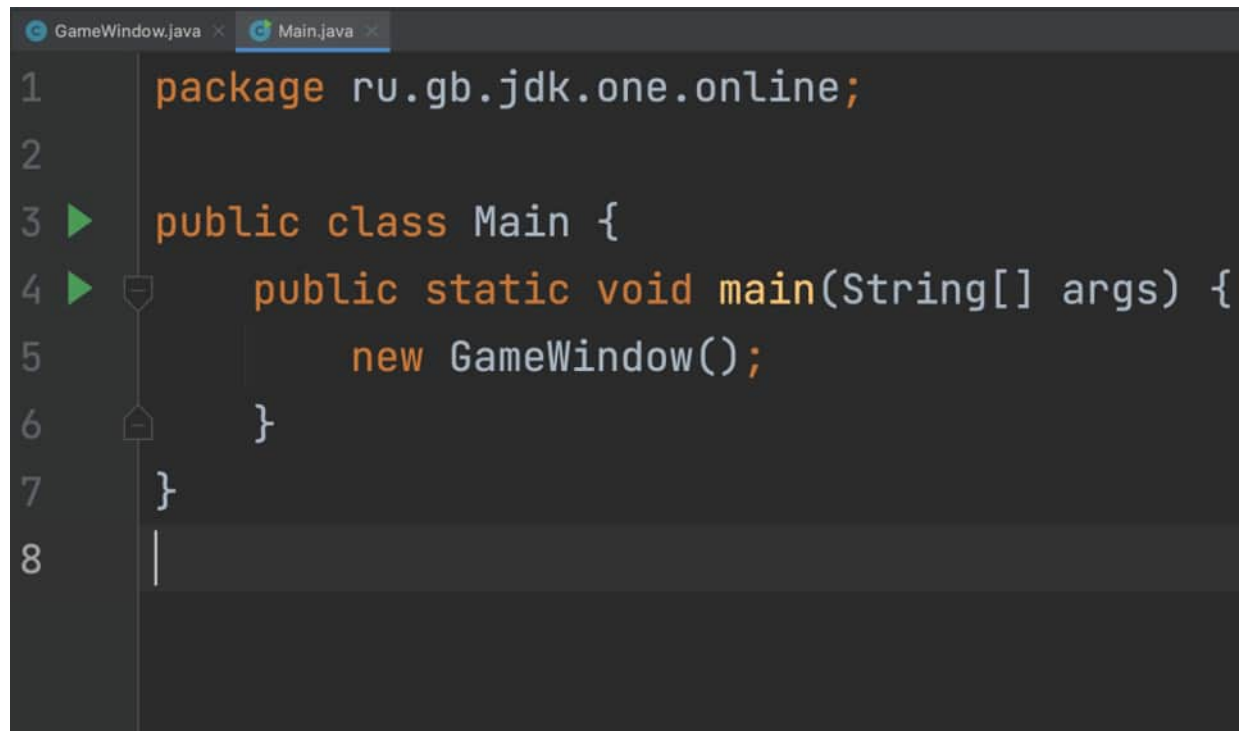

Подготовка класса главного окна

```
GameWindow.java x Main.java x
1 package ru.gb.jdk.one.online;
2
3 import javax.swing.*;
4
5 public class GameWindow extends JFrame {
6     GameWindow() {
7         |
8     }
9 }
10
```





Подготовка класса главного окна



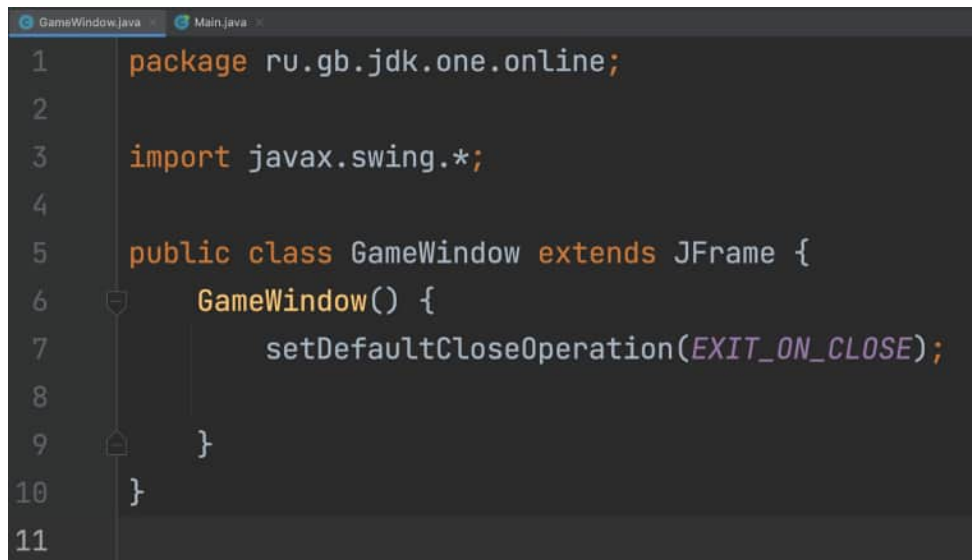
```
1 package ru.gb.jdk.one.online;  
2  
3 public class Main {  
4     public static void main(String[] args) {  
5         new GameWindow();  
6     }  
7 }  
8 |
```



Базовые свойства окон

Все окна по-умолчанию — невидимые.

Нажатие на крестик, по-умолчанию, делает окно невидимым, а не завершает программу.



```
1 package ru.gb.jdk.one.online;
2
3 import javax.swing.*;
4
5 public class GameWindow extends JFrame {
6     GameWindow() {
7         setDefaultCloseOperation(EXIT_ON_CLOSE);
8     }
9 }
10
11
```

Ответьте на вопрос сообщением в чат

Вопрос:

Почему стало возможным просто вызывать методы, которые никогда не были написаны в приложении из конструктора окна, даже не обращаясь ни к каким объектам через точку?

- из-за устройства фреймворка Swing
- из-за наследования от JFrame
- из-за импорта классов Swing

```
5 public class GameWindow extends JFrame {  
6     GameWindow() {  
7         setDefaultCloseOperation(EXIT_ON_CLOSE);  
8     }
```



Конструктор окна

```
5 public class GameWindow extends JFrame {
6     private static final int WINDOW_HEIGHT = 555;
7     private static final int WINDOW_WIDTH = 507;
8     private static final int WINDOW_POSX = 800;
9     private static final int WINDOW_POSY = 300;
10
11     GameWindow() {
12         setDefaultCloseOperation(EXIT_ON_CLOSE);
13     }
14 }
15
16
17
18
```





Конструктор окна

```
5 public class GameWindow extends JFrame {
6     private static final int WINDOW_HEIGHT = 555;
7     private static final int WINDOW_WIDTH = 507;
8     private static final int WINDOW_POSX = 800;
9     private static final int WINDOW_POSY = 300;
10
11     GameWindow() {
12         setDefaultCloseOperation(EXIT_ON_CLOSE);
13         setLocation(WINDOW_POSX, WINDOW_POSY);
14         setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
15
16         setVisible(true);
17     }
18 }
```



Конструктор окна

```
6 private static final int WINDOW_HEIGHT = 555;  
7 private static final int WINDOW_WIDTH = 507;  
8 private static final int WINDOW_POSX = 800;  
9 private static final int WINDOW_POSY = 300;  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19
```

The code defines a constructor for a window with the following parameters:

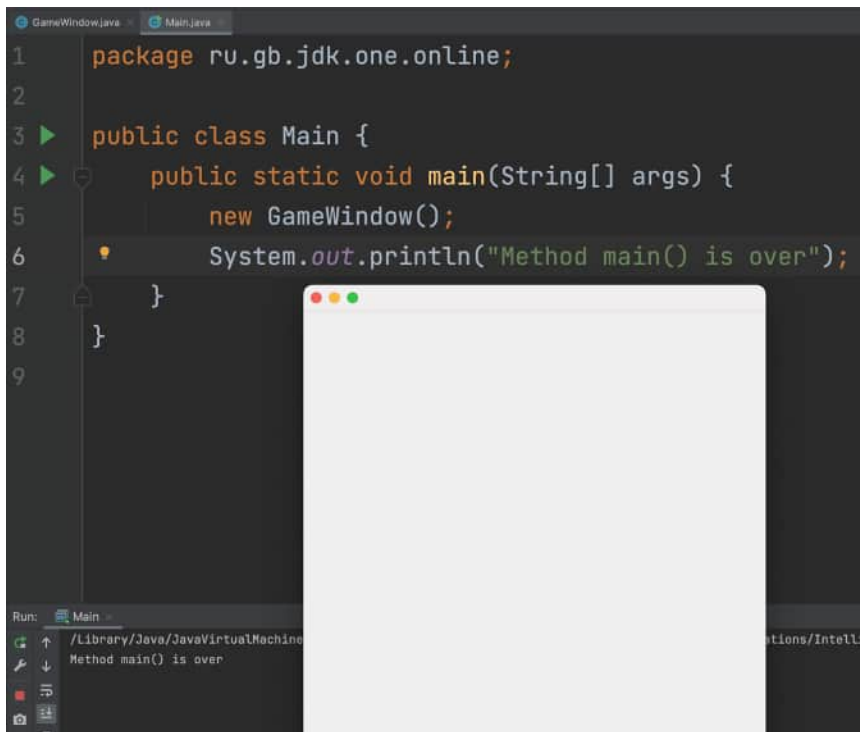
- `WINDOW_HEIGHT`: 555
- `WINDOW_WIDTH`: 507
- `WINDOW_POSX`: 800
- `WINDOW_POSY`: 300

The constructor method is named `GameWindow()` and includes the following calls:

- `setDefaultLookAndFeel()`
- `setLocation()`
- `setSize()`
- `setVisible()`

A preview of the window is shown, displaying a white rectangular area with a standard macOS-style title bar (red, yellow, and green buttons).

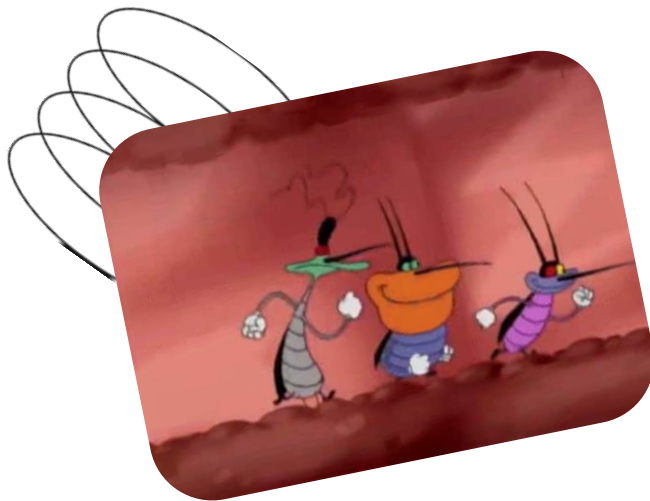
Многопоточность



The screenshot shows an IDE with two tabs: 'GameWindow.java' and 'Main.java'. The 'Main.java' tab is active, displaying the following code:

```
1 package ru.gb.jdk.one.online;
2
3 public class Main {
4     public static void main(String[] args) {
5         new GameWindow();
6         System.out.println("Method main() is over");
7     }
8 }
9
```

Below the code editor, a small window titled 'Main' is open, showing a blank white area. At the bottom of the IDE, a 'Run' console shows the output: 'Method main() is over'.



Ответьте на вопрос сообщением в чат

Вопросы:

1. Чтобы создать пустое окно в программе нужно
 1. импортировать библиотеку Swing
 2. создать класс MainWindow
 3. создать класс-наследник JFrame
2. Свойства окна, такие как размер и заголовок возможно задать
 1. написанием методов в классе-наследнике
 2. вызовом методов в конструкторе
 3. созданием констант в первых строках класса





Компоненты и менеджеры размещений





Компонент “Кнопка”

```
GameWindow.java  Main.java
9      private static final int WINDOW_POSY = 300;
10
11      GameWindow() {
12          setDefaultCloseOperation(EXIT_ON_CLOSE);
13          setLocation(WINDOW_POSX, WINDOW_POSY);
14          setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
15          setTitle("TicTacToe");
16          setResizable(false);
17
18          setVisible(true);
19      }
20  }
21  |
```



Компонент “Кнопка”

```
GameWindow.java Main.java
9      private static final int WINDOW_POSY = 300;
10
11      JButton btnStart = new JButton("New Game");
12      JButton btnExit = new JButton("Exit");
13
14      GameWindow() {
15          setDefaultCloseOperation(EXIT_ON_CLOSE);
16          setLocation(WINDOW_POSX, WINDOW_POSY);
17          setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
18          setTitle("TicTacToe");
19          setResizable(false);
20
21          setVisible(true);
22      }
23  }
24
```



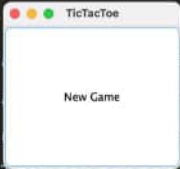
Компонент “Кнопка”

```
GameWindow.java Main.java
9      private static final int WINDOW_POSY = 300;
10
11      JButton btnStart = new JButton("New Game");
12      JButton btnExit = new JButton("Exit");
13
14      GameWindow() {
15          setDefaultCloseOperation(EXIT_ON_CLOSE);
16          setLocation(WINDOW_POSX, WINDOW_POSY);
17          setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
18          setTitle("TicTacToe");
19          setResizable(false);
20          add(btnStart);
21
22          setVisible(true);
23      }
24  }
```

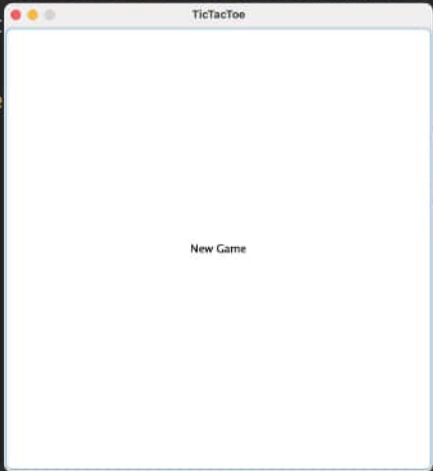


Компонент “Кнопка”

```
GameWindow.java  Main.java
9      private static final int WINDOW_POSY = 300;
10
11      JButton btnStart = new JButton("New Game");
12      JButton btnExit = new JButton("Exit");
13
14      GameWindow() {
15          setDefaultCloseOperation(EXIT_ON_CLOSE);
16          setLocation(0, WINDOW_POSY);
17          setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
18          setTitle("TicTacToe");
19          // setResizable(false);
20          add(btnStart);
21
22          setVisible(true);
23      }
24  }
```

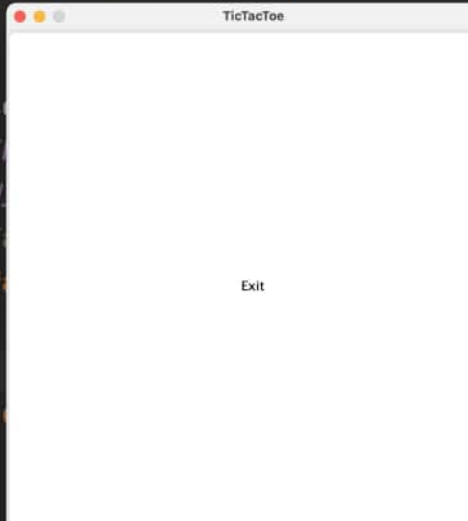


```
GameWindow.java  Main.java
9      private static final int WINDOW_POSY = 300;
10
11      JButton btnStart = new JButton("New Game");
12      JButton btnExit = new JButton("Exit");
13
14      GameWindow() {
15          setDefaultCloseOperation(EXIT_ON_CLOSE);
16          setLocation(0, WINDOW_POSY);
17          setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
18          setTitle("TicTacToe");
19          // setResizable(false);
20          add(btnStart);
21
22          setVisible(true);
23      }
24  }
```



Проблема размещения двух компонентов

```
GameWindow.java  Main.java
9      private static final int WINDOW_POSY = 300;
10
11      JButton btnStart = new JButton("New Game");
12      JButton btnExit = new JButton("Exit");
13
14      GameWindow() {
15          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16          setLocation(WINDOW_POSX, WINDOW_POSY);
17          setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
18          setTitle("TicTacToe");
19          setResizable(false);
20          add(btnStart);
21          add(btnExit);
22          setVisible(true);
23      }
24  }
```





Понятие менеджера размещений

Компоновщик — это специальный объект, который помещается на некоторые (наследники `RootPaneContainer`) компоненты и осуществляет автоматическую расстановку добавляемых на него компонентов, согласно правилам.

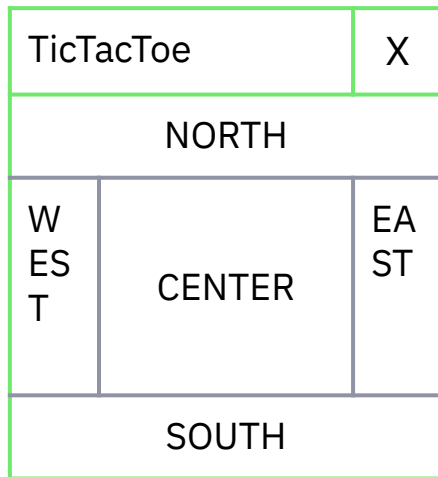
- `BorderLayout` (по умолчанию)
- `BoxLayout`
- `CardLayout`
- `FlowLayout`
- `GridBagLayout`
- `GridLayout`
- `GroupLayout`
- `SpringLayout`





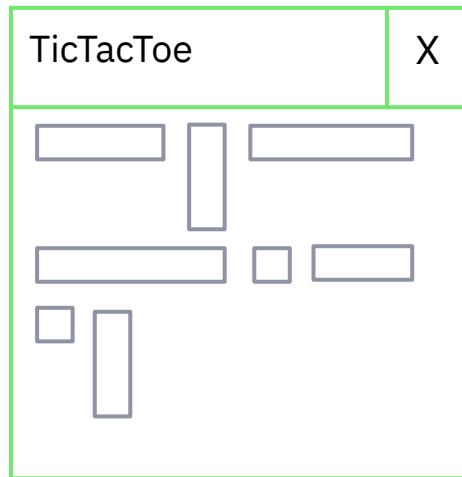
Часто используемые компоновщики

BorderLayout



```
add(btnStart,  
    BorderLayout.SOUTH);
```

FlowLayout



```
setLayout(new  
    FlowLayout());
```

GridLayout

TicTacToe			X
0,0	0,1	0,2	0,3
0	1	2	3
1,0	1,1	1,2	1,3
4	5	6	7

```
new GridLayout(2, 4);
```



JPanel

The screenshot shows an IDE with two tabs: `GameWindow.java` and `Main.java`. The `GameWindow.java` tab is active, displaying the following code:

```
15  GameWindow() {  
16      setDefaultCloseOperation(EXIT_ON_CLOSE);  
17      setLocation(WINDOW_POSX, WINDOW_POSY);  
18      setSize(WINDOW_WIDTH, WINDOW_HEIGHT);  
19      setTitle("TicTacToe");  
20      setResizable(false);  
21  
22      JPanel panBottom = new JPanel(new GridLayout(1, 2));  
23      panBottom.add(btnStart);  
24      panBottom.add(btnExit);  
25      add(panBottom, BorderLayout.SOUTH);  
26      setVisible(true);  
27  }  
28  }  
29
```

To the right of the code editor, a preview window titled "TicTacToe" is shown. It features a large, empty white rectangular area for the game board. At the bottom of the window, there are two buttons: "New Game" and "Exit".




JPanel

```
GameWindow.java x Main.java x Map.java x
1  package ru.gb.jdk.one.online;
2
3  import javax.swing.*;
4  import java.awt.*;
5
6  public class Map extends JPanel {
7      Map() {
8          setBackground(Color.BLACK);
9      }
10 }
11
```



JPanel

```
GameWindow.java Main.java Map.java
15  GameWindow() {
16      setDefaultCloseOperation(EXIT_ON_CLOSE);
17      setLocation(WINDOW_POSX, WINDOW_POSY);
18      setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
19      setTitle("TicTacToe");
20      setResizable(false);
21
22      Map map = new Map();
23
24      JPanel panBottom = new JPanel(new GridLayout(1, 2));
25      panBottom.add(btnStart);
26      panBottom.add(btnExit);
27      add(panBottom, BorderLayout.SOUTH);
28      add(map);
29      setVisible(true);
30  }
```



Игровое поле

```
GameWindow.java Main.java Map.java
1  package ru.gb.jdk.one.online;
2
3  import javax.swing.*;
4  import java.awt.*;
5
6  public class Map extends JPanel {
7      Map() {
8          setBackground(Color.BLACK);
9      }
10
11     void startNewGame(int mode, int fSzX, int fSzY, int wLen) {
12         System.out.printf("Mode: %d;\nSize: x=%d, y=%d;\nWin Length: %d",
13             mode, fSzX, fSzY, wLen);
14     }
15 }
16
```



Подготовка окон, архитектура проекта

```
1 package ru.gb.jdk.one.online;  
2  
3 import javax.swing.*;  
4  
5 public class SettingsWindow extends JFrame {  
6     SettingsWindow(GameWindow gameWindow) {  
7  
8     }  
9 }  
10 |
```



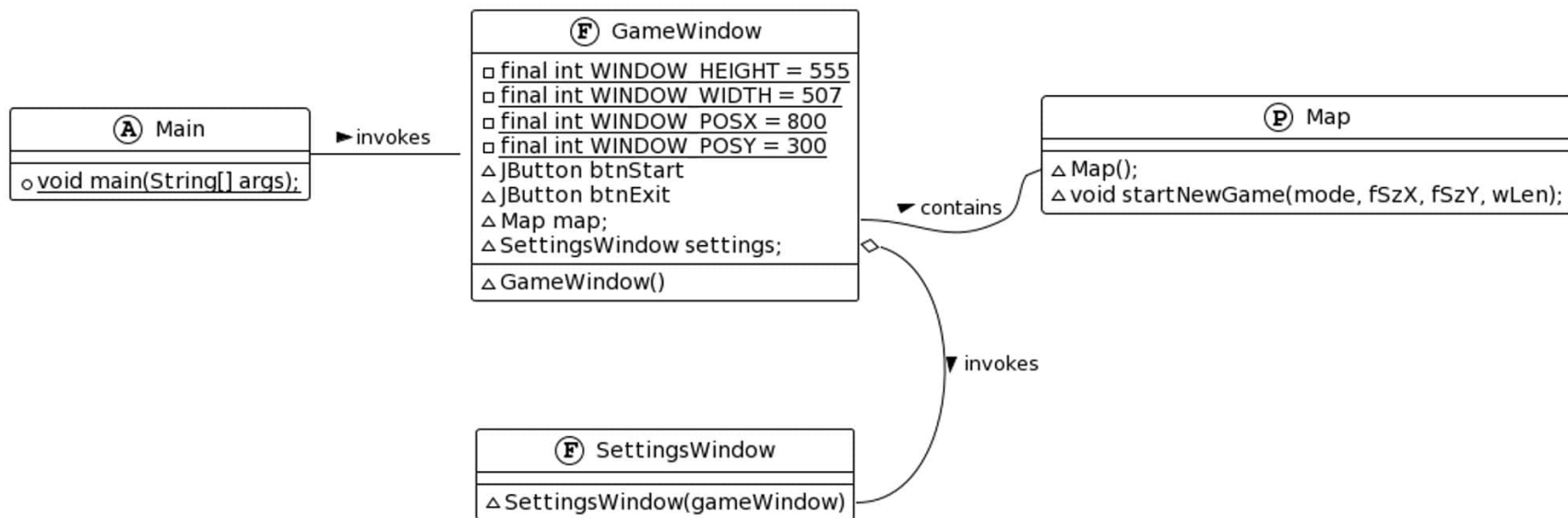
Подготовка окон, архитектура проекта

```
GameWindow.java  Map.java  SettingsWindow.java
15      Map map;
16      SettingsWindow settings;
17
18      GameWindow() {
19          setDefaultCloseOperation(EXIT_ON_CLOSE);
20          setLocation(WINDOW_POSX, WINDOW_POSY);
21          setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
22          setTitle("TicTacToe");
23          setResizable(false);
24
25          map = new Map();
26          settings = new SettingsWindow(this);
27          settings.setVisible(true);
28
```



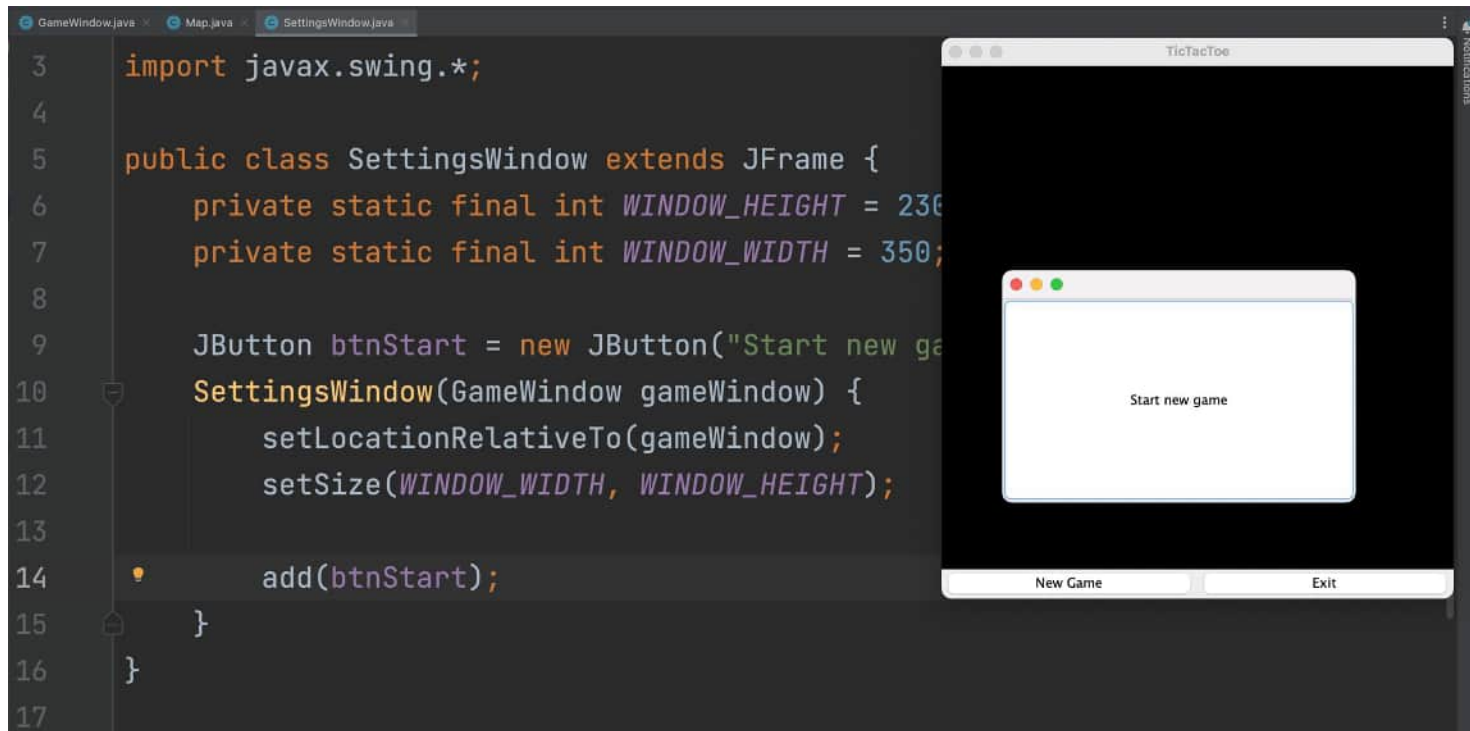


Подготовка окон, архитектура проекта





Окно с настройками проекта



Обработка событий на кнопках

```
28         settings = new SettingsWindow(this);
29
30         btnExit.addActionListener(new ActionL);
31         ActionListener{...} (java...
32         JPanel panBottom = new JPanel(new GridLayout(1, 2));
```



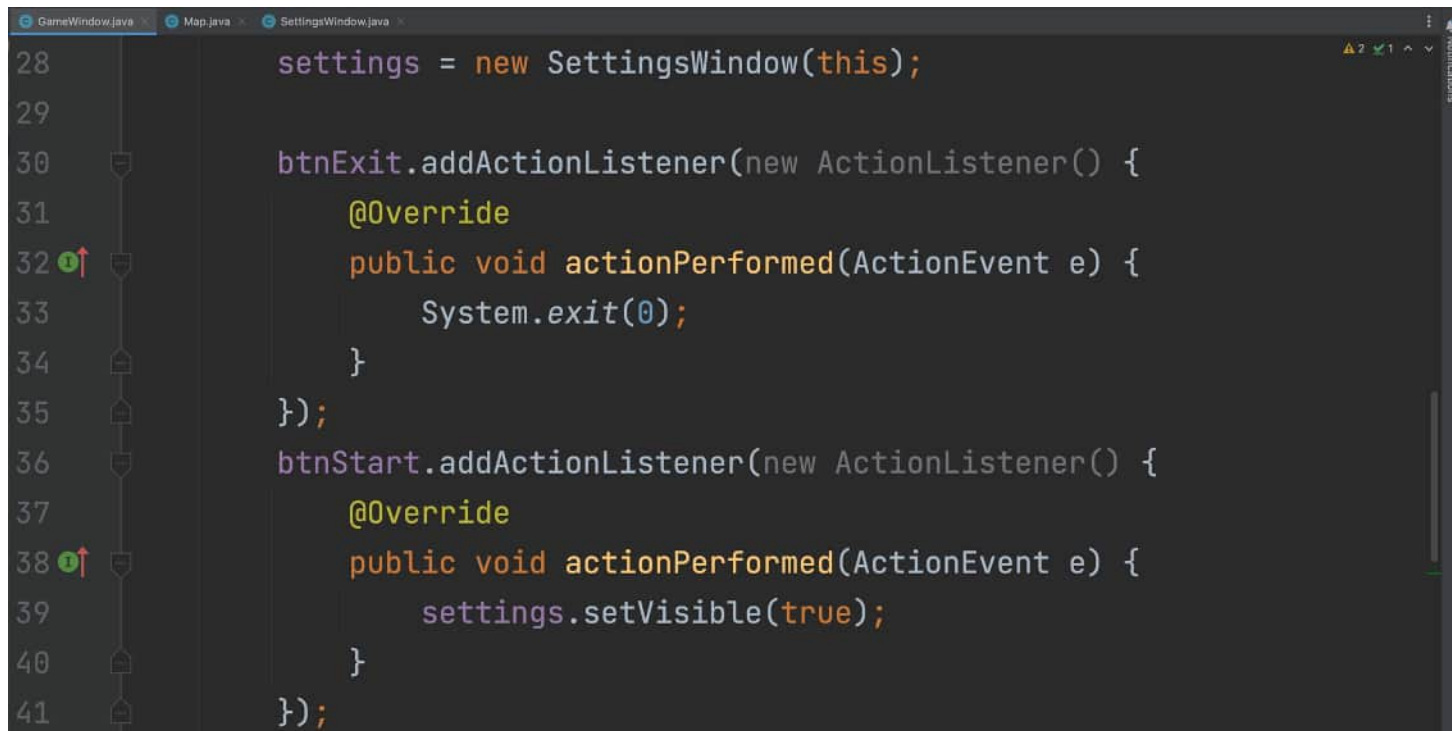


Обработка событий на кнопках

```
30      btnExit.addActionListener(new ActionListener() {  
31          @Override  
32          public void actionPerformed(ActionEvent e) {  
33              |  
34          }  
35      });  
36
```



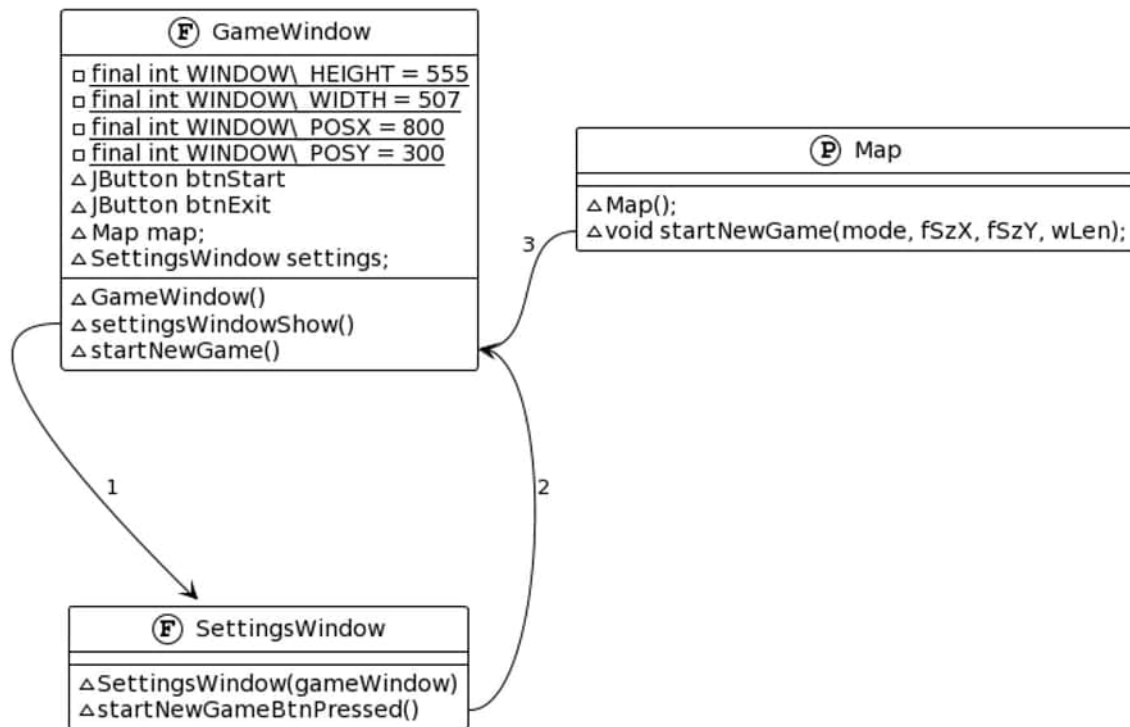
Обработка событий на кнопках



```
28 settings = new SettingsWindow(this);
29
30 btnExit.addActionListener(new ActionListener() {
31     @Override
32     public void actionPerformed(ActionEvent e) {
33         System.exit(0);
34     }
35 });
36 btnStart.addActionListener(new ActionListener() {
37     @Override
38     public void actionPerformed(ActionEvent e) {
39         settings.setVisible(true);
40     }
41 });
```



Устройство программы и цепочка вызовов



Устройство программы и цепочка вызовов

```
11 JButton btnStart = new JButton("Start new game");
12 SettingsWindow(gameWindow) {
13     setLocationRelativeTo(gameWindow);
14     setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
15     btnStart.addActionListener(new ActionListener() {
16         @Override
17         public void actionPerformed(ActionEvent e) {
18             gameWindow.startNewGame(0, 3, 3, 3);
19             setVisible(false);
20         }
21     });
22     add(btnStart);
23 }
24 }
```

Run: Main ×

Mode: 0;
Size: x=3, y=3;
Win Length: 3

```
47 add(map);
48 setVisible(true);
49 }
50
51 void startNewGame(int mode, int fSzX, int fSzY, int wLen) {
52     map.startNewGame(mode, fSzX, fSzY, wLen);
53 }
```

Ответьте на несколько вопросов сообщением в чат

Вопросы:

1. Менеджер размещений — это

1. сотрудник, занимающийся разработкой интерфейса
2. объект, выполняющий расстановку компонентов на окне приложения
3. механизм, проверяющий возможность отображения окна в ОС

2. Экземпляр JPanel позволяет

1. применять комбинации из компоновщиков
2. добавить к интерфейсу больше компонентов
3. создавать группы компонентов
4. всё вышеперечисленное

3. Для выполнения кода по нажатию кнопки на интерфейсе нужно

1. создать обработчик кнопки и вписать код в него
2. переопределить метод нажатия у компонента кнопки
3. использовать специальный класс «слушателя» кнопок





Рисование графики на панели



Рисование самих компонентов

```
6 public class Map extends JPanel {
7     Map() {}
10
11     void startNewGame(int mode, int fSzX, int fSzY, int wLen) {...}
15
16     @Override
17     protected void paintComponent(Graphics g) {
18         super.paintComponent(g);
19         render(g);
20     }
21
22     private void render(Graphics g) {
23
24     }
25 }
26
```





Рисование на компоненте JPanel

The screenshot shows an IDE with three tabs: `GameWindow.java`, `Map.java`, and `SettingsWindow.java`. The `GameWindow.java` tab is active, displaying the following code:

```
10
11 void startNewGame(int mode, int fSzX, int fSzY, int wLen) {...}
15
16 @Override
17 protected void paintComponent(Graphics g) {
18     super.paintComponent(g);
19     render(g);
20 }
21
22 @
23 private void render(Graphics g) {
24     g.setColor(Color.BLACK);
25     g.drawLine(0, 0, 100, 100);
26     int panelWidth = getWidth();
27     int panelHeight = getHeight();
28 }
```

On the right side, there is a preview window titled "TicTacToe". It shows a white square with a single black diagonal line from the top-left corner to the bottom-right corner. At the bottom of the window, there are two buttons: "New Game" and "Exit".



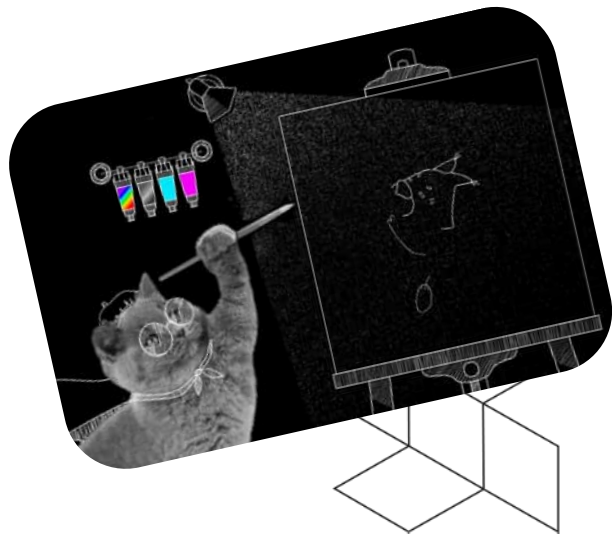
Рисование на компоненте JPanel

```
GameWindow.java  Map.java  SettingsWindow.java
6   public class Map extends JPanel {
7       private int panelWidth;
8       private int panelHeight;
9       private int cellHeight;
10      private int cellWidth;
11
12      Map() {}
13
14
15
16      void startNewGame(int mode, int fSzX, int fSzY, int wLen) {...}
17
18
19
20
21      @Override
22      protected void paintComponent(Graphics g) {...}
23
24
25
26
27      @
28      private void render(Graphics g) {
29          panelWidth = getWidth();
30          panelHeight = getHeight();
31      }
32  }
```

Рисование на компоненте JPanel

```
39 @ private void render(Graphics g) {
40     panelWidth = getWidth();
41     panelHeight = getHeight();
42     cellHeight = panelHeight / 3;
43     cellWidth = panelWidth / 3;
44
45     g.setColor(Color.BLACK);
46     for (int h = 0; h < 3; h++) {
47         int y = h * cellHeight;
48         g.drawLine(0, y, panelWidth, y);
49     }
50     for (int w = 0; w < 3; w++) {
51         int x = w * cellWidth;
52         g.drawLine(x, 0, x, panelHeight);
53     }
54 }

12 Map() {}
15
16 void startNewGame(int mode, int fSzX, int fSzY, int wLen) {
17     System.out.printf("Mode: %d;\nSize: x=%d, y=%d;\nWin Length: %d",
18         mode, fSzX, fSzY, wLen);
19     repaint();
20 }
21 }
```

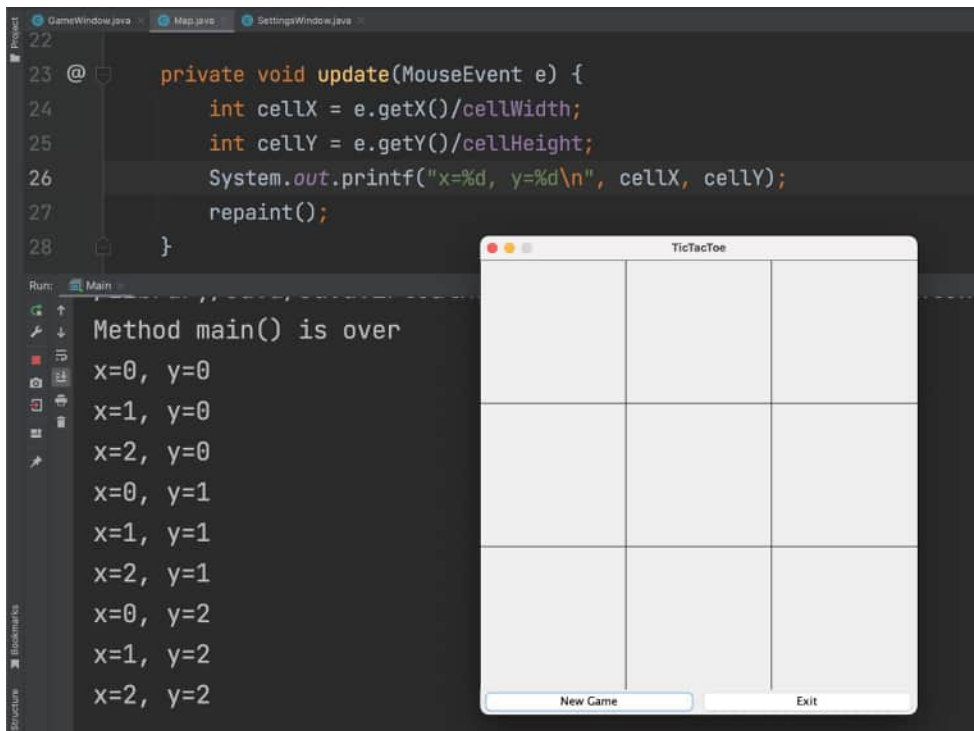




Обработчик клика мышки

```
GameWindow.java x Map.java x SettingsWindow.java x
13
14 Map() {
15     addMouseListener(new MouseAdapter() {
16         @Override
17         public void mouseReleased(MouseEvent e) {
18             update(e);
19         }
20     });
21 }
22
23 private void update(MouseEvent e) {
24     repaint();
25 }
```

Обработчик клика мышки





Логика игры в крестики-нолики





Логика игры

```
Project
  GameWindow.java x
  Map.java x
  SettingsWindow.java x

7  import java.util.Random;
8
9  public class Map extends JPanel {
10      private static final Random RANDOM = new Random();
11      private final int HUMAN_DOT = 1;
12      private final int AI_DOT = 2;
13      private final int EMPTY_DOT = 0;
14      private int fieldSizeY = 3;
15      private int fieldSizeX = 3;
16      private char[][] field;
```




Логика игры

```
Project
  GameWindow.java
  Map.java
  SettingsWindow.java

68  /**
69   * Tic Tac Toe game logic
70   * */
71  private void initMap() {
72      fieldSizeY = 3;
73      fieldSizeX = 3;
74      field = new char[fieldSizeY][fieldSizeX];
75      for (int i = 0; i < fieldSizeY; i++) {
76          for (int j = 0; j < fieldSizeX; j++) {
77              field[i][j] = EMPTY_DOT;
78          }
79      }
80  }
```



Логика игры

```
Project
GameWindow.java
Map.java
SettingsWindow.java

81
82 private boolean isValidCell(int x, int y) {
83     return x >= 0 && x < fieldSizeX && y >= 0 && y < fieldSizeY;
84 }
85
86 private boolean isEmptyCell(int x, int y) {
87     return field[y][x] == EMPTY_DOT;
88 }
89
```





Логика игры

```
89
90     private void aiTurn() {
91         int x, y;
92         do {
93             x = RANDOM.nextInt(fieldSizeX);
94             y = RANDOM.nextInt(fieldSizeY);
95         } while (!isEmptyCell(x, y));
96         field[y][x] = AI_DOT;
97     }
98
```



Логика игры

```
Project
  GameWindow.java
  Map.java
  SettingsWindow.java

70
99 private boolean checkWin(char c) {
100     if (field[0][0]==c && field[0][1]==c && field[0][2]==c) return true;
101     if (field[1][0]==c && field[1][1]==c && field[1][2]==c) return true;
102     if (field[2][0]==c && field[2][1]==c && field[2][2]==c) return true;
103
104     if (field[0][0]==c && field[1][0]==c && field[2][0]==c) return true;
105     if (field[0][1]==c && field[1][1]==c && field[2][1]==c) return true;
106     if (field[0][2]==c && field[1][2]==c && field[2][2]==c) return true;
107
108     if (field[0][0]==c && field[1][1]==c && field[2][2]==c) return true;
109     if (field[0][2]==c && field[1][1]==c && field[2][0]==c) return true;
110     return false;
111 }
```



Логика игры

```
Project
  GameWindow.java x
  Map.java x
  SettingsWindow.java x

113 private boolean isMapFull() {
114     for (int i = 0; i < fieldSizeY; i++) {
115         for (int j = 0; j < fieldSizeX; j++) {
116             if (field[i][j] == EMPTY_DOT) return false;
117         }
118     }
119     return true;
120 }
121
```



Встраивание логики игры, отрисовка

```
23  Map() {...}
31
32  @
33      private void update(MouseEvent e) {
34          int cellX = e.getX()/cellWidth;
35          int cellY = e.getY()/cellHeight;
36          if (!isValidCell(cellX, cellY) || !isEmptyCell(cellX, cellY)) return;
37          field[cellY][cellX] = HUMAN_DOT;
38
39          repaint();
}
```



Встраивание логики игры, отрисовка

```
53 @ private void render(Graphics g) {  
68  
69     for (int y = 0; y < fieldSizeY; y++) {  
70         for (int x = 0; x < fieldSizeX; x++) {  
71             if (field[y][x] == EMPTY_DOT) continue;  
72  
73             if (field[y][x] == HUMAN_DOT) {  
74  
75             } else if (field[y][x] == AI_DOT) {  
76  
77             } else {  
78                 throw new RuntimeException("Unexpected value " + field[y][x] +  
79                     " in cell: x=" + x + " y=" + y);  
80             }  
81         }  
82     }  
83 }  
84
```

Встраивание логики игры, отрисовка

The image shows a code editor with three tabs: `GameWindow.java`, `Map.java`, and `SettingsWindow.java`. The `GameWindow.java` tab is active, displaying the following code:

```
75     if (field[y][x] == HUMAN_DOT) {
76         g.setColor(Color.BLUE);
77         g.fillOval(x * cellWidth + DOT_PADDING,
78                 y * cellHeight + DOT_PADDING,
79                 cellWidth - DOT_PADDING * 2,
80                 cellHeight - DOT_PADDING * 2);
81     } else if (field[y][x] == AI_DOT) {
82         g.setColor(new Color(0xff0000));
83         g.fillOval(x * cellWidth + DOT_PADDING,
84                 y * cellHeight + DOT_PADDING,
85                 cellWidth - DOT_PADDING * 2,
86                 cellHeight - DOT_PADDING * 2);
87     } else {
88         throw new RuntimeException("Unexpected value
89             " in cell: x=" + x + " y=" + y);
90     }
91 }
```

To the right of the code editor is a preview window titled "TicTacToe". It displays a 3x3 grid. The first two rows are filled with blue circles, representing the human player's moves. The third row is empty. At the bottom of the window, there are two buttons: "New Game" and "Exit".



Встраивание логики игры, отрисовка

```
Project
  GameWindow.java
  Map.java
  SettingsWindow.java

8
9  public class Map extends JPanel {
10     private static final Random RANDOM = new Random();
11     private static final int DOT_PADDING = 5;
12
13     private int gameOverType;
14     private static final int STATE_DRAW = 0;
15     private static final int STATE_WIN_HUMAN = 1;
16     private static final int STATE_WIN_AI = 2;
17
18     private static final String MSG_WIN_HUMAN = "Победил игрок!";
19     private static final String MSG_WIN_AI = "Победил компьютер!";
20     private static final String MSG_DRAW = "Ничья!";
```



Последние проверки

```
44 @ private void update(MouseEvent e) {  
50     if (checkEndGame(HUMAN_DOT, STATE_WIN_HUMAN)) return;  
51     aiTurn();  
52     repaint();  
53     if (checkEndGame(AI_DOT, STATE_WIN_AI)) return;  
54 }  
55  
56 private boolean checkEndGame(int dot, int gameOverType) {  
57     if (checkWin(dot)) {  
58         this.gameOverType = gameOverType;  
59         repaint();  
60         return true;  
61     }  
62     if (isMapFull()) {  
63         this.gameOverType = STATE_DRAW;  
64         repaint();  
65         return true;  
66     }  
67     return false;  
68 }
```





Последние проверки

```
84 @ private void render(Graphics g) {  
122     if (isGameOver) showMessageGameOver(g);  
123 }  
124  
125 @ private void showMessageGameOver(Graphics g) {  
126     g.setColor(Color.DARK_GRAY);  
127     g.fillRect(0, 200, getWidth(), 70);  
128     g.setColor(Color.YELLOW);  
129     g.setFont(new Font("Times new roman", Font.BOLD, 48));  
130     switch (gameOverType) {  
131         case STATE_DRAW:  
132             g.drawString(MSG_DRAW, 180, getHeight() / 2); break;  
133         case STATE_WIN_AI:  
134             g.drawString(MSG_WIN_AI, 20, getHeight() / 2); break;  
135         case STATE_WIN_HUMAN:  
136             g.drawString(MSG_WIN_HUMAN, 70, getHeight() / 2); break;  
137         default:  
138             throw new RuntimeException("Unexpected gameOver state: " + gameOverType);  
139     }  
140 }
```

Последние проверки

```
Project
  GameWindow.java
  Map.java
  SettingsWindow.java

35 private boolean isGameOver;
36 private boolean isInitialized;

39 Map() {
40     addMouseListener((MouseListener) this);
46     isInitialized = false;
47 }

48
49 private void update(MouseEvent e) {
50     if (isGameOver || !isInitialized) return;
51     int cellX = e.getX()/cellWidth;
```

```
76 void startNewGame(int mode, int fSzX, int fSzY, int wLen) {
77     System.out.printf("Mode: %d;\nSize: x=%d, y=%d;\nWin Length: %d",
78         mode, fSzX, fSzY, wLen);
79     initMap();
80     isGameOver = false;
81     isInitialized = true;
82     repaint();
83 }

GameWindow.java
Map.java
SettingsWindow.java

91 private void render(Graphics g) {
92     if (!isInitialized) return;

62 private boolean checkEndGame(int dot, int gameOverType) {
63     if (checkWin(dot)) {
64         this.gameOverType = gameOverType;
65         isGameOver = true;
66         repaint();
67         return true;
68     }
69     if (isMapFull()) {
70         this.gameOverType = STATE_DRAW;
71         isGameOver = true;
72         repaint();
73         return true;
74     }
75     return false;
76 }
```





Результаты игры







На этом уроке

Изучили:

-  как создавать окна,
-  менеджеры размещений,
-  элементы графического интерфейса,
-  обработчики событий.

Чтобы продолжить, нужно:

-  многопоточность,
-  программные интерфейсы



Практическое задание

- 📌 Полностью разобраться с кодом.
- 📌 Переделать проверку победы, чтобы она не была реализована просто набором условий.
- 📌 Попробовать переписать логику проверки победы, чтобы она работала для поля 5x5 и количества фигур 4.
- 📌 ** Доработать искусственный интеллект, чтобы он мог примитивно блокировать ходы игрока, и примитивно пытаться выиграть сам.





Научиться можно только
тому, что любишь.

Иоганн Гёте.

