Откроем среду разработки Eclipse для Java и создадим новый проект Java. Для этого в меню выберем: File-New- Java Project (см. рис. 9)

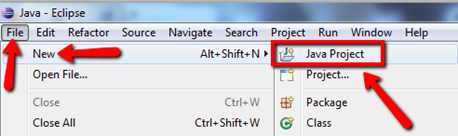


Рис. 9

Укажем название проекта:mb (см. рис. 10) и нажмем кнопку Finish (Готово).

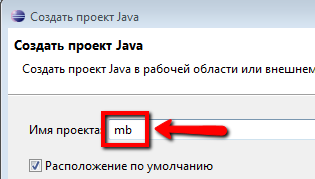


Рис. 10

В списке проектов добавится новый проект mb. Добавим к нему новый класс. Для этого нажмем на имя проекта левой клавиши мыши (выделим). Далее нажмем правую клавишу мыши, откроется дополнительное окно.Выберем New, Class(см. рис. 11).

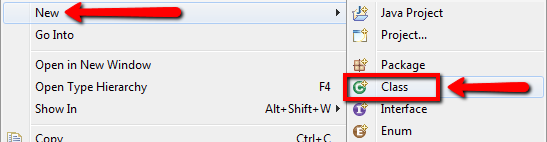


Рис. 11

Это будет главный класс нашего проекта,в нем будет находиться метод main(). Укажем название класса: mb (см. рис. 12) и поставим галочку public static void main(см.рис. 12)

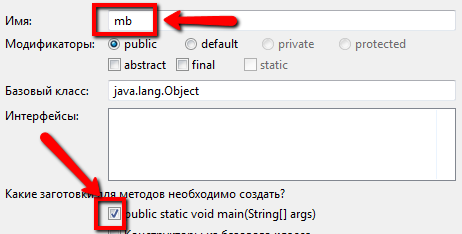


Рис. 12

Далее нажимаем кнопку Finish (Готово). В результате появится программный код нового класса, содержащий метод main() (см. рис. 13)

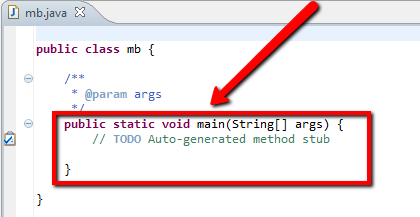


Рис. 13

Класс mb находится в файле mb.java, это главный класс программы, его задача будет заключаться в запуске нашего приложения.

Добавим следующий класс:okno. Для этого снова выделим заголовок проекта mb в списке проектов(см. рис. 14).

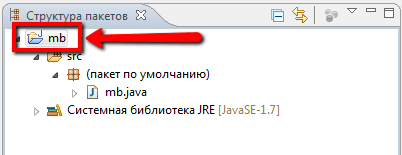


Рис. 14

Важно!!!При добавлении нового класса к проекту – необходимо выделить именно заголовок проекта, а не какую-либо вложенную папку!

Далее нажмем правую клавишу мыши, откроется дополнительное окно. Выберем New-Class(см. рис. 15).

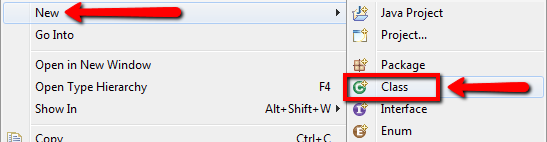


Рис. 15

Это будет класс для окна нашего приложения. Укажем название класса: okno (см. рис. 16) и галочку public static void main(см. рис. 16) в этот раз ставить не нужно!

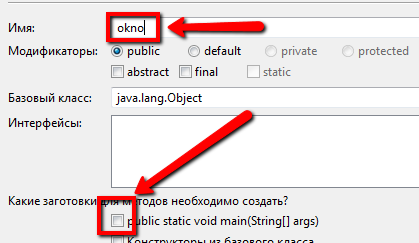


Рис. 16

Галочка public static void main нужна для автоматического создания метода main(), но метод main() уже определен в главном классе mb. Поэтому для всех остальных классов эта галочка не нужна! Аналогично классу okno добавим еще два класса: pole и game. Класс pole для игрового поля, класс game для реализации логики игры с использованием двухмерных массивов.

В результате в нашем проекте будет четыре файла –четыре класса

(см. рис. 17) Обратите внимание, что если все четыре файла добавлены к проекту правильно–они все должны располагаться внутри пакета по умолчанию (см. рис.18).

При разработке игры Морской бой мы не будем использовать визуальный конструктор, потому что интерфейс довольно простой. Элементы интерфейса в виде двух кнопок мы создадим из программного кода.

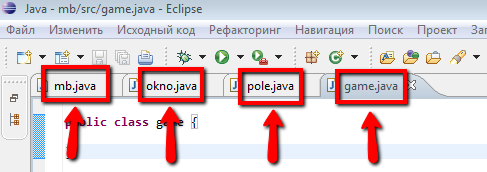


Рис. 17

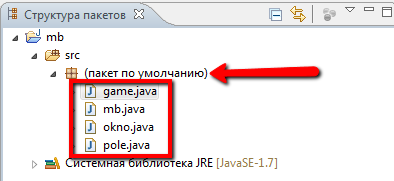


Рис. 18

После реализации первого уровня сложности мы должны получить нарисованное игровое поле и элементы интерфейса (см. рис.19).

Рис. 19

Начнем с класса mb. Перейдем в файл mb.java (см. рис. 20).Реализация класса mb будет самой простой, его задача только в одном запустить наше приложение и создать объект окна.

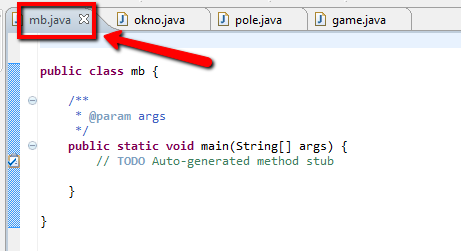


Рис. 20

Программный код этого класса будет выглядеть так:

// Главный класс программы

public class mb

{

// Метод для запуска приложения

public static void main(String[] args)

{

// Создание объекта окна игрового поля

// Класс "okno" находится в файле"okno.java"

okno window = new okno();

}

}

В методе main() создается объект на основании класса okno. Класс okno будет реализован в файле okno.java. Данный программный код класса mb является окончательным для всех уровней сложности. Больше к этому классу мы возвращаться не будем!Это самый маленький и простой класс нашего проекта.

Теперь перейдем в класс okno, в файл okno.java (см. рис. 21).

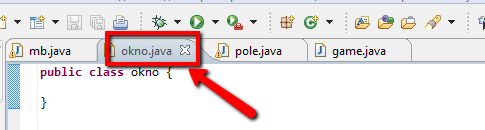


Рис. 21

В этом классе нам необходимо реализовать окно, в котором будет находиться игровое поле. Само игровое поле будет реализовано в классе pole. Похожая схема классов была использована при создании игр "Новогодний дождь" и"Змейка". Программный код класса okno будет выглядеть так:

// Необходимые библиотеки

import java.awt.\*;

import javax.swing.\*;

//Класс окна игрового поля

public class okno extends JFrame

{

// Конструктор класса

public okno()

{

//Создание объекта панели и подключения ее к окну

pole pan = new pole();

Container cont = getContentPane();

cont.add(pan);

//Заголовок окна

setTitle("Игра \"Морской бой\"");

//Границы окна: расположение и размеры

setBounds(0, 0, 900, 600);

//Операция при закрытии окна - завершение приложения

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//Запрет изменения размеров окна

setResizable(false);

//Отображение (показ) окна

setVisible(true);

}

}

Данный программный код класса okno является окончательным для всех уровней сложности. Больше к этому классу мы возвращаться не будем!

Программный код этого класса практически ничем не отличается от программного кода для класса окна в игре"Змейка", есть лишь небольшие отличия:

// Заголовок окна

setTitle("Игра \"Морской бой\"");

Заголовок окна теперь стал Морской бой, и несколько изменились размеры окна в соответствии с техническим заданием:

// Границы окна: расположение и размеры

setBounds(0, 0, 900, 600);

Класс наследуется от класса JFrame и его задача настроить свойства окна и показать его пользователю. Внутри класса okno создается объект игрового поля:

// Создание объекта панели и подключения ее к окну

pole pan = new pole();

Сам класс pole будет реализован в файле pole.java. На этом этапе классы mb и okno реализованы полностью. Так как к ним больше не придется возвращаться –можно закрыть закладки mb.java и okno.java (см. рис. 22).

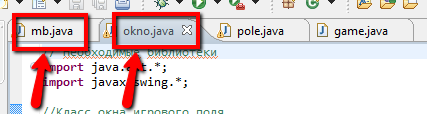


Рис. 22

В результате останутся два открытых файла pole.java и game.java (см. рис.23). В этих двух классах будет основная часть программного кода!

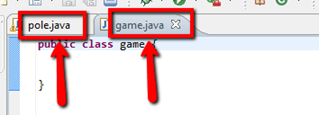


Рис. 23

На первом уровне сложности нам необходимо реализовать рисование игрового поля. При нажатии на кнопку Новая игра, пока, ничего происходить не будет.Это значит, что логику игры нам реализовывать не придется и класс game,пока,останется пустым (см.рис. 24).

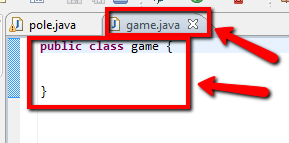


Рис. 24

Все действия для рисования игрового поля и реализации интерфейса будут происходить в классе pole. Перейдем в файл pole.java (см. рис. 25)и приступим к написанию программного кода. Класс pole будет наследоваться от класса JPanel. Он будет представлять собой прямоугольную область,которая вложена в рамку окна. Первое, что мы сделаем –добавим необходимые библиотеки. Здесь нам понадобятся библиотеки для элементов интерфейса пользователя, рисования,обработки событий, работы с файлами,загрузки изображений.

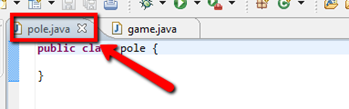


Рис. 25

Все они нам знакомы,добавим их в самом верху класса:

// Для обработки событий

import java.awt.event.\*;

// Для работы с окнами

import javax.swing.\*;

// Для работы с графикой

import java.awt.\*;

// Для работы с изображениями

import javax.imageio.\*;

// Для работы с файлами

import java.io.\*;

Теперь выполним наследование от класса JPanel:

//Класс панели игрового поля

public class pole extends JPanel

{

}

При реализации класса,прежде всего, необходимо подумать какие свойства класса нам понадобятся. Свойства–это переменные класса:

// Для обработки событий

import java.awt.event.\*;

// Для работы с окнами

import javax.swing.\*;

// Для работы с графикой

import java.awt.\*;

// Для работы с изображениями

import javax.imageio.\*;

// Для работы с файлами

import java.io.\*;

//Класс панели игрового поля

public class pole extends JPanel

{

// Таймера отрисовки и изменения логики игры

c

Добавлен таймер для отрисовки:

//Таймер для отрисовки и изменения логики игры

private Timer tmDraw;

Добавлены семь переменных типа Image–для загрузки семи изображений.

//Изображения, используемые в игре

private Image fon, paluba, ubit, ranen, end1, end2, bomba;

Добавлены две кнопкиJButton–это кнопки Новая игра и Выход:

//Две кнопки

private JButton btn1,btn2;

Прежде чем выполнять рисование игрового поля необходимо выполнить начальные (стартовые) действия. Например, загрузку изображений. Подобные действия выполняются в конструкторе. Под переменными класса добавим конструктор класса и первым делом загрузим изображения в игру:

//Конструктор класса

public pole()

{

//Попытка загрузки всех изображений для игры

try

{

fon = ImageIO.read(new File("c:\\fon.png"));

paluba = ImageIO.read(new File("c:\\paluba.png"));

ranen = ImageIO.read(new File("c:\\ranen.png"));

ubit = ImageIO.read(new File("c:\\ubit.png"));

end1 = ImageIO.read(new File("c:\\end1.png"));

end2 = ImageIO.read(new File("c:\\end2.png"));

bomba = ImageIO.read(new File("c:\\bomba.png"));

}

catch (Exception ex) {}

}

В процессе разработки все изображения находятся в корне диска C:\. При успешной загрузке изображений –они станут доступны в игре. После загрузки изображений создаем,настраиваем и запускаем таймер отрисовки игрового поля. Программный код для таймера будет также внутри конструктора, под загрузкой изображений:

//Создаем, настраиваем и запускаем таймер

//для отрисовки игрового поля

tmDraw = new Timer(50,new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

// Вызываем перерисовку -paintComponent()

repaint();

}

});

tmDraw.start();

Отображение фона, сеток игровых полей из линий, выведение надписей мы будем выполнять в методе paintComponent(). Две кнопки Новая игра и Выход добавим в конструкторе класса под кодом для таймера. Полный код конструктора класса pole будет выглядеть так:

//Конструктор класса

public pole()

{

//Попытка загрузки всех изображений для игры

try

{

fon = ImageIO.read(new File("c:\\fon.png"));

paluba = ImageIO.read(new File("c:\\paluba.png"));

ranen = ImageIO.read(new File("c:\\ranen.png"));

ubit = ImageIO.read(new File("c:\\ubit.png"));

end1 = ImageIO.read(new File("c:\\end1.png"));

end2 = ImageIO.read(new File("c:\\end2.png"));

bomba = ImageIO.read(new File("c:\\bomba.png"));

}

catch (Exception ex) {}

//Создаем, настраиваем и запускаем таймер

//для отрисовки игрового поля

tmDraw = new Timer(50,new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

// Вызываем перерисовку -paintComponent()

repaint();

}

});

tmDraw.start();

// Включаем возможность произвольного размещения

//элементов интерфейса на панели

setLayout(null);

//Создаем кнопку Новая игра

btn1 = new JButton();

btn1.setText("Новая игра");

btn1.setForeground(Color.BLUE);

btn1.setFont(new Font("serif",0,30));

btn1.setBounds(130, 450, 200, 80);

btn1.addActionListener(new ActionListener() {

// Обработчик события при нажатии на кнопку Новая игра

public void actionPerformed(ActionEvent arg0) {

}

});

add(btn1);

//Создаем кнопку Выход

btn2 = new JButton();

btn2.setText("Выход");

btn2.setForeground(Color.RED);

btn2.setFont(new Font("serif",0,30));

btn2.setBounds(530, 450, 200, 80);

btn2.addActionListener(new ActionListener() {

// Обработчик события при нажатии на кнопку Новая игра

public void actionPerformed(ActionEvent arg0) {

// Выход их игры -завершение работы приложения

System.exit(0);

}

});

add(btn2);

}

Если выполнить запуск,то мы увидим следующее (см.рис. 26).

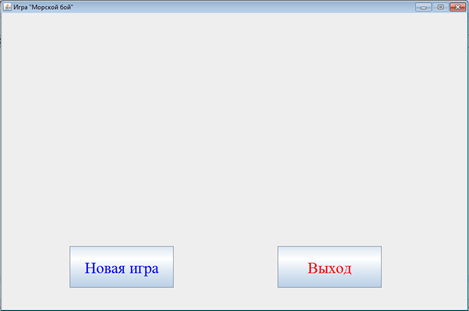


Рис. 26

Добавление кнопок происходит привычным способом. Точно также были добавлены кнопки в игре "Змейка".Обработчик события для кнопки Новая игра на первом уровне сложности ничего не выполняет.

Остальные действия для рисования игрового поля мы выполним, как и обычно, в методе paintComponent(). Добавим его под конструктором класса.

//Выведем изображение фона и две надписи Компьютер, Игрок:

// Метод отрисовки

public void paintComponent(Graphics gr)

{

//Очищение игрового поля

super.paintComponent(gr);

//Отрисовка фона

gr.drawImage(fon,0,0,900,600,null);

//Установка шрифта

gr.setFont(new Font("serif",3,40));

//Установка цвета

gr.setColor(Color.BLUE);

//Выведение надписей

gr.drawString("Компьютер", 150, 50);

gr.drawString("Игрок", 590, 50);

}

Запустим приложение на выполнение и увидим (см. рис.27).

Рис. 27

Выведение текста происходит при помощи метода drawString():

// Установка шрифта

gr.setFont(new Font("serif",3,40));

//Установка цвета

gr.setColor(Color.BLUE);

//Выведение надписей

gr.drawString("Компьютер", 150, 50);

gr.drawString("Игрок", 590, 50);

Перед выведением текста методом drawString() необходимо указать цвет и шрифт выводимой надписи. Координаты для вывода надписей можно для начала указать приблизительно, а потом при запуске смотреть расположение надписи и корректировать координаты в программном коде.

Далее в методе paintComponent() нам предстоит нарисовать сетку игрового поля.Программный код будет выглядеть так:

//Отрисовка сетки игрового поля из синих линий

gr.setColor(Color.BLUE);

for (int i = 0; i <= 10; i++)

{

// Рисование линий сетки игрового поля Компьютера

gr.drawLine(100+i\*30, 100, 100+i\*30, 400);

gr.drawLine(100, 100+i\*30, 400, 100+i\*30);

// Рисование линий сетки игрового поля Игрока

gr.drawLine(500+i\*30, 100, 500+i\*30, 400);

gr.drawLine(500, 100+i\*30, 800, 100+i\*30);

}

Запустим приложение на выполнение и увидим две сетки из линий.Каждое игровое поле имеет десять строк и десять столбцов. Ширина и высота одной клетки составляют 30 пикселей. Для рисования сетки из линий был использован цикл for. В данном случае этот цикл выполняет одиннадцать шагов:

for (int i = 0; i <= 10; i++)

Внутри тела цикла происходит рисование сразу четырех линий.После первого шага цикла нарисуются четыре линии. На первом шагу цикла переменнаяi=0, поэтому произойдет следующее:

// Рисование линий сетки игрового поля Компьютера

gr.drawLine(100+0\*30, 100, 100+0\*30, 400);

gr.drawLine(100, 100+0\*30, 400, 100+0\*30);

// Рисование линий сетки игрового поля Игрока

gr.drawLine(500+0\*30, 100, 500+0\*30, 400);

gr.drawLine(500, 100+0\*30, 800, 100+0\*30);

После второго шага цикла нарисуются уже восемь линий

На втором шагу цикла переменная i=1,значит вместо i подставится значение единица 1. Чтобы создать такой цикл необходимо приблизительно рассчитать начальные и конечные точки линий. При запуске приложения можно наблюдать за положением линий и корректировать координаты. Более простым способом будет поставить себе задачу нарисовать только один вид линий в цикле. И только потом по одному виду линии рисовать следующие.

Чтобы выявить закономерность при рисовании линий в цикле–необходимо нарисовать первую и вторую линию без цикла. И посмотреть какие параметры метода drawLine() изменяются.

Нам осталось в методе paintComponent() выполнить вывод цифр и букв для нумерации строк и столбцов. Для этого необходимо добавить код:

//Установка шрифта

gr.setFont(new Font("serif",0,20));

//Установка цвета

gr.setColor(Color.RED);

//Введение цифр и букв слева и сверху от игровых полей

for (int i = 1; i <= 10; i++)

{

// Вывод цифр

gr.drawString(""+i, 73, 93+i\*30);

gr.drawString(""+i, 473, 93+i\*30);

// Вывод букв

gr.drawString(""+(char)('A'+i-1), 78+i\*30, 93);

gr.drawString(""+(char)('A'+i-1), 478+i\*30, 93);

Запустим приложение на выполнение и увидим цифры и буквы (см. рис.32).

Рис. 32

Процесс выведения цифр и букв напоминает алгоритм рисования сетки из линий, с тем отличием, что вместо рисования линии мы выводим цифру или букву.Для решения этой задачи использован цикл for, в данном случае он выполняет десять шагов:

for (int i = 1; i <= 10; i++)

Такой вызов метода drawString() в цикле позволяет выводить числа от 1до 10:

gr.drawString(""+i, 73, 93+i\*30);

К пустой строке прибавляется значение переменной i.

Для вывода букв был использован метод drawString()в несколько более сложном варианте:

gr.drawString(""+(char)('A'+i-1), 78+i\*30, 93);

Каждый символ имеет свой числовой код. Например, цифра один‘1’имеет код 49. Код каждого символа уникален. Коды цифр и букв идут по порядку.

Из этого следует, что если к коду символа ‘A’прибавить единицу 1, то получится код символа ‘B’.Если к коду символа ‘A’прибавлять переменную i, которая увеличивается на единицу на каждом шагу цикла, то мы будем получать код букв алфавита по порядку, начиная с ‘A’. Остается перевести полученный код в символ и вывести на игровое поле. Для этого использовано преобразование к типу (char):

char ch = (char)(100);

При такой записи код 100 преобразуется в символ с кодом 100. После этого переменная ch станет равна символу ‘d’.

Мы выполнили реализацию игры Морской бой для первого уровня сложности. Полный программный код класса pole выглядит так:

// Для обработки событий

import java.awt.event.\*;

// Для работы с окнами

import javax.swing.\*;

// Для работы с графикой

import java.awt.\*;

// Для работы с изображениями

import javax.imageio.\*;

// Для работы с файлами

import java.io.\*;

//Класс панели игрового поля

public class pole extends JPanel

{

// Таймер отрисовки

private Timer tmDraw;

// Изображения, используемые в игре

private Image fon, paluba, ubit, ranen, end1, end2, bomba;

// Две кнопки

private JButton btn1,btn2;

// Конструктор класса

public pole()

{

//Попытка загрузки всех изображений для игры

try

{

fon = ImageIO.read(new File("c:\\fon.png"));

paluba = ImageIO.read(new File("c:\\paluba.png"));

ranen = ImageIO.read(new File("c:\\ranen.png"));

ubit = ImageIO.read(new File("c:\\ubit.png"));

end1 = ImageIO.read(new File("c:\\end1.png"));

end2 = ImageIO.read(new File("c:\\end2.png"));

bomba = ImageIO.read(new File("c:\\bomba.png"));

}

catch (Exception ex) {}

//Создаем, настраиваем и запускаем таймер

//для отрисовки игрового поля

tmDraw = new Timer(50,new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

// Вызываем перерисовку -paintComponent()

repaint();

}

});

tmDraw.start();

//Включаем возможность произвольного размещения

//элементов интерфейса на панели

setLayout(null);

//Создаем кнопку Новая игра

btn1 = new JButton();

btn1.setText("Новая игра");

btn1.setForeground(Color.BLUE);

btn1.setFont(new Font("serif",0,30));

btn1.setBounds(130, 450, 200, 80);

btn1.addActionListener(new ActionListener() {

//Обработчик события при нажатии на кнопку Новая игра

public void actionPerformed(ActionEvent arg0) {

}

});

add(btn1);

//Создаем кнопку Выход

btn2 = new JButton();

btn2.setText("Выход");

btn2.setForeground(Color.RED);

btn2.setFont(new Font("serif",0,30));

btn2.setBounds(530, 450, 200, 80);

btn2.addActionListener(new ActionListener() {

// Обработчик события при нажатии на кнопку Новая игра

public void actionPerformed(ActionEvent arg0) {

// Выход их игры -завершение работы приложения

System.exit(0);

}

});

add(btn2);

}

// Метод отрисовки

public void paintComponent(Graphics gr)

{

//Очищение игрового поля

super.paintComponent(gr);

//Отрисовка фона

gr.drawImage(fon,0,0,900,600,null);

//Установка шрифта

gr.setFont(new Font("serif",3,40));

//Установка цвета

gr.setColor(Color.BLUE);

//Выведение надписей

gr.drawString("Компьютер", 150, 50);

gr.drawString("Игрок", 590, 50);

//Отрисовка сетки игрового поля из синих линий

gr.setColor(Color.BLUE);

for (int i = 0; i <= 10; i++)

{

// Рисование линий сетки игрового поля Компьютера

gr.drawLine(100+i\*30, 100, 100+i\*30, 400);

gr.drawLine(100, 100+i\*30, 400, 100+i\*30);

// Рисование линий сетки игрового поля Игрока

gr.drawLine(500+i\*30, 100, 500+i\*30, 400);

gr.drawLine(500, 100+i\*30, 800, 100+i\*30);

}

//Установка шрифта

gr.setFont(new Font("serif",0,20));

//Установка цвета

gr.setColor(Color.RED);

//Введение цифр и букв слева и сверху от игровых полей

for (int i = 1; i <= 10; i++)

{

// Вывод цифр

gr.drawString(""+i, 73, 93+i\*30);

gr.drawString(""+i, 473, 93+i\*30);

// Вывод букв

gr.drawString(""+(char)('A'+i-1), 78+i\*30, 93);

gr.drawString(""+(char)('A'+i-1), 478+i\*30, 93);

}

}

}