**Практическая работа 4.4. Реализация второго уровня создания игры змейка.**

Реализация игры Змейка на втором уровне сложности является продолжением первого уровня сложности. Поэтому мы продолжаем дорабатывать уже существующий проект.

Начнем доработку с класса game. Для этого перейдем в файл game.java.

В первую очередь добавим новые свойства класса к уже имеющемуся свойству mas:

// Двумерный массив для хранения игрового поля

public int[][] mas;

// Текущее направление движения

public int napr;

// Координаты головы змейки

private int gX, gY;

// Количество очков

public int kol;

Голова змейки будет перемещаться в четырех возможных направлениях, поэтому понадобится переменная:

// Текущее направление движения

private int napr;

Условимся, что каждое из четырех направлений будет обозначаться цифрой от 0 до 3 (0 –влево, 1 –вверх, 2 –вправо, 3 –вниз). Для хранения позиции головы змейки в массиве введем еще две переменные:

// Координаты головы змейки

private int gX, gY;

Это номер строки в массиве gY и номер элемента в строке gX.

Для хранения набранных очков добавим переменную:

// Количество очков

public int kol;

Далее добавим несколько строчек в метод start(), который выполняет начальный запуск игры:

// Запуск игры (Старт)

public void start()

{

//Заполняем весь массив нулями

for (int i = 0; i < 30; i++) {

for (int j = 0; j < 30; j++) {

mas[i][j] = 0;

}

}

//Начальное направление - влево

napr = 0;

//Количество очков

kol = 0;

//Записываем в переменные координаты головы змейки

gX = gY = 15;

//Создаем начальную змейку с длиной три ячейки

mas[15][15] = 1;

//Формируем первый объект для поедания змейкой

make\_new();

}

Зададим начальное направление движения змейки – влево:

//Начальное направление - влево

napr = 0;

Начальное количество очков обнуляем:

//Количество очков

kol = 0;

Координаты головы змейки (номер строки в массиве и номер элемента в строке массива) устанавливаем 15:

//Записываем в переменные координаты головы змейки

В класс game нужно добавить новый метод peremGolova(),который будет отвечать за перемещение головы змейки. Голова змейки отмечена в массиве значением "единица": 1. Перемещение головы змейки сводится к перемещению единицы в массиве, а на прежнее место следует помещать нулевое значение. Единица будет перемещаться в массиве в зависимости от направления движения головы змейки. При перемещении необходимо учитывать выход за границы игрового поля. В этом случае голова змейки должна появляться с противоположной стороны. Для решения этих задач добавим новый метод:

// Перемещение головы змейки

public void peremGolova()

{

//Сразу удаляем голову змейки из текущей ячейки

mas[gY][gX] = 0;

// Если текущее направление влево

if (napr == 0)

{

if ((gX - 1) >= 0)

gX--;

else

gX = 29;

}

// Если текущее направление вверх

else if (napr == 1)

{

if ((gY - 1) >= 0)

gY--;

else

gY = 29;

}

// Если текущее направление вправо

else if (napr == 2)

{

if ((gX + 1) <= 29)

gX++;

else

gX = 0;

}

// Если текущее направление вниз

else if (napr == 3)

{

if ((gY + 1) <= 29)

gY++;

else

gY = 0;

}

// Если съеден объект

if (mas[gY][gX]==-1)

{

// Генерируем новый объект для поедания

make\_new();

// Увеличиваем количество очков на 10

kol += 10;

}

//Помещаем голову змейки в новую позицию

mas[gY][gX] = 1;

}

Нам известно положение головы змейки в массиве благодаря переменным gY,gX. На текущем месте головы змейки записываем нулевое значение:

// Сразу удаляем голову змейки из текущей ячейки

mas[gY][gX] = 0;

Далее при помощи условной конструкции рассматриваем четыре возможных текущих направления движения змейки:

C

Если текущее значение gX либо -1, либо больше или равно нулю, то мы не выходим за пределы массива и можно смело уменьшить значение переменной gXна единицу. В противном случае, мы уже выходим за границы массива влево, и голова змейки должна появиться с крайней правой стороны.Поэтому помещаем в переменную gX значение 29. Номера строк массива и позиций в строке массива лежат от 0 до 29 включительно!Используя похожую схему,рассматриваем варианты для трех других направлений движения. Очень удобно зарисовать на бумаге массив, проставить номера строк и номера позиций в строке и рассматривать каждую ситуацию на рисунке,глядя на программный код. Тогда понимание станет более ясным.

Проверяем, что голова змейки перемещается в ячейку с объектом для поедания, такой объект отмечен значением "минус единица" в массиве: -1.

// Если съеден объект

}

// Если текущее направление вверх

else if (napr == 1)

{

if ((gY - 1) >= 0)

gY--;

else

gY = 29;

}

// Если текущее направление вправо

else if (napr == 2)

{

if ((gX + 1) <= 29)

gX++;

else

gX = 0;

}

// Если текущее направление вниз

else if (napr == 3)

{

if ((gY + 1) <= 29)

gY++;

else

gY = 0;

}if (mas[gY][gX]==-1)

{

// Генерируем новый объект для поедания

make\_new();

// Увеличиваем количество очков на 10

kol += 10;

}

Если объект съеден, то создаем новый объект для поедания с помощью метода make\_new()и увеличиваем количество очков на10.

В самом конце метода помещаем единицу в новую позицию головы змейки:

// Помещаем голову змейки в новую позицию

mas[gY][gX] = 1;

Обратите внимание, что до этого мы изменили только значения переменных gYили gX, которые отвечают за позицию головы змейки, но значение в массиве устанавливаем в самом конце.

На этом программный код класса game для второго уровня сложности готов и полностью выглядит так:

public class game

{

// Двухмерный массив для хранения игрового поля

public int[][] mas;

// Текущее направление движения

public int napr;

// Координаты головы змейки

private int gX, gY;

// Количество очков

public int kol;

// Конструктор класса

public game()

{

//Создаем новый массив 30x30

mas = new int[30][30];

}

// Генерация нового объекта в случайном месте

private void make\_new()

{

//Глухой (бесконечный цикл)

while(true)

{

// Получаем случайные значения x,y от 0 до 29

int x = (int)(Math.random()\*30);

int y = (int)(Math.random()\*30);

// Если в этом месте массива нулевое значение

// то помещаем туда объект для поедания змейкой

// и прерываем цикл

if (mas[y][x]==0)

{

mas[y][x] = -1;

break;

}

}

}

// Запуск игры (Старт)

public void start()

{

//Заполняем весь массив нулями

for (int i = 0; i < 30; i++) {

for (int j = 0; j < 30; j++) {

mas[i][j] = 0;

}

}

//Начальное направление - влево

napr = 0;

//Количество очков

kol = 0;

//Создаем начальную змейку с длиной три ячейки

mas[15][15] = 1;

//Записываем в переменные координаты головы змейки

gX = gY = 15;

//Формируем первый объект для поедания змейкой

make\_new();

}

// Перемещение головы змейки

public void peremGolova()

{

//Сразу удаляем голову змейки из текущей ячейки

mas[gY][gX] = 0;

// Если текущее направление влево

if (napr == 0)

{

if ((gX - 1) >= 0)

gX--;

else

gX = 29;

// Если съеден объект

if (mas[gY][gX]==-1)

{

// Генерируем новый объект для поедания

make\_new();

// Увеличиваем количество очков на 10

kol += 10;

}

//Помещаем голову змейки в новую позицию

mas[gY][gX] = 1;

}

}

Все необходимые доработки в классе game для второго уровня сложности уже выполнены. В этом классе добавлены новые возможности, определяющие поведение змейки. Все это относится к логике игры.Теперь перейдем во второй файл нашего проекта zmeika.java.

Классы zmeika и myFrame больше дорабатывать не нужно, мы их описали полностью! Они остаются без изменений для любого варианта игры. Все необходимые библиотеки также добавлены! Все это находится в самой верхней части файла zmeika.java и выглядит следующим образом:

// 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.\*;

// Главный класс программы

public class zmeika

{

// Метод запуска приложения

public static void main(String[] args)

{

//Создание объекта окна игрового поля

myFrame okno = new myFrame();

}

}

// Класс окна игрового поля

class myFrame extends JFrame

{

// Конструктор класса

public myFrame()

{

//Создание объекта панели и подключения ее к окну

myPanel pan = new myPanel();

Container cont = getContentPane();

cont.add(pan);

//Заголовок окна

setTitle("Игра \"Змейка\"");

//Границы окна: расположение и размеры

setBounds(0, 0, 800, 650);

//Операция при закрытии окна - завершение приложения

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//Запрет изменения размеров окна

setResizable(false);

//Отображение (показ) окна

setVisible(true);

}

}

Все изменения, которые мы сейчас внесем, затронут класс myPanel –класс панели окна, на которой расположено игровое поле. Начнем со свойств (переменных) класса. Нам понадобится еще один таймер, который будет изменять логику игры, а другими словами,изменять двумерный массив. Этот таймер будет вызывать метод для перемещения головы змейки. Назовем переменную esa/.xz\w.,/."KI/;0NU5GN4Y86RT7G96RFNCJ4.9R

\ATQ N5I ZGSZ\*22222222222222222222222222222222222222222222215*Update:

// Переменная для реализации логики игры

private game myGame;

// Два таймера: отрисовки и изменения логики игры

private Timer tmDraw, tmUpdate;

// Изображения, используемые в игре

private Image fon,telo,golova,ob,endg;

// Надпись для количества очков

private JLabel lb;

// Две кнопки

private JButton btn1,btn2;4

Сразу под блоком свойств(переменных) класса добавим класс для обработки событий от клавиатуры. Этот класс был использован в игре "Новогодний дождь":

// Класс для обработки событий от клавиатуры

private class myKey implements KeyListener

{

//Метод при нажатии на клавишу

public void keyPressed(KeyEvent e)

{

// Получение кода нажатой клавиши

int key = e.getKeyCode();

// Если нажатие одной из четырех стрелочек, то

// изменение направления змейки

if (key==KeyEvent.VK\_LEFT) myGame.napr = 0;

else if (key==KeyEvent.VK\_UP) myGame.napr = 1;

else if (key==KeyEvent.VK\_RIGHT) myGame.napr = 2;

else if (key==KeyEvent.VK\_DOWN) myGame.napr = 3;

}

public void keyReleased(KeyEvent e) {}

public void keyTyped(KeyEvent e) {}

}

При разработке программ невозможно запомнить названия всех классов и методов. И нет смысла ставить себе такую цель! Если программист использует разные языки программирования и много различных библиотек, то держать в голове все –это сложная задача.При регулярном написании программного кода, названия некоторых классов и методов запоминаются автоматически. Также помогает подсветка кода со списком подходящих наименований. Но переключившись на длительное время на другой язык программирования или другое направление разработки, программист забывает часть наименований. Для решения этого вопроса существует простой способ–база рецептов программирования. Допустим,программист познакомился с методикой обработки событий от клавиатуры на языке Java при разработке для настольных компьютеров.Этот фрагмент программного кода можно сохранить, а потом использовать при разработке какого-либо проекта, внося при необходимости изменения.

Описанный выше класс для обработки нажатий клавиш уже использовался при создании игры"Новогодний дождь". Нужно лишь взять его и вставить в наш программный код.

Код нажатой клавиши попадает в переменную key:

// Получение кода нажатой клавиши

int key = e.getKeyCode();

Нас интересуют клавиши http://ok-t.ru/life-prog/baza2/334145791143.files/image120.jpg ("вправо"), http://ok-t.ru/life-prog/baza2/334145791143.files/image122.jpg ("влево"), http://ok-t.ru/life-prog/baza2/334145791143.files/image124.jpg ("вверх"), http://ok-t.ru/life-prog/baza2/334145791143.files/image126.jpg ("вниз"),

Условная конструкция определяет - какое именно направление было выбрано пользователем:

// Если нажатие одной из четырех стрелочек,то

// изменение направления змейки

if (key==KeyEvent.VK\_LEFT) myGame.napr = 0;

else if (key==KeyEvent.VK\_UP) myGame.napr = 1;

else if (key==KeyEvent.VK\_RIGHT) myGame.napr = 2;

else if (key==KeyEvent.VK\_DOWN) myGame.napr = 3;

В зависимости от направления в переменную napr, которая находится в классе game,передается необходимое значение:0,1,2,3. Для удобства анализа кодов клавиш в языке Java имеется удобное перечисление KeyEvent.

после его названия через оператор "." ("точка"), можно выбрать наименование нужной клавиши. Наименования клавиш начинаются с двух букв VK.

Примеры:

VK\_LEFT–клавиша http://ok-t.ru/life-prog/baza2/334145791143.files/image122.jpg ("влево"), VK\_DOWN–клавиша http://ok-t.ru/life-prog/baza2/334145791143.files/image126.jpg ("вниз"), VK\_3–цифра 3,VK\_A–буква A, VK\_ALT–клавиша<Alt>.

Таким образом, мы связываем нажатия нужных нам клавиш со значениями переменной napr в классе game. Созданный класс–это лишь шаблон, описание. Пока мы не воспользуемся классом –его содержимое не будет работать. Поэтому в самом начале конструктора класса myPanel добавим строчки:

// Конструктор класса

public myPanel()

{

// Подключение обработчика события для клавиатуры к панели

this.addKeyListener(new myKey());

// Делаем панель в фокусе -для приема событий от клавиатуры

this.setFocusable(true);

this –это обращение к текущему объекту, к панели окна, которое будет принимать события от клавиатуры.Метод addKeyListener() добавляет к панели обработчик события клавиатуры. Внутри круглых скобок метода создается объект на основании класса, созданного на предыдущем шаге:new myKey().

// Делаем панель в фокусе - для приема событий от клавиатуры

this.setFocusable(true);

Следующая строка делает панель в фокусе. Это нужно для приема панелью событий от клавиатуры.

Внутри метода paintComponent()для второго уровня сложности ничего дорабатывать не придется. Этот метод выполняет отрисовку игрового поля. Он отрисовывает его на основании данных двумерного массива. Если значение массива, соответствующее голове змейки будет перемещаться в самом массиве,то метод paintComponent() будет выводить ее уже в новом месте на игровом поле. Мы добавили переменную класса tmUpdate –это таймер для изменения логики игры. В конструкторе класса myPanel рядом с настройкой первого таймера tmDraw (выше или ниже)добавим код для таймера tmUpdate:

//Создаем, настраиваем и запускаем таймер для изменения

//логики игры

tmUpdate = new Timer(100,new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

// Перемещаем голову змейки

myGame.peremGolova();

// Выводим информацию о количестве очков

lb.setText("Счет: "+myGame.kol);

}

});

tmUpdate.start();

Этот участок программного кода описывает настройку и запуск таймера tmUpdate. Таймер будет срабатывать через 100 миллисекунд, 10 раз в секунду. Он будет 10 раз в секунду вызывать метод peremGolova(), который изменяет положение головы змейки в массиве. Этот метод находится в классе game.

// Перемещаем голову змейки

myGame.peremGolova();

Кроме этого таймер будет постоянно обновлять счет игры:

//Выводим информацию о количестве очков

lb.setText("Счет: "+myGame.kol);

После слова "Счет: "будет выводиться текущее значение переменной kol, которая находится в классе game. При поедании объекта змейкой–значение переменной kol будет увеличиваться на 10.

Сохраним наш проект и запустим игру(см. рис. 1).

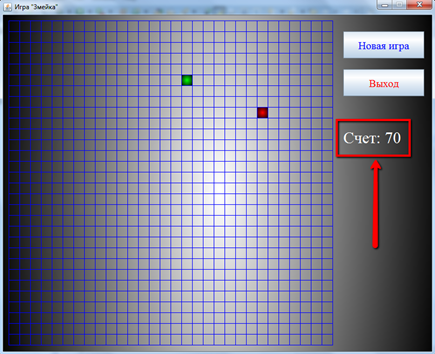


Рис. 1

Голова змейки будет перемещаться и можно изменять ее направление при помощи клавиатуры. При выходе за пределы игрового поля –голова змейки будет появляться с противоположной стороны. При поедании объекта змейкой –счет будет увеличиваться (см. рис.1).

Но есть еще одна проблема, которую предстоит решить. Если в процессе игры нажать кнопку Новая игра, то счет обнулится –и это правильно! (см. рис.2)

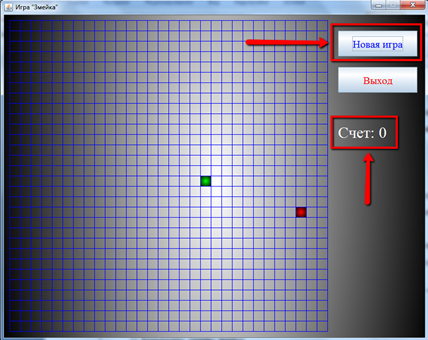


Рис. 2

Но проблема в том, что после начала новой игры пропадет возможность управлением головой змейки.Она не будет реагировать на нажатие клавиш!Это связано с тем, что при нажатии на кнопку Новая игра она получает фокус, при этом панель теряет фокус. После этого все события от клавиатуры принимаются кнопкой,но наш обработчик события подключен именно к панели и не получает информацию о событиях кнопки. Поэтому после нажатия на кнопку нужно снова передавать фокус панели.

Добавим к классу myPanel еще одну переменную класса:

// Ссылка на панель

private myPanel pan;

В эту переменную мы поместим ссылку на панель с игровым полем.Обратите внимание, что тип переменной–myPanel. В самом верху конструктора класса myPanel добавим строку:

// Конструктор класса

public myPanel()

{

// Помещаем ссылку на саму панель в переменную

pan = this;

Переменная pan будет ссылаться на панель. Теперь через эту переменную мы сможем обратиться к панели в любом месте программного кода класса myPanel. Последнее, что нужно сделать –это добавить три строки в обработчик события - нажатие на кнопку Новая игра:

btn1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

// Запуск игры

myGame.start();

// Забираем фокус у кнопки Новая игра

btn1.setFocusable(false);

// Забираем фокус у кнопки Выход

btn2.setFocusable(false);

// Отдаем фокус панели

pan.setFocusable(true);

}

});

После перезапуска игры мы забираем фокус у двух кнопок и передаем фокус панели. Если мы запустим игру снова,то увидим, что после перезапуска игры– все работает правильно! Создание игры Змейка второго уровня сложности закончено. Полный программный код файла zmeika.java второго уровня сложности 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.\*;

// Главный класс программы

public class zmeika

{

// Метод запуска приложения

public static void main(String[] args)

{

//Создание объекта окна игрового поля

myFrame okno = new myFrame();

}

}

// Класс окна игрового поля

class myFrame extends JFrame

{

// Конструктор класса

public myFrame()

{

//Создание объекта панели и подключения ее к окну

myPanel pan = new myPanel();

Container cont = getContentPane();

cont.add(pan);

//Заголовок окна

setTitle("Игра \"Змейка\"");

//Границы окна: расположение и размеры

setBounds(0, 0, 800, 650);

//Операция при закрытии окна - завершение приложения

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//Запрет изменения размеров окна

setResizable(false);

//Отображение (показ) окна

setVisible(true);

}

}

// Класс панели игрового поля

class myPanel extends JPanel

{

// Переменная для реализации логики игры

private game myGame;

// Два таймера: отрисовки и изменения логики игры

private Timer tmDraw, tmUpdate;

// Изображения, используемые в игре

private Image fon,telo,golova,ob,endg;

// Надпись для количества очков

private JLabel lb;

// Две кнопки

private JButton btn1,btn2;

// Ссылка на панель

private myPanel pan;

// Класс для обработки событий от клавиатуры

private class myKey implements KeyListener

{

//Метод при нажатии на клавишу

public void keyPressed(KeyEvent e)

{

// Получение кода нажатой клавиши

int key = e.getKeyCode();

// Если нажатие одной из четырех стрелочек, то

// изменение направления змейки

if (key==KeyEvent.VK\_LEFT) myGame.napr = 0;

else if (key==KeyEvent.VK\_UP) myGame.napr = 1;

else if (key==KeyEvent.VK\_RIGHT) myGame.napr = 2;

else if (key==KeyEvent.VK\_DOWN) myGame.napr = 3;

}

public void keyReleased(KeyEvent e) {}

public void keyTyped(KeyEvent e) {}

}

// Конструктор класса

public myPanel()

{

// Помещаем ссылку на саму панель в переменную

pan = this;

//Подключение обработчика события для клавиатуры к панели

this.addKeyListener(new myKey());

// Делаем панель в фокусе - для приема событий от клавиатуры

this.setFocusable(true);

//Попытка загрузки всех изображений для игры

try

{

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

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

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

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

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

}

catch (Exception ex) {}

//Создаем объект новой игры

myGame = new game();

myGame.start();

//Создаем, настраиваем и запускаем таймер для

//отрисовки игрового поля

tmDraw = new Timer(20,new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

// Вызываем перерисовку -paintComponent()

repaint();

}

});

tmDraw.start();

//Создаем, настраиваем и запускаем таймер

// для изменения логики игры

tmUpdate = new Timer(100,new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

// Перемещаем голову змейки

myGame.peremGolova();

// Выводим информацию о количестве очков

lb.setText("Счет: "+myGame.kol);

}

});

tmUpdate.start();

//Включаем возможность произвольного размещения

//элементов интерфейса на панели

setLayout(null);

//Создаем текстовую надпись

lb = new JLabel("Счет: 0");

lb.setForeground(Color.WHITE);

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

lb.setBounds(630, 200, 150, 50);

add(lb);

//Создаем кнопку Новая игра

btn1 = new JButton();

btn1.setText("Новая игра");

btn1.setForeground(Color.BLUE);

btn1.setFont(new Font("serif",0,20));

btn1.setBounds(630, 30, 150, 50);

btn1.addActionListener(new ActionListener() {

// Обработчик события при нажатии на кнопку Новая игра

public void actionPerformed(ActionEvent arg0) {

// Запуск игры

myGame.start();

// Забираем фокус у кнопки Новая игры

btn1.setFocusable(false);

// Забираем фокус у кнопки Выход

btn2.setFocusable(false);

// Отдаем фокус панели

pan.setFocusable(true);

}

});

add(btn1);

//Создаем кнопку Выход

btn2 = new JButton();

btn2.setText("Выход");

btn2.setForeground(Color.RED);

btn2.setFont(new Font("serif",0,20));

btn2.setBounds(630, 100, 150, 50);

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,0,0,800,650,null);

//Отрисовка игрового поля на основании массива

for (int i = 0; i < 30; i++) {

for (int j = 0; j < 30; j++) {

if (myGame.mas[i][j]!=0)

{

if (myGame.mas[i][j]==1)

{

// Выводим голову змейки в ячейку игрового поля

gr.drawImage(golova,10+j\*20, 10+i\*20,20,20,null);

}

else if (myGame.mas[i][j]==-1)

{

// Выводим объект для поедания в ячейку игрового поля gr.drawImage(ob,10+j\*20, 10+i\*20,20,20,null);

}

}

}

}

//Отрисовка сетки игрового поля из синих линий

gr.setColor(Color.BLUE);

for (int i = 0; i <= 30; i++)

{

// Рисование линий сетки

gr.drawLine(10+i\*20, 10, 10+i\*20, 610);

gr.drawLine(10, 10+i\*20, 610, 10+i\*20);

}

}

}