Министерство образования и науки

Российской Федерации

Федеральное государственное бюджетное

Образовательное учреждение

Высшего профессионального образования

«Уральский государственный горный университет»

Отчет

По дисциплине «Программирование»

Игра Грузчик.

Выполнил: Путилов. А.А

Группа: ИНФ-4з

Проверил: Волкова Е. А

Екатеринбург - 2016

Оглавление

[Введение 3](#_Toc469347218)

[Конструктор игры 4](#_Toc469347219)

[Класс EditorFrame 4](#_Toc469347220)

[Класс EditorBoard 4](#_Toc469347221)

[Игра 9](#_Toc469347222)

[Класс GameFrame 9](#_Toc469347223)

[Класс Mario 9](#_Toc469347224)

[Класс Caisse 10](#_Toc469347225)

[Класс Objectif 11](#_Toc469347226)

[Класс Mur 11](#_Toc469347227)

[Класс GameBoard 12](#_Toc469347228)

[Меню запуска игры 17](#_Toc469347229)

[Класс Main 17](#_Toc469347230)

[Класс MenuFrame 17](#_Toc469347231)

[Заключение 19](#_Toc469347232)

[Список литературы 20](#_Toc469347233)

# Введение

Все мы когда то были детьми, наверняка самые светлые воспоминания и впечатления связаны у нас с играми.

Игра теснейшим образом связана с развитием личности

Игра - это жизненно важный и необходимый элемент в развитии как индивидуума, так и общества в целом.

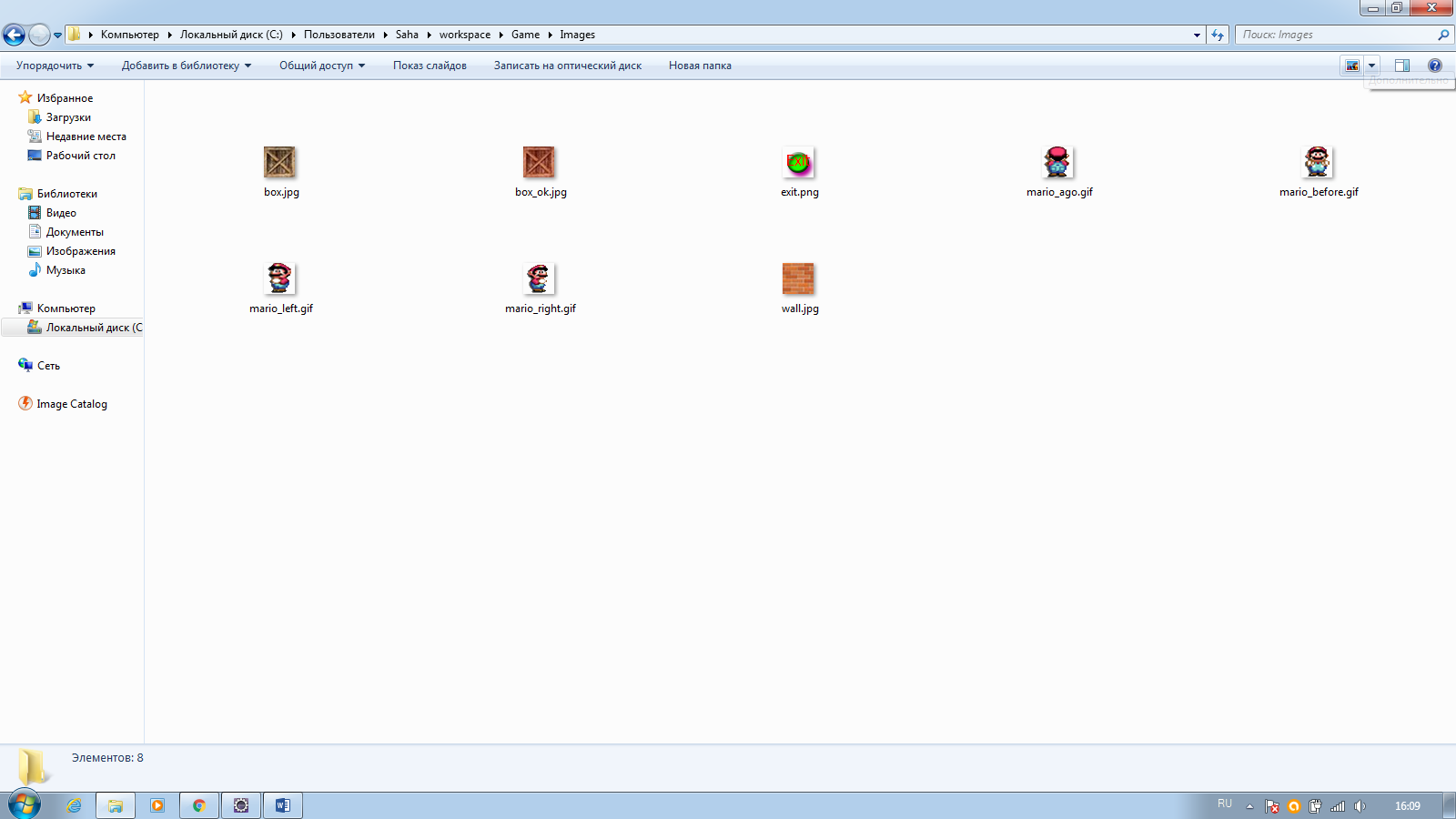
В данной работе была реализована игра «Sokoban». Sokoban (Soko-Ban, яп. сокобан -- «кладовщик») - логическая игра-головоломка, в которой игрок передвигает ящики по лабиринту, показанному в виде плана, с целью поставить все ящики на заданные конечные позиции. Только один ящик может быть передвинут за раз, причём герой игры -- «кладовщик» -- может только толкать ящики, но не тянуть их. Поскольку игру достаточно сложно воссоздать физически, обычно она реализуется в виде компьютерной игры.

Игра Sokoban была создана в 1980 году Hiroyuki Imabayashi, и издана в 1982 году компанией Thinking Rabbit (Япония). Кроме того, компания выпустила три сиквела: Boxxle, Sokoban Perfect и Sokoban Revenge.

Игра была реализована для множества компьютерных платформ, включая практически все домашние и персональные компьютеры. Также существуют версии игры для карманных компьютеров, игровых приставок и мобильных телефонов.

# Конструктор игры

Создаем новый проект назовем его Game. Далее в появившуюся папку добавляем заготовки для игры (изображения).



## Класс EditorFrame

В этом классе будет реализован GUI интерфейс для редактора карт.

**public** **class** EditorFrame **extends** JFrame {

**public** EditorFrame(){

**this**.setTitle("Редактор карт");

**this**.setSize(765, 780);

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

**this**.add(**new** EditorBoard(**this**));

**this**.setVisible(**true**);

**this**.setLocationRelativeTo(**null**);

}

}

## Класс EditorBoard

Это основной класс конструктора в котором описываем логику построения карт их редактирования и сохранения.

**public** **class** EditorBoard **extends** JPanel **implements** MouseListener, MouseMotionListener,MouseWheelListener, KeyListener{

String Editeur[][] = **new** String[24][24];

String ImageSelect[] = {"WALL","BOX","MARIO","EXIT"};

String ImageCourante = "WALL";

**int** Mx, My;

**int** indexInc = 0;

Image Mario;

Image Mur;

Image Caisse;

Image Objectif;

FileWriter fw;

FileReader fr;

Frame Eframe;

**public** EditorBoard(Frame ef){

ImageIcon iMario = **new** ImageIcon("Images/mario\_before.gif");

Mario = iMario.getImage();

ImageIcon iMur = **new** ImageIcon("Images/wall.jpg");

Mur = iMur.getImage();

ImageIcon iCaisse = **new** ImageIcon("Images/box.jpg");

Caisse = iCaisse.getImage();

ImageIcon iObjectif = **new** ImageIcon("Images/exit.png");

Objectif = iObjectif.getImage();

Eframe = ef;

setFocusable(**true**);

addMouseListener(**this**);

addMouseMotionListener(**this**);

addMouseWheelListener(**this**);

addKeyListener(**this**);

}

Метод перерисовки построение карты

**public** **void** paint (Graphics g){

**super**.paint(g);

Graphics2D g2d = (Graphics2D) g;

**for** (**int** i= 0; i <= 23; i++){

**for** (**int** j = 0; j <= 23; j++){

**if** (Editeur[i][j]== "WALL"){

g2d.drawImage(Mur, i \* 34, j \* 34, **null**);

}

**if** (Editeur[i][j]== "MARIO"){

g2d.drawImage(Mario, i \* 34, j \* 34, **null**);

}

**if** (Editeur[i][j]== "BOX"){

g2d.drawImage(Caisse, i \* 34, j \* 34, **null**);

}

**if** (Editeur[i][j]== "EXIT"){

g2d.drawImage(Objectif, i \* 34, j \* 34, **null**);

}

}

}

**if** (ImageCourante == "WALL"){

g2d.drawImage(Mur, Mx, My, **null**);

}

**else** **if** (ImageCourante == "BOX"){

g2d.drawImage(Caisse, Mx, My, **null**);

}

**else** **if** (ImageCourante == "MARIO"){

g2d.drawImage(Mario, Mx, My, **null**);

}

**else** **if** (ImageCourante == "EXIT"){

g2d.drawImage(Objectif, Mx, My, **null**);

}

}

@Override

**public** **void** keyPressed(KeyEvent arg0) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** keyReleased(KeyEvent arg0) {

**char** key =arg0.getKeyChar();

Сохранение созданных уровней

**if** (key =='s'){

**try**{

fw = **new** FileWriter (JOptionPane.*showInputDialog*(**null**, "Введите путь резервного копирования:", "Редактор карт", JOptionPane.***QUESTION\_MESSAGE***));

**for** (**int** i = 0; i < 24; i++){

**for** (**int** j = 0; j < 24; j++){

**if** (Editeur[j][i] == "WALL"){

fw.write("0");

}

**else** **if** (Editeur[j][i] == "MARIO"){

fw.write("1");

}

**else** **if** (Editeur[j][i] == "BOX"){

fw.write("2");

}

**else** **if** (Editeur[j][i] == "EXIT"){

fw.write("3");

}

**else** **if** (Editeur[j][i] == **null**){

fw.write(" ");

}

}

fw.write("\r\n");

}

fw.close();

}

**catch** (Exception ex){}

}

Добавляет редактор уровне созданных

**else** **if** (key == 'l'){

**try**{

fr = **new** FileReader (JOptionPane.*showInputDialog*(**null**, "Введите путь резервного копирования:", "Редактор карт", JOptionPane.***QUESTION\_MESSAGE***));

**int** i = 0;

**int** x = 0, y = 0;

**while** ((i = fr.read()) != -1){

**char** strImg = (**char**) i;

**if** (strImg == '0'){

Editeur[x][y] = "WALL";

}

**else** **if** (strImg =='1'){

Editeur[x][y] = "MARIO";

}

**else** **if** (strImg =='2'){

Editeur[x][y] = "BOX";

}

**else** **if** (strImg =='3'){

Editeur[x][y] = "EXIT";

}

**else** **if** (strImg ==' '){

Editeur[x][y] = **null**;

}

**else** **if** (strImg =='\r' || strImg =='\n'){

x--;

}

**if** (x == 23){

y++;

x = 0;

}

**else**{

x++;

}

}

}

**catch** (Exception ex){}

}

**else** **if** (key == KeyEvent.***VK\_ESCAPE***){

MenuFrame frm = **new** MenuFrame();

Eframe.dispose();

}

}

@Override

**public** **void** keyTyped(KeyEvent arg0) {

// **TODO** Auto-generated method stub

}

Задает функцию для колеса мыши смена иконок

**public** **void** mouseWheelMoved(MouseWheelEvent arg0) {

**int** rot = arg0.getWheelRotation();

**if** (rot < 0){

**if** (indexInc > 0){

indexInc--;

}

}

**else** **if** (rot > 0){

**if** (indexInc < 3){

indexInc++;

}

}

ImageCourante = ImageSelect[indexInc];

repaint();

}

@Override

**public** **void** mouseDragged(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

**public** **void** mouseMoved(MouseEvent arg0) {

Mx = arg0.getX() - 17;

My = arg0.getY() - 17;

repaint();

}

@Override

**public** **void** mouseClicked(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseEntered(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseExited(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mousePressed(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

Отслеживает нажатие мыши

**public** **void** mouseReleased(MouseEvent arg0) {

**int** x = arg0.getX() / 34;

**int** y = arg0.getY() / 34;

**if** (arg0.getButton()== MouseEvent.***BUTTON1***){

Editeur[x][y] = ImageCourante;

}

**else** **if** (arg0.getButton() == MouseEvent.***BUTTON3***){

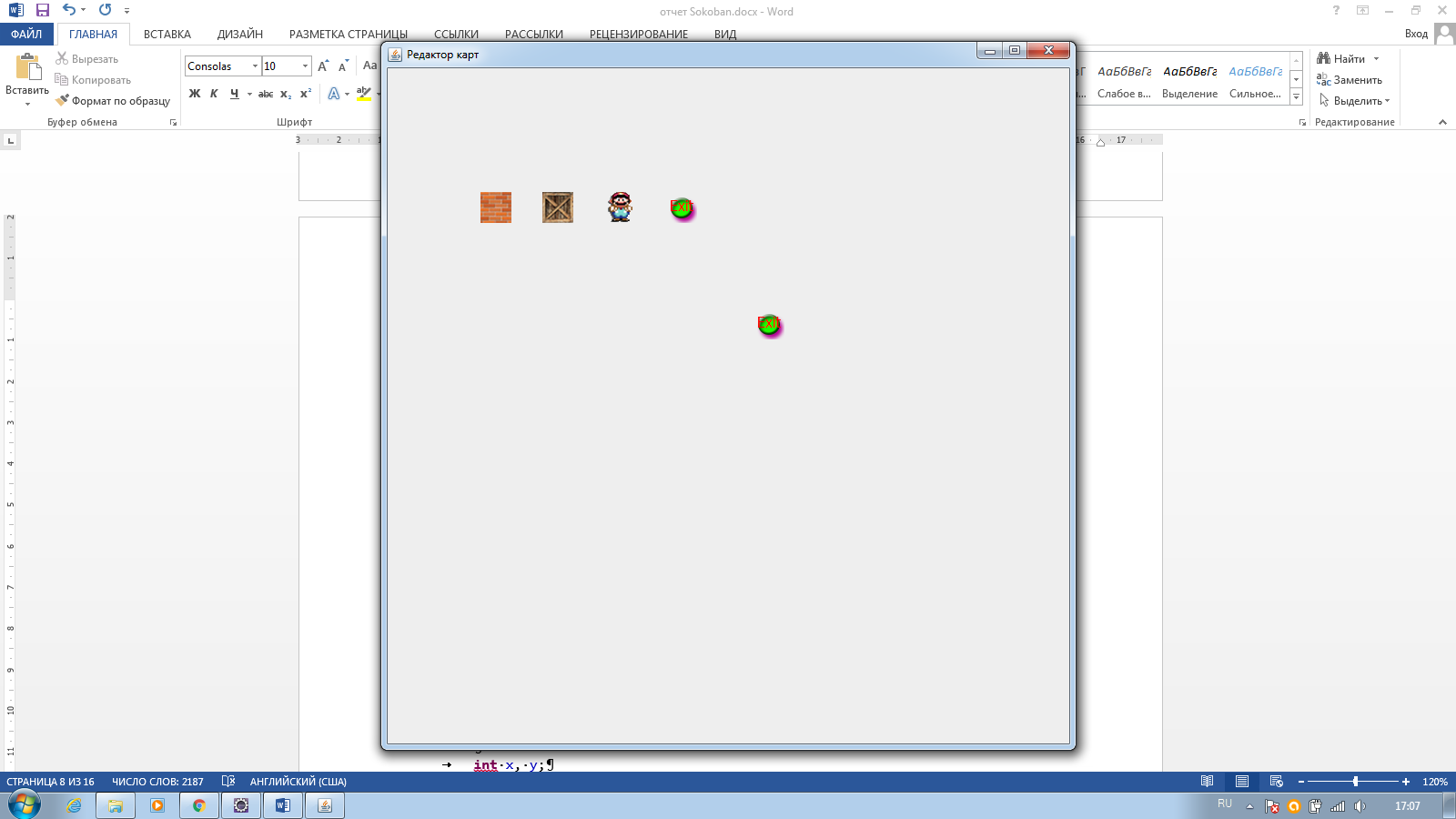
Editeur[x][y] = **null**;

}

}

}

Результат работы конструктора:



# Игра

## Класс GameFrame

Описываем GUI интерфейс игры

**public** **class** GameFrame **extends** JFrame{

**public** GameFrame(){

**this**.setTitle("Игра");

**this**.setSize(765, 780);

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

**this**.add(**new** GameBoard(**this**));

**this**.setVisible(**true**);

**this**.setResizable(**false**);

**this**.setLocationRelativeTo(**null**);

}

}

## Класс Mario

Классы Mario, Caisse, Objectif, Mur добавляют переменные и ссылки на картинки

**public** **class** Mario {

**int** x, y;

String MarioDir = "BAS";

Image Mario;

ImageIcon iMarioBas = **new** ImageIcon("Images/mario\_before.gif");

ImageIcon iMarioDroite = **new** ImageIcon("Images/mario\_right.gif");

ImageIcon iMarioGauche = **new** ImageIcon("Images/mario\_left.gif");

ImageIcon iMarioHaut = **new** ImageIcon("Images/mario\_ago.gif");

**public** Mario(**int** Startx, **int** Starty){

x = Startx;

y = Starty;

}

**public** Rectangle getBounds(){

Rectangle Box = **new** Rectangle (x, y, 34, 34);

**return** Box;

}

**public** **int** getX(){

**return** x;

}

**public** **int** getY(){

**return** y;

}

**public** String getDir(){

**return** MarioDir;

}

**public** **void** setX(**int** newX){

**this**.x = newX;

}

**public** **void** setY(**int** newY){

**this**.y = newY;

}

**public** **void** setDir(String newDir){

**this**.MarioDir = newDir;

}

**public** Image getImage(){

**if** (MarioDir == "BAS"){

Mario = iMarioBas.getImage();

}

**else** **if** (MarioDir == "DROITE"){

Mario = iMarioDroite.getImage();

}

**else** **if** (MarioDir == "GAUCHE"){

Mario = iMarioGauche.getImage();

}

**else** **if** (MarioDir == "HAUT"){

Mario = iMarioHaut.getImage();

}

**return** Mario;

}

**public** **void** Move(){

**if** (MarioDir == "BAS"){

**this**.y += 34;

}

**else** **if** (MarioDir == "DROITE"){

**this**.x += 34;

}

**else** **if** (MarioDir == "GAUCHE"){

**this**.x -= 34;

}

**else** **if** (MarioDir == "HAUT"){

**this**.y -= 34;

}

}

}

## Класс Caisse

**public** **class** Caisse {

**int** x,y;

String Etat = "NORMALE";

Image Caisse;

ImageIcon iCaisseNormale = **new** ImageIcon("Images/box.jpg");

ImageIcon iCaisseObjectif = **new** ImageIcon("Images/box\_ok.jpg");

**boolean** caissejuste = **false**;

**public** Caisse(**int** Startx, **int** Starty){

x = Startx;

y = Starty;

}

**public** Rectangle getBounds(){

Rectangle Box = **new** Rectangle (x, y, 34, 34);

**return** Box;

}

**public** **int** getX(){

**return** x;

}

**public** **int** getY(){

**return** y;

}

**public** **void** setX(**int** newX){

**this**.x = newX;

}

**public** **void** setY(**int** newY){

**this**.y = newY;

}

**public** **boolean** getJuste(){

**return** caissejuste;

}

**public** **void** setJuste(**boolean** newJuste){

**this**.caissejuste = newJuste;

}

**public** **void** setEtat(String newEtat){

**this**.Etat = newEtat;

}

**public** Image getImage(){

**if** (Etat == "NORMALE"){

Caisse = iCaisseNormale.getImage();

}

**else** **if**(Etat == "OBJECTIF"){

Caisse = iCaisseObjectif.getImage();

}

**return** Caisse;

}

## Класс Objectif

**public** **class** Objectif {

**int** x, y;

Image Objectif;

**boolean** caisseDessus = **false**;

**public** Objectif(**int** Startx, **int** Starty){

x = Startx;

y = Starty;

ImageIcon iObjectif = **new** ImageIcon("Images/exit.png");

Objectif = iObjectif.getImage();

}

**public** Rectangle getBounds(){

Rectangle Box = **new** Rectangle(x, y, 34, 34);

**return** Box;

}

**public** **boolean** getDessus(){

**return** caisseDessus;

}

**public** **void** setDessus(**boolean** newDessus){

**this**.caisseDessus = newDessus;

}

**public** **int** getX(){

**return** x;

}

**public** **int** getY(){

**return** y;

}

**public** Image getImage(){

**return** Objectif;

}

}

## Класс Mur

**public** **class** Mur {

**int** x,y;

Image Mur;

**public** Mur(**int** Startx, **int** Starty){

x = Startx;

y = Starty;

ImageIcon iMur = **new** ImageIcon ("Images/wall.jpg");

Mur = iMur.getImage();

}

**public** Rectangle getBounds(){

Rectangle Box = **new** Rectangle (x, y, 34, 34);

**return** Box;

}

**public** **int** getX(){

**return** x;

}

**public** **int** getY(){

**return** y;

}

**public** Image getImage(){

**return** Mur;

}

}

## Класс GameBoard

Основной класс игры. Запускает карту из папки Maps. Добавляет счетчик уровней. Перерисовывает карту и создает логику игры.

**public** **class** GameBoard **extends** JPanel **implements** KeyListener{

String Game[][] = **new** String [24][24];

**int** Level = 1;

**private** **static** ArrayList<Mur> *Murs*;

**private** **static** ArrayList<Objectif> *Objectifs*;

**private** **static** ArrayList<Caisse> *Caisses*;

Mur mur;

Mario mario;

Caisse caisse;

Objectif objectif;

Font levelFont = **new** Font("SansSerif", Font.***BOLD***, 20);

FileReader fr;

**public** GameBoard(Frame gf){

ChargerLevel();

setFocusable(**true**);

addKeyListener(**this**);

}

Построение карты по сохраненному файлу

**public** **void** ChargerLevel(){

**try**{

fr = **new** FileReader("Maps/level" + Level + ".level");

**int** x = 0, y = 0, i= 0;

*Murs* = **new** ArrayList<Mur>();

*Caisses* = **new** ArrayList<Caisse>();

*Objectifs* = **new** ArrayList<Objectif>();

**while** ((i = fr.read()) != -1){

**char** strImg = (**char**) i;

**if** (strImg == '0'){

Game[x][y] = "MUR";

mur = **new** Mur(x \* 34, y \* 34);

*Murs*.add(mur);

}

**else** **if** (strImg =='1'){

Game[x][y] = "MARIO";

mario = **new** Mario(x \* 34, y \* 34);

}

**else** **if** (strImg =='2'){

Game[x][y] = "CAISSE";

caisse = **new** Caisse(x \* 34, y \* 34);

*Caisses*.add(caisse);

}

**else** **if** (strImg =='3'){

Game[x][y] = "OBJECTIOF";

objectif = **new** Objectif(x \* 34, y \* 34);

*Objectifs*.add(objectif);

}

**else** **if** (strImg ==' '){

Game[x][y] = **null**;

}

**else** **if** (strImg =='\r'|| strImg == '\n'){

x--;

}

**if** (x == 23){

y++;

x = 0;

}

**else**{

x++;

}

}

}

**catch** (Exception ex){}

repaint();

}

Метод перерисовки

**public** **void** paint (Graphics g){

**super**.paint(g);

Graphics2D g2d = (Graphics2D) g;

**for** ( **int** i = 0; i < *Murs*.size(); i++){

mur = (Mur) *Murs*.get(i);

g2d.drawImage(mur.getImage(), mur.getX(), mur.getY(), **null**);

}

**for** ( **int** i = 0; i < *Objectifs*.size(); i++){

objectif = (Objectif) *Objectifs*.get(i);

g2d.drawImage(objectif.getImage(), objectif.getX(), objectif.getY(), **null**);

}

**for** ( **int** i = 0; i < *Caisses*.size(); i++){

caisse = (Caisse) *Caisses*.get(i);

g2d.drawImage(caisse.getImage(), caisse.getX(), caisse.getY(), **null**);

}

**try**{

g2d.drawImage(mario.getImage(), mario.getX(), mario.getY(), **null**);

}

**catch**(Exception ex){}

g.setColor(Color.***BLACK***);

g.setFont(levelFont);

g.drawString("LEVEL : " + Level, 10, 25);

}

@Override

**public** **void** keyPressed(KeyEvent arg0) {

// **TODO** Auto-generated method stub

}

Управление персонажем

**public** **void** keyReleased(KeyEvent arg0) {

**int** Touche = arg0.getKeyCode();

**if** (Touche == KeyEvent.***VK\_DOWN***){

mario.setDir("BAS");

mario.Move();

CheckCollision();

}

**else** **if** (Touche == KeyEvent.***VK\_UP***){

mario.setDir("HAUT");

mario.Move();

CheckCollision();

}

**else** **if** (Touche == KeyEvent.***VK\_RIGHT***){

mario.setDir("DROITE");

mario.Move();

CheckCollision();

}

**else** **if** (Touche == KeyEvent.***VK\_LEFT***){

mario.setDir("GAUCHE");

mario.Move();

CheckCollision();

}

**else** **if** (Touche == KeyEvent.***VK\_R***){

ChargerLevel();

}

repaint();

VerifierLevelFini();

}

@Override

**public** **void** keyTyped(KeyEvent arg0) {

// **TODO** Auto-generated method stub

}

Код проверяет все ли ящики стоят на месте если да то переход на следующий уровень

**public** **void** VerifierLevelFini(){

**int** nbCaisseJuste = 0;

**for** (**int** i = 0; i < *Caisses*.size(); i++){

caisse = (Caisse) *Caisses*.get(i);

**if** (caisse.Etat == "OBJECTIF"){

nbCaisseJuste++;

}

**if** (nbCaisseJuste == *Caisses*.size()){

Level++;

ChargerLevel();

}

}

}

Проверяет взаимодействие персонажа с обьектами

**public** **void** CheckCollision() {

Rectangle marioRec;

marioRec = mario.getBounds();

**for** (**int** i = 0; i < *Murs*.size(); i++){

mur = (Mur) *Murs*.get(i);

Rectangle murRec = mur.getBounds();

**if** (marioRec.intersects(murRec)){

**if** (mario.getDir() == "BAS"){

mario.setY(mario.getY() - 34);

}

**else** **if**(mario.getDir() == "HAUT"){

mario.setY(mario.getY() + 34);

}

**else** **if**(mario.getDir() == "GAUCHE"){

mario.setX(mario.getX() + 34);

}

**else** **if**(mario.getDir() == "DROITE"){

mario.setX(mario.getX() - 34);

}

}

}

**for** (**int** i = 0; i < *Caisses*.size(); i++){

caisse = (Caisse) *Caisses*.get(i);

Rectangle caisseRec = caisse.getBounds();

**if** (marioRec.intersects(caisseRec)){

**if** (mario.getDir() == "BAS"){

**if** (Game[caisseRec.x / 34][(caisseRec.y + 34) / 34] !="MUR"){

caisse.setY(caisse.getY() + 34);

}

**else** **if**(Game[caisseRec.x / 34][(caisseRec.y + 34) / 34] == "MUR"){

mario.setY(mario.getY() - 34);

}

}

**else** **if**(mario.getDir() == "HAUT"){

**if**(Game[caisseRec.x / 34][(caisseRec.y - 34) / 34] != "MUR"){

caisse.setY(caisse.getY() - 34);

}

**else** **if** (Game[caisseRec.x / 34][(caisseRec.y - 34) / 34] == "MUR")

{

mario.setY(mario.getY() + 34);

}

}

**else** **if**(mario.getDir() == "GAUCHE"){

**if**(Game[(caisseRec.x - 34) / 34][caisseRec.y / 34] != "MUR"){

caisse.setX(caisse.getX() - 34);

}

**else** **if** (Game[(caisseRec.x - 34) / 34][caisseRec.y / 34] == "MUR")

{

mario.setX(mario.getX() + 34);

}

}

**else** **if**(mario.getDir() == "DROITE"){

**if**(Game[(caisseRec.x + 34) / 34][caisseRec.y / 34] != "MUR"){

caisse.setX(caisse.getX() + 34);

}

**else** **if**(Game[(caisseRec.x + 34) / 34][caisseRec.y / 34] == "MUR")

{

mario.setX(mario.getX() - 34);

}

}

}

}

**for** (**int** i = 0; i < *Objectifs*.size(); i++){

objectif = (Objectif) *Objectifs*.get(i);

Rectangle objectifRec = objectif.getBounds();

**for** (**int** j = 0; j < *Caisses*.size(); j++){

caisse = (Caisse) *Caisses*.get(i);

Rectangle caisseRec = caisse.getBounds();

**if**(caisseRec.intersects(objectifRec) && !caisse.getJuste() && !objectif.getDessus()){

caisse.setEtat("OBJECTIF");

caisse.setJuste(**true**);

objectif.setDessus(**true**);

}

**else** **if** (!caisseRec.intersects(objectifRec) && caisse.getJuste() && objectif.getDessus()){

caisse.setEtat("NORMALE");

caisse.setJuste(**false**);

objectif.setDessus(**false**);

}

}

}

}

}

# Меню запуска игры

## Класс Main

Класс запуска программы. Строка MenuFrame frm = new MenuFrame(); запускает игру с меню. Строка //marioSokobanEditor.EditorFrame frm = new marioSokobanEditor.EditorFrame(); запускает конструктор. Строка //marioSokobanGame.GameFrame frm2 = new marioSokobanGame.GameFrame(); запускает игру.

**public** **class** Main {

**public** **static** **void** main(String[] args) {

//MenuFrame frm = new MenuFrame();

//marioSokobanEditor.EditorFrame frm = new marioSokobanEditor.EditorFrame();

//marioSokobanGame.GameFrame frm2 = new marioSokobanGame.GameFrame();

}

}

## Класс MenuFrame

Класс создает окно в котором размещаются две кнопки и текст Так же в класс добавляется слушатель и обработчик событий.

**public** **class** MenuFrame **extends** JFrame **implements** ActionListener {

JButton cmdGame = **new** JButton(" Играть ");

JButton cmdEditor = **new** JButton(" Конструктор ");

JPanel pan = **new** JPanel();

JLabel p1 = **new** JLabel ("s-сохранить; "

+ "r- начать сначала");

**public** MenuFrame(){

**this**.setTitle("Меню");

**this**.setSize(765, 780);

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

**this**.setResizable(**false**);

**this**.setLocationRelativeTo(**null**);

pan.add(cmdEditor);

pan.add(cmdGame);

pan.add(p1);

p1.setHorizontalAlignment(JLabel.***CENTER***);

cmdGame.addActionListener(**this**);

cmdEditor.addActionListener(**this**);

**this**.setContentPane(pan);

**this**.setVisible(**true**);

}

Запускает фреймы при нажатии на кнопки

**public** **void** actionPerformed(ActionEvent arg0) {

**if** (arg0.getSource() == cmdGame){

GameFrame frm = **new** GameFrame();

dispose();

}

**else** **if** (arg0.getSource() == cmdEditor){

EditorFrame frm = **new** EditorFrame();

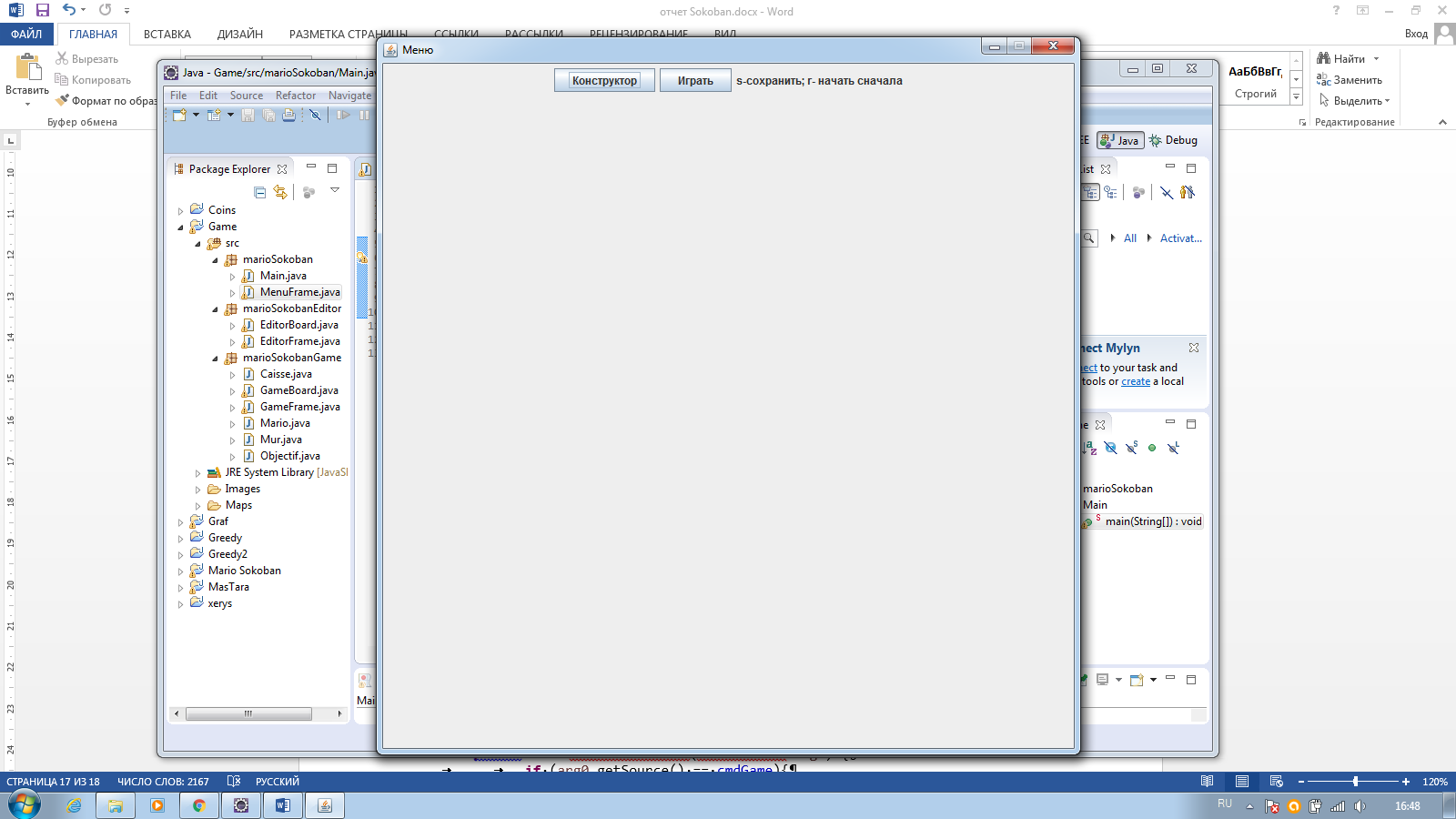
dispose();

}

}

}

Результат работы класса



# Заключение

В ходе данной задачи было создано программное приложение «Игра Sokoban», основной задачей которого является развлечение.

# Список литературы

1. Чен М.С. и др. Программирование на JAVA:1001 совет: Наиболее полное руководство по Java и Visual J++:Пер.с англ./Чен М.С.,Грифис С.В.,Изи Э.Ф..-Минск:Попурри,1997.-640с.ил.+ Прил.(1диск.)
2. Майкл Эферган Java: справочник.- QUE Corporation, 1997, Издательство "Питер Ком", 1998
3. Джо Вебер Технология Java в подлиннике.- QUE Corporation, 1996, "BHV-Санкт-Петербург",1997
4. Джейсон Мейнджер Java: Основы программирования .- McGraw-Hill,Inc.,1996, Издательская группа BHV, Киев,1997
5. И.Ю.Баженова Язык программирования Java.- АО "Диалог-МИФИ", 1997
6. Джон Родли Создание Java-апплетов.- The Coriolis Group,Inc.,1996, Издательство НИПФ "ДиаСофт Лтд.",1996
7. Нотон П. JAVA:Справ.руководство:Пер.с англ./Под ред.А.Тихонова.-М.:БИНОМ:Восточ.Кн.Компания,1996:Восточ.Кн.Компания.-447с..-(Club Computer)
8. Патрик Нотон, Герберт Шилдт Полный справочник по Java .- McGraw-Hill,1997, Издательство "Диалектика",1997
9. Дэвид Флэнэген Java in a Nutshell .- O'Reilly & Associates, Inc., 1997, Издательская группа BHV, Киев, 1998
10. Ренеган Э.Дж.(мл.) 1001 адрес WEB для программистов:Новейший путеводитель программиста по ресурсам World Wide Web:Пер.с англ..-Минск:Попурри,1997.-512с.ил