Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Глазовский государственный педагогический институт имени В.Г. Короленко»

**ОТЧЕТ**

**по учебной практике**

**ПМ 03Участие в интеграции программных модулей**

**Специальность 09.02.03 Программирование в компьютерных системах**

|  |
| --- |
| Выполнил(а): |
| студент(ка) ОП СПО группа 43 |
| Шабалин И.А. |
|  |
| Руководитель практики: ст. преподаватель Касаткин К.А. |

г. Глазов 2019

Оглавление

[Введение 2](#_Toc4074454)

[Цель разработки 3](#_Toc4074455)

[Обзор аналогов 3](#_Toc4074456)

[Постановка задачи 3](#_Toc4074457)

[Описание алгоритма работы программы 4](#_Toc4074458)

[Исходный код программы 5](#_Toc4074459)

[Тестирование программы 11](#_Toc4074460)

[Руководство по установке 11](#_Toc4074461)

# Введение

На данный момент существует великое множество компьютерных игр, так же разнообразны подходы в их создании.

Например, популярные в последнее время небольшие игры (чаще всего многопользовательские, через сеть Интернет) создаются по технологии Flash с использованием языка ActionScript. Такие игры крайне популярны на разнообразных развлекательных интернет-ресурсах и а социальных сетях.

Большие, сложные игры с реалистичной 3D графикой пишутся на C++, отдельные модули к ним могут быть написаны на других языках (например AI - «искусственный интеллект» - на Lisp). Вообще, язык для создания игры выбирается как компромисс между требованиями к игре, к компьютерному железу и к стоимости разработки.

На подавляющем большинстве мобильных телефонов установлены ARM процессоры. Поэтому игры для мобильных телефонов на платформе Java пишутся на специальном языке для мобильных телефонов MobileBasic, а также на диалекте языка Pascal - MIDletPascal.

В рамках учебной практики необходимо разработать небольшую программу игры в «Сапёра» поле размера 9х9 клеток с применением графики на языке высокого уровня с использованием объектно-ориентированного программирования.

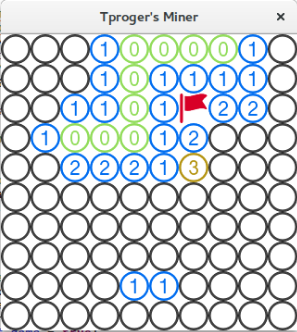
«Сапёр» - популярная игра людей всех возрастов. Она отличается простотой, способствует развитию внимания и элементарной логики. В данной работе эта игра реализована на языке Java.

# Цель разработки

Создать интересную и логическую игру, для детей и взрослых, игра будет в открытом доступе и все желающие могут воспользоваться.

# Обзор аналогов

1. Сапёр на Java от пользователя DoKel  
   Управление осуществляется мышкой  
   Правой и левой кнопкой мыши.

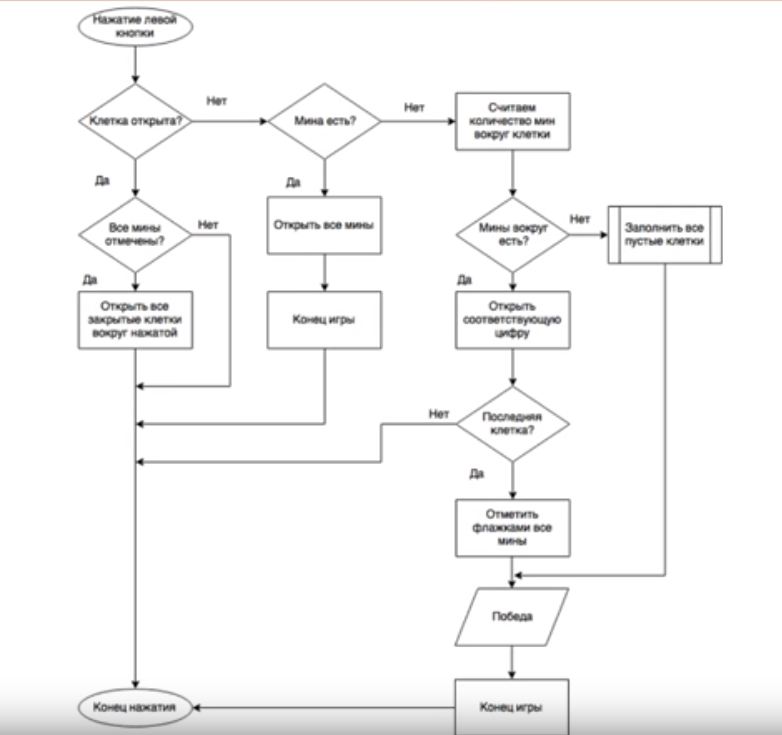
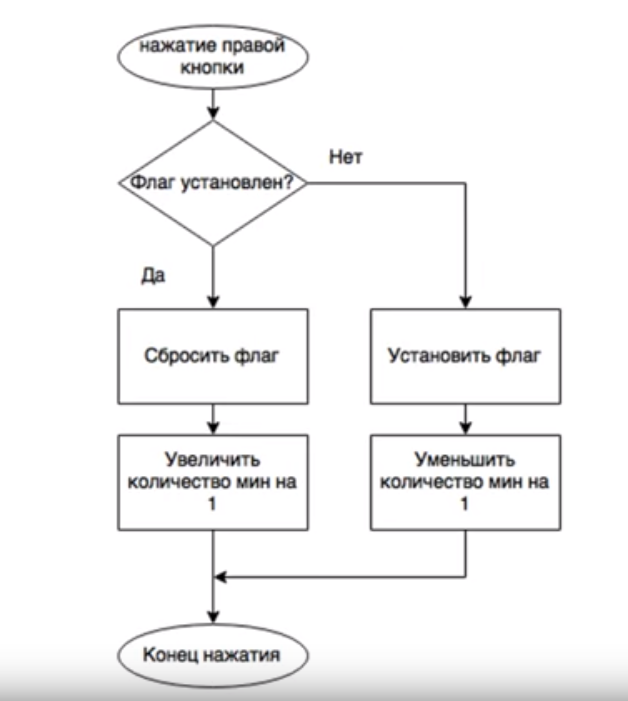


# Постановка задачи

1. Написать программу, реализующую идею игры. Использовать классы и функции. Использовать графический режим.

# Описание алгоритма работы программы

1. Отрисовка игрового поля.
2. Прием команды с мышки пользователя.
3. Соответствующее действие на принятую команду.
4. Сообщение о проигрыше или выйгрыше.



# Исходный код программы

Для написания игры, я взял язык программирования Java. Так как этот язык очень удобный и я хотел работать с ним.

# Bomb

package sweeper;  
  
public class Bomb  
{  
 private Matrix bombMap;  
 private int totalBombs;  
  
 Bomb (int totalBombs)  
 {  
 this.totalBombs = totalBombs;  
 fixBombsCount();  
 }  
  
 void start()  
 {  
 bombMap = new Matrix (Box.*ZERO*);  
 for (int j = 0; j < totalBombs; j++)  
 placeBomb ();  
 }  
  
 Box get (Coord coord)  
 {  
 return bombMap.get(coord);  
 }  
  
 private void fixBombsCount ()  
 {  
 int maxBombs = Ranges.*getSize*().x \* Ranges.*getSize*().y / 2;  
 if (totalBombs > maxBombs)  
 totalBombs = maxBombs;  
 }  
  
 private void placeBomb ()  
 {  
 while (true)  
 {  
 Coord coord = Ranges.*getRandomCoord*();  
 if (Box.*BOMB* == bombMap.get(coord))  
 continue;  
 bombMap.set (coord, Box.*BOMB*);  
 incNumbersAroundBomb(coord);  
 break;  
 }  
 }  
  
 private void incNumbersAroundBomb (Coord coord)  
 {  
 for (Coord around : Ranges.*getCoordsAround*(coord))  
 if (Box.*BOMB* != bombMap.get (around))  
 bombMap.set (around, bombMap.get(around).getNextNumberBox());  
 }  
  
 int getTotalBombs()  
 {  
 return totalBombs;  
 }  
}

**Box**

package sweeper;  
  
public enum Box  
{  
 *ZERO*,  
 *NUM1*,  
 *NUM2*,  
 *NUM3*,  
 *NUM4*,  
 *NUM5*,  
 *NUM6*,  
 *NUM7*,  
 *NUM8*,  
 *BOMB*,  
 *OPENED*,  
 *CLOSED*,  
 *FLAGED*,  
 *BOMBED*,  
 *NOBOMB*;  
  
 public Object image;  
  
 Box getNextNumberBox()  
 {  
 return Box.*values*() [this.ordinal() + 1];  
 }  
  
 int getNumber ()  
 {  
 return this.ordinal();  
 }  
}

**Coord**

package sweeper;  
  
public class Coord  
{  
 public int x;  
 public int y;  
  
 public Coord (int x, int y)  
 {  
 this.x = x;  
 this.y = y;  
 }  
  
 @Override  
 public boolean equals(Object o)  
 {  
 if (o instanceof Coord)  
 {  
 Coord to = (Coord)o;  
 return to.x == x && to.y == y;  
 }  
 return super.equals(o);  
 }  
}

**Flag**

package sweeper;  
  
class Flag  
{  
 private Matrix flagMap;  
 private int countOfClosedBoxes;  
  
 void start ()  
 {  
 flagMap = new Matrix(Box.*CLOSED*);  
 countOfClosedBoxes = Ranges.*getSize*().x \* Ranges.*getSize*().y;  
 }  
  
 Box get (Coord coord)  
 {  
 return flagMap.get (coord);  
 }  
  
 void setOpenedToBox(Coord coord)  
 {  
 flagMap.set (coord, Box.*OPENED*);  
 countOfClosedBoxes --;  
 }  
  
 void toggleFlagedToBox (Coord coord)  
 {  
 switch (flagMap.get(coord))  
 {  
 case *FLAGED* : setClosedToBox (coord); break;  
 case *CLOSED* : setFlagedToBox (coord); break;  
 }  
 }  
  
 private void setClosedToBox(Coord coord)  
 {  
 flagMap.set (coord, Box.*CLOSED*);  
 }  
  
 private void setFlagedToBox(Coord coord)  
 {  
 flagMap.set (coord, Box.*FLAGED*);  
 }  
  
 int getCountOfClosedBoxes()  
 {  
 return countOfClosedBoxes;  
 }  
  
 void setBombedToBox(Coord coord)  
 {  
 flagMap.set (coord, Box.*BOMBED*);  
 }  
  
 public void setOpenedToClosedBombBox(Coord coord)  
 {  
 if (flagMap.get (coord) == Box.*CLOSED*)  
 flagMap.set (coord, Box.*OPENED*);  
 }  
  
 public void setNobombToFlagedSafeBox(Coord coord)  
 {  
 if (flagMap.get (coord) == Box.*FLAGED*)  
 flagMap.set (coord, Box.*NOBOMB*);  
 }  
  
  
  
 int getCountOfFlagedBoxesAround(Coord coord)  
 {  
 int count = 0;  
 for (Coord around : Ranges.*getCoordsAround*(coord))  
 if (flagMap.get(around) == Box.*FLAGED*)  
 count ++;  
 return count;  
 }  
}

**Game**

package sweeper;  
  
public class Game  
{  
 private Bomb bomb;  
 private Flag flag;  
  
 private GameState state;  
 public GameState getState()  
 {  
 return state;  
 }  
  
  
 public Game (int cols, int rows, int bombs)  
 {  
 Ranges.*setSize*(new Coord (cols, rows));  
 bomb = new Bomb(bombs);  
 flag = new Flag();  
 }  
  
 public void start ()  
 {  
 bomb.start();  
 flag.start();  
 state = GameState.*PLAYED*;  
 }  
  
 public Box getBox (Coord coord)  
 {  
 if (flag.get(coord) == Box.*OPENED*)  
 return bomb.get(coord);  
 else  
 return flag.get(coord);  
 }  
  
 public void pressLeftButton(Coord coord)  
 {  
 if (gameOver ()) return;  
 openBox (coord);  
 checkWinner();  
 }  
  
 private void checkWinner ()  
 {  
 if (state == GameState.*PLAYED*)  
 if (flag.getCountOfClosedBoxes() == bomb.getTotalBombs())  
 state = GameState.*WINNER*;  
 }  
  
 private void openBox(Coord coord)  
 {  
 switch (flag.get(coord))  
 {  
 case *OPENED* : setOpenedToClosedBoxesAroundNumber (coord); return;  
 case *FLAGED* : return;  
 case *CLOSED* :  
 switch (bomb.get (coord))  
 {  
 case *ZERO*: openBoxesAround (coord); return;  
 case *BOMB*: openBombs (coord); return;  
 default : flag.setOpenedToBox(coord); return;  
 }  
 }  
 }  
  
 private void setOpenedToClosedBoxesAroundNumber(Coord coord)  
 {  
 if (bomb.get (coord) != Box.*BOMB*)  
 if (flag.getCountOfFlagedBoxesAround(coord) == bomb.get(coord).getNumber())  
 for (Coord around : Ranges.*getCoordsAround*(coord))  
 if (flag.get (around) == Box.*CLOSED*)  
 openBox (around);  
 }  
  
 private void openBombs(Coord bombed)  
 {  
 state = GameState.*BOMBED*;  
 flag.setBombedToBox (bombed);  
 for (Coord coord : Ranges.*getAllCoords*())  
 if (bomb.get (coord) == Box.*BOMB*)  
 flag.setOpenedToClosedBombBox (coord);  
 else  
 flag.setNobombToFlagedSafeBox (coord);  
 }  
  
 private void openBoxesAround(Coord coord)  
 {  
 flag.setOpenedToBox (coord);  
 for (Coord around : Ranges.*getCoordsAround*(coord))  
 openBox(around);  
 }  
  
 public void pressRightButton(Coord coord)  
 {  
 if (gameOver ()) return;  
 flag.toggleFlagedToBox (coord);  
 }  
  
 private boolean gameOver()  
 {  
 if (state == GameState.*PLAYED*)  
 return false;  
 start();  
 return true;  
 }  
}

**GameState**

package sweeper;  
  
public enum GameState  
{  
 *PLAYED*,  
 *BOMBED*,  
 *WINNER*}

**Matrix**

package sweeper;  
  
class Matrix  
{  
 private Box [] [] matrix;  
  
 Matrix (Box defaultBox)  
 {  
 matrix = new Box[Ranges.*getSize*().x][Ranges.*getSize*().y];  
 for (Coord coord : Ranges.*getAllCoords* ())  
 matrix [coord.x] [coord.y] = defaultBox;  
 }  
  
 Box get (Coord coord)  
 {  
 if (Ranges.*inRange* (coord))  
 return matrix [coord.x] [coord.y];  
 return null;  
 }  
  
 void set (Coord coord, Box box)  
 {  
 if (Ranges.*inRange* (coord))  
 matrix [coord.x] [coord.y] = box;  
 }  
}

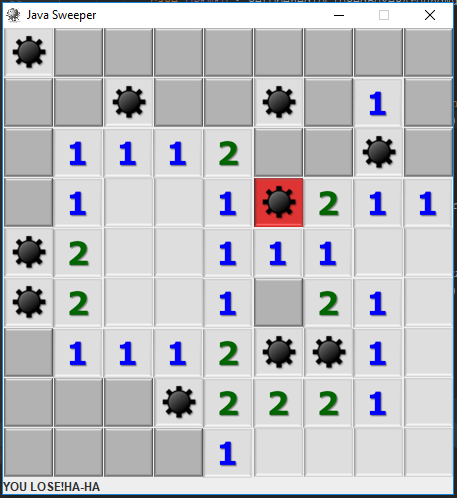
**Ranges**

package sweeper;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
  
public class Ranges  
{  
  
 private static Coord *size*;  
 private static ArrayList<Coord> *allCoords*;  
 private static Random *random* = new Random();  
  
 static void setSize (Coord \_size)  
 {  
 *size* = \_size;  
 *allCoords* = new ArrayList<Coord>();  
 for (int y = 0; y < *size*.y; y ++)  
 for (int x = 0; x< *size*.x; x ++)  
 *allCoords*.add(new Coord (x, y));  
 }  
  
 public static Coord getSize()  
 {  
 return *size*;  
 }  
  
 public static ArrayList<Coord> getAllCoords ()  
 {  
 return *allCoords*;  
 }  
  
 static boolean inRange (Coord coord)  
 {  
 return coord.x >= 0 && coord.x < *size*.x &&  
 coord.y >=0 && coord.y < *size*.y;  
 }  
  
 static Coord getRandomCoord()  
 {  
 return new Coord (*random*.nextInt (*size*.x),  
 *random*.nextInt(*size*.y));  
 }  
  
 static ArrayList<Coord> getCoordsAround(Coord coord)  
 {  
 Coord around;  
 ArrayList<Coord> list = new ArrayList<Coord>();  
 for (int x = coord.x - 1; x <= coord.x + 1; x++)  
 for (int y = coord.y -1; y <= coord.y +1; y++)  
 if (*inRange*(around = new Coord (x,y)))  
 if (!around.equals(coord))  
 list.add (around);  
 return list;  
 }  
}

**JavaSweeper**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
  
import sweeper.Box;  
import sweeper.Coord;  
import sweeper.Game;  
import sweeper.Ranges;  
  
public class JavaSweeper extends JFrame  
{  
 private Game game;  
 private JPanel panel;  
 private JLabel label;  
 private final int COLS = 9;  
 private final int ROWS = 9;  
 private final int BOMBS = 10;  
 private final int IMAGE\_SIZE = 50;  
  
 public static void main(String[] args)  
 {  
 new JavaSweeper();  
 }  
  
 private JavaSweeper ()  
 {  
 game = new Game (COLS, ROWS, BOMBS);  
 game.start();  
 setImages();  
 initLabel();  
 initPanel();  
 initFrame();  
 }  
  
 private void initLabel ()  
 {  
 label = new JLabel("Welcome!");  
 add (label, BorderLayout.*SOUTH*);  
 }  
  
 private void initPanel ()  
 {  
 panel = new JPanel()  
 {  
 @Override  
 protected void paintComponent (Graphics g)  
 {  
 super.paintComponent(g);  
 for (Coord coord : Ranges.*getAllCoords*())  
 {  
 g.drawImage((Image) game.getBox(coord).image, coord.x \* IMAGE\_SIZE, coord.y \* IMAGE\_SIZE, this);  
 }  
 }  
 };  
  
 panel.addMouseListener(new MouseAdapter()  
 {  
 @Override  
 public void mousePressed(MouseEvent e)  
 {  
 int x = e.getX() / IMAGE\_SIZE;  
 int y = e.getY() / IMAGE\_SIZE;  
 Coord coord = new Coord (x, y);  
 if (e.getButton() == MouseEvent.*BUTTON1*)  
 game.pressLeftButton (coord);  
 if (e.getButton() == MouseEvent.*BUTTON3*)  
 game.pressRightButton (coord);  
 if (e.getButton() == MouseEvent.*BUTTON2*)  
 game.start ();  
 label.setText (getMessage ());  
 panel.repaint();  
 }  
 });  
 panel.setPreferredSize(new Dimension(  
 Ranges.*getSize*().x \* IMAGE\_SIZE,  
 Ranges.*getSize*().y \* IMAGE\_SIZE));  
 add (panel);  
 }  
  
 private String getMessage()  
 {  
 switch (game.getState())  
 {  
 case *PLAYED*: return "Think twice!";  
 case *BOMBED*: return "YOU LOSE!HA-HA";  
 case *WINNER*: return "CONGRATULATIONS! YOU WIN!";  
 default : return "Welcome!";  
 }  
 }  
  
 private void initFrame ()  
 {  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 setTitle ("Java Sweeper");  
 setResizable(false);  
 setVisible(true);  
 pack();  
 setLocationRelativeTo(null);  
 setIconImage(getImage("icon"));  
 }  
  
 private void setImages ()  
 {  
 for (Box box : Box.*values*())  
 box.image = getImage(box.name().toLowerCase());  
 }  
  
 private Image getImage (String name)  
 {  
 String filename = "img/" + name + ".png";  
 ImageIcon icon = new ImageIcon (getClass().getResource(filename));  
 return icon.getImage();  
 }  
}

# Тестирование программы



# Руководство по установке

Установить приложение по открытию jar файлов

Скачать файл JavaSweeper.jar с github и запустить его.