МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники



ОТЧЁТ

по лабораторной работе №4

по дисциплине: «Технология программирования»

Тема: Многопотоковые приложения

Вариант №7

Выполнил: Проверил:

Студент гр. АВТ-716 *Михайленко Д. А.*

*Имамов В.М.*

Новосибирск, 2019 г.

**Цели работы:**

Изучить особенности реализации и работы потоков в Java, управлением приоритетами потоков и синхронизацией потоков.

***Вариант 7***

1. Капитальные дома двигаются (в городах будущего и не такое возможно) в левую верхнюю четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке 0;0, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если дом сгенерировался сразу в этой области, то он никуда не движется. По прибытии в конечную точку дом больше не движется.
2. Деревянные дома после генерации начинают двигаться в нижнюю правую четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке w/2;h/2, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если дом сгенерировался сразу в этой области, то он никуда не движется. По прибытии в конечную точку дом больше не движется.

**Задание к лабораторной работе №4**

1. Доработать программу, созданную в лабораторной работе № 3:
2. создать абстрактный класс BaseAI, описывающий «интеллектуальное поведение» объектов. Класс должен создавать поток, обеспечивающий движения объектов коллекции;
3. реализовать класс BaseAI для каждого из видов объекта, включив в него поведение, описанное в индивидуальном задании по варианту;
4. синхронизовать работу потоков расчета интеллекта объектов, их рисования и генерации новых объектов. Рисование должно остаться в основном потоке;
5. добавить в панель управления кнопки для остановки и возобновления работы интеллекта каждого вида объектов. Реализовать через управление монитором (методы wait() и notify());
6. добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

**Пример работы программы**

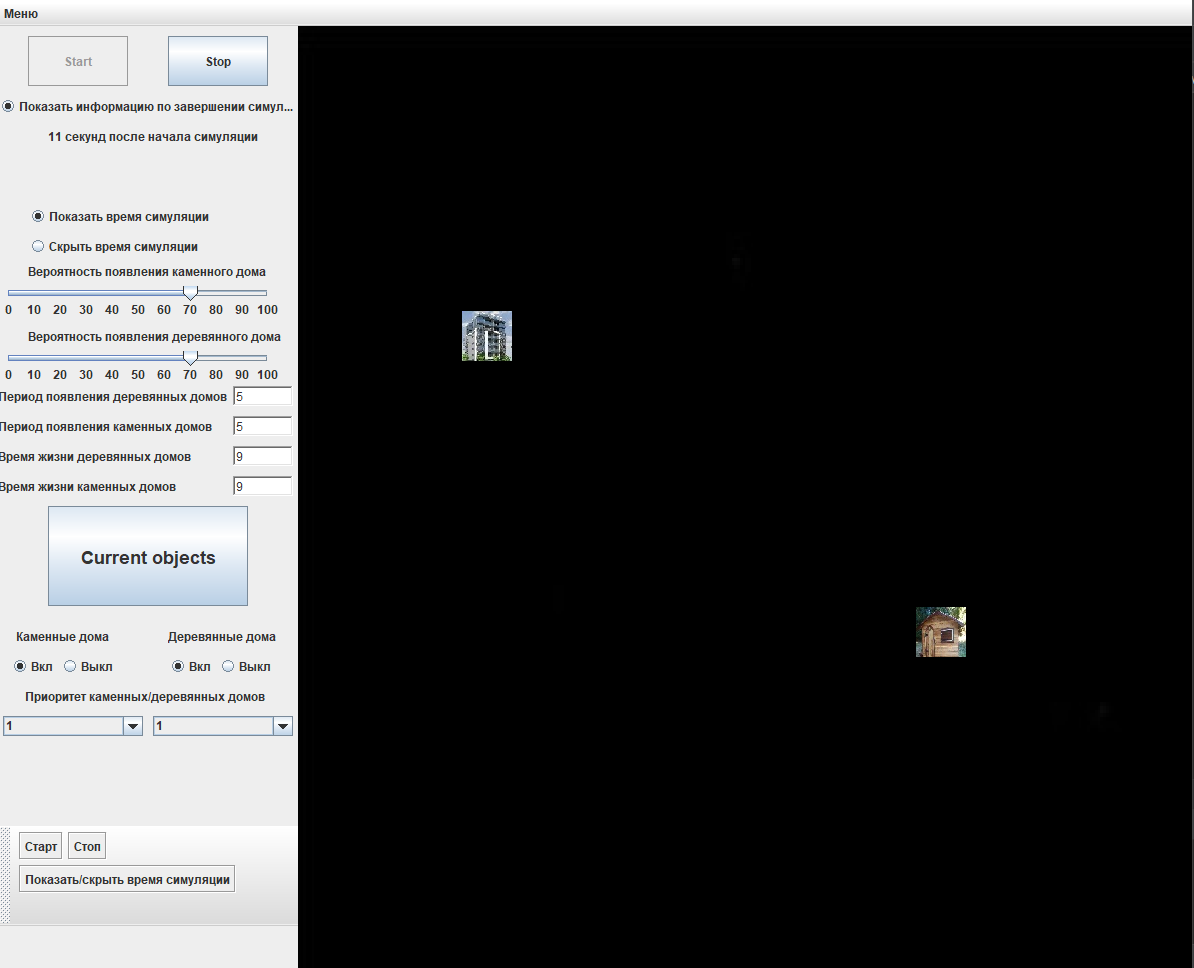


Рис.1 Окно при запуске

Добавлены кнопки для остановки и возобновления работы интеллекта каждого вида объектов (реализующие методы wait() и notify())(рис4). При нажатии на кнопку «Заснуть» нужный поток приостанавливает свою работу, при нажатии кнопки «Проснуться», соответственно, продолжает.

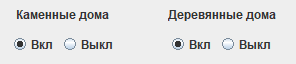


Рис.2 Остановка и возобновление работы интеллекта

На рисунке 3 показаны выпадающие списки для выставления приоритета каждого из потоков, реализующих движение.

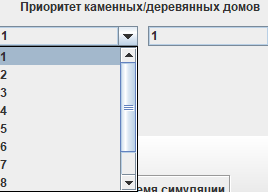
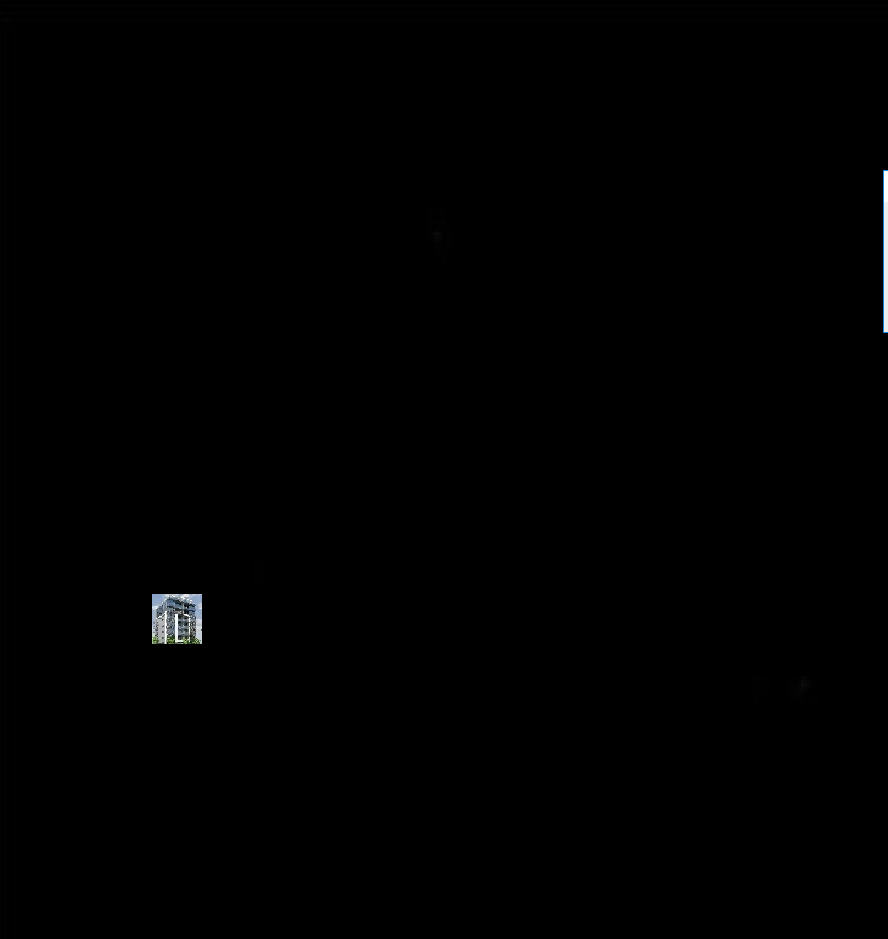


Рис.3 Выпадающие списки

На рисунках ниже представлено движение объектов с течением времени.





**Листинг программы**

Класс-среда Habitat

package Tutorial;  
  
import java.util.\*;  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Habitat {  
  
 volatile static Vector *houses*=new Vector();  
 static HashSet<Integer> *ids*=new HashSet<Integer>();  
 static Map<Integer,Integer> *borns*=new TreeMap<Integer,Integer>();  
 static int *N1*,*N2*,*P1*,*P2*;  
 static int *K1*,*K2*;  
 private ImageIcon imageIcon = new ImageIcon(new ImageIcon(  
 "src/screen.jpg").getImage().getScaledInstance(900,1000, Image.*SCALE\_DEFAULT*));  
  
 Habitat()  
 {  
 JLabel tmp = new JLabel();  
 tmp.setBounds( 0, 0, imageIcon.getIconWidth(), imageIcon.getIconHeight());  
 tmp.setIcon(imageIcon);  
 Frame.*objs*.add(tmp,0,0);  
 Frame.*objs*.repaint();  
 }  
 void update(int t)  
 {  
 synchronized (*houses*) {  
 Random random = new Random();  
 if (t % *N1* == 0) {  
 int i = random.nextInt(101);  
 if (i <= *P1*) {  
 int x = random.nextInt(601) + 250;  
 int y = random.nextInt(851);  
 Wood wood = new Wood(x, y);  
 wood.setcoords();  
 wood.setBorn(t);  
 wood.setTimelife(*K1*);  
 *ids*.add(wood.id);  
 *borns*.put(wood.id, wood.born);  
 *houses*.add(wood);  
 Wood.*woods*++;  
 }  
 }  
 if (t % *N2* == 0) {  
 int j = random.nextInt(101);  
 if (j <= *P2*) {  
 int x = random.nextInt(601) + 250;  
 int y = random.nextInt(851);  
 Stone stone = new Stone(x, y);  
 stone.setcoords();  
 stone.setBorn(t);  
 stone.setTimelife(*K2*);  
 *ids*.add(stone.id);  
 *borns*.put(stone.id, stone.born);  
 *houses*.add(stone);  
 Stone.*stones*++;  
 }  
 }  
 dying(t);  
 }  
 }  
 void dying(int t)  
 {  
 Iterator<Object> hIterator= *houses*.iterator();  
 while(hIterator.hasNext()) {  
 Object current = hIterator.next();  
 if (current instanceof Wood) {  
 Wood nextwood = (Wood) current;  
 if (nextwood.isDead(t)) {  
 Frame.*objs*.remove(nextwood.jl);  
 *ids*.remove(nextwood.getId());  
 *borns*.remove(nextwood.getId());  
 hIterator.remove();  
 Wood.*woods*--;  
 }  
 }  
  
 if (current instanceof Stone) {  
 Stone nextstone = (Stone) current;  
 if (nextstone.isDead(t)) {  
 Frame.*objs*.remove(nextstone.jl);  
 *ids*.remove(nextstone.getId());  
 *borns*.remove(nextstone.getId());  
 hIterator.remove();  
 Stone.*stones*--;  
 }  
 }  
 }  
 }  
  
 static void currentobject(long t)  
 {  
 String currentObjects[]=new String[Habitat.*houses*.capacity()];  
 int k=0;  
 Iterator<Map.Entry<Integer, Integer>> iIterator = *borns*.entrySet().iterator();  
 while(iIterator.hasNext())  
 {  
 Map.Entry<Integer,Integer> kv= iIterator.next();  
 int id = kv.getKey();  
 int born=kv.getValue();  
 currentObjects[k++]=("Тип "+*type*(kv.getKey())+" "+id+" "+born );  
 }  
 JList<String> jlst = new JList<String>(currentObjects);  
 JScrollPane jscrl = new JScrollPane(jlst);  
 int j=JOptionPane.*showConfirmDialog*(Frame.*frame*, jscrl,"Живые",JOptionPane.*OK\_CANCEL\_OPTION*);  
 if(j==JOptionPane.*OK\_OPTION*)  
 {  
 Frame.*habitat*.clear();  
 Frame.*btStop*.setEnabled(false);  
 Frame.*btStart*.setEnabled(true);  
 Frame.*time*=0;  
  
 }  
 else  
 {  
 Frame.*start*=Frame.*start*+(System.*currentTimeMillis*()-t);  
 Frame.*simulation*=true;  
 }  
 Frame.*frame*.requestFocus();  
 }  
 static String type(int i) {  
 Iterator<Object> hIterator = *houses*.iterator();  
 while (hIterator.hasNext()) {  
 Object current = hIterator.next();  
  
 if (current instanceof Wood) {  
 Wood nextwood = (Wood) current;  
 if (nextwood.getId() == i) return "Wood";  
 } else if (current instanceof Stone) {  
 Stone nextstone = (Stone) current;  
 if (nextstone.getId() == i) return "Stone";  
 }  
 }  
 return "";  
 }  
 void resume()  
 {  
 Iterator<Object> hIterator= *houses*.iterator();  
 while(hIterator.hasNext()) {  
 Object current = hIterator.next();  
 if (current instanceof Wood) {  
 Wood nextwood = (Wood) current;  
 nextwood.*woods*++;  
 nextwood.setBorn(0);  
 Frame.*objs*.add(nextwood.jl,0,0);  
 }  
 else  
 {  
 Stone nextstone = (Stone) current;  
 nextstone.*stones*++;  
 nextstone.setBorn(0);  
 Frame.*objs*.add(nextstone.jl,0,0);  
 }  
 }  
 Iterator<Map.Entry<Integer, Integer>> iIterator = *borns*.entrySet().iterator();  
 while(iIterator.hasNext()) {  
 Map.Entry<Integer, Integer> kv = iIterator.next();  
 kv.setValue(0);  
 }  
 Frame.*objs*.repaint();  
 }  
  
 void analyser(String s)  
 {  
 String str1 = "Установить вероятность генерации каменных домов ";  
 String str2 = "Получить вероятность генерации каменных домов";  
 int g = s.length();  
 int a=str2.length();  
 int y = str1.length();  
 String str = "";  
 int i = 0;  
 int j = 0;  
 int k=0;  
 boolean flag = false;  
 switch(1) {  
 case 1: {  
 while (g > y) {  
 if (str1.charAt(i) == s.charAt(i)) {  
 if ((i + 1) == str1.length()) {  
 i++;  
 if (s.charAt(g - 1) != '%') break;  
 while (s.charAt(i) != '%') {  
 str += String.*valueOf*(s.charAt(i));  
 j++;  
 i++;  
 }  
 if (j != 0) {  
 flag = true;  
 break;  
 }  
 }  
 } else {  
  
 flag = false;  
 break;  
 }  
 i++;  
 }  
 if (flag) {  
  
 int perc = Integer.*parseInt*(str);  
 *P2* = perc;  
 Frame.*slider1*.setValue(*P2*);  
 Frame.*field*.append("\n");  
 Frame.*field*.append("Установлена вероятность: " + perc);  
 break;  
 }  
 }  
 case 2:  
 {  
 while (g == a) {  
 if (str2.charAt(k) != s.charAt(k)) {  
 flag = false;  
 break;  
 }  
 k++;  
 if(k==a)  
 {  
 flag=true;  
 break;  
 }  
 }  
 if(flag)  
 {  
 Frame.*field*.append("\n");  
 Frame.*field*.append("Вероятность появления: " + *P2*);  
 break;  
 }  
 }  
 default:  
 {  
 if (!flag) {  
 Frame.*field*.append("\n");  
 Frame.*field*.append("Некорректная команда");  
 }  
 break;  
 }  
 }  
 }  
  
  
 void clear()  
 {  
 *ids*.clear();  
 *borns*.clear();  
 *houses*.clear();  
 Frame.*objs*.removeAll();  
 Wood.*woods*=Stone.*stones*=0;  
 new Habitat();  
 }  
}

Класс Frame:

package Tutorial;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
  
public class Frame {  
 static public JPanel *objs*=new JPanel();  
 static public JFrame *frame*=new JFrame("Labs");  
 static public JPanel *panel*=new JPanel();  
 static Habitat *habitat*;  
  
 static public int *WIDTH*=1200,*HEIGHT*=1000;  
 static int *t*;  
 static long *time*=0,*start*=0;  
  
 static boolean *simulation*=false;  
 static boolean *text*=true;  
 static boolean *bt*=true;  
 static String *TextArea*[]=new String[4];  
  
 static JLabel *textTimeSimulation* = new JLabel(""),  
 *percent1*=new JLabel("Вероятность появления каменного дома"),*percent2*=new JLabel("Вероятность появления деревянного дома"),  
 *period1*=new JLabel("Период появления деревянных домов"),*period2*=new JLabel("Период появления каменных домов"),  
 *period3*=new JLabel("Время жизни деревянных домов"),*period4*=new JLabel("Время жизни каменных домов"),  
 *thread1*=new JLabel("Каменные дома"),*thread2*=new JLabel("Деревянные дома"),  
 *pr*=new JLabel("Приоритет каменных/деревянных домов");  
 static TextField *field1*=new TextField(""),*field2*=new TextField(""),  
 *field3*=new TextField(""),*field4*=new TextField("");  
 static JButton *btStart* = new JButton("Start"), *btStop* = new JButton("Stop"),*current*=new JButton("<html><center><font size=\"5\" >Current objects</font>");  
 static JRadioButton *timeOn* = new JRadioButton("Показать время симуляции"),  
 *timeOff* = new JRadioButton("Скрыть время симуляции"),  
 *info*=new JRadioButton("Показать информацию по завершении симуляции"),  
 *threadon1*=new JRadioButton("Вкл"),*threadoff1*=new JRadioButton("Выкл"),  
 *threadon2*=new JRadioButton("Вкл"),*threadoff2*=new JRadioButton("Выкл");  
 static JSlider *slider1*= new JSlider(0,100),*slider2*=new JSlider(0,100);  
  
 static JMenuBar *jMenuBar* = new JMenuBar();  
 static JMenu *jmCommands* = new JMenu("Меню");  
 static JMenuItem *jmiStart* = new JMenuItem("Старт"), *jmiStop* = new JMenuItem("Стоп"), *jmiTime* = new JMenuItem("Показать/скрыть время симуляции"),  
 *jmiConsol*=new JMenuItem("Консоль"),*jmiSave*=new JMenuItem("Сохранить"),*jmiLoad*=new JMenuItem("Загрузить");  
 static JToolBar *jToolBar* = new JToolBar();  
 static JButton *jtbStart* = new JButton("Старт"), *jtbStop* = new JButton("Стоп"), *jtbTime* = new JButton("Показать/скрыть время симуляции");  
  
 static String[] *priority*= {"1","2","3","4","5","6","7","8","9","10"};  
 static JComboBox *jcb1*=new JComboBox(*priority*);  
 static JComboBox *jcb2*=new JComboBox(*priority*);  
  
 static JDialog *consol*=new JDialog(*frame*,"Консоль",false);  
 static JTextArea *field*=new JTextArea();  
  
 Configure conf=new Configure();  
  
 Frame()  
 {  
 try {  
 conf.reading();  
 *field1*.setText(Integer.*toString*(Habitat.*N1*));  
 *field2*.setText(Integer.*toString*(Habitat.*N2*));  
 *field3*.setText(Integer.*toString*(Habitat.*K1*));  
 *field4*.setText(Integer.*toString*(Habitat.*K2*));  
 *slider1*.setValue(Habitat.*P2*);  
 *slider2*.setValue(Habitat.*P1*);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 *frame*.setLayout(null);  
 *frame*.setSize(*WIDTH*,*HEIGHT*);  
 *frame*.addWindowListener(new WindowAdapter(){  
 public void windowClosing(WindowEvent e) {  
 try {  
 conf.writing();  
 } catch (Exception e1) {  
 e1.printStackTrace();  
 }  
 System.*exit*(0);  
 }  
 });  
 *frame*.setVisible(true);  
 *frame*.requestFocus();  
  
 *objs*.setLayout(null);  
 *objs*.setBounds(300,0,*WIDTH*,*HEIGHT*);  
  
 *panel*.setLayout(null);  
 *panel*.setBounds(0,0,300,*HEIGHT*);  
 *textTimeSimulation*.setBounds(50,100, 300, 20);  
 *panel*.add(*textTimeSimulation*);  
  
 *consol*.setSize(500,300);  
 *field*.setBounds(0,0,500,300);  
 *consol*.add(*field*);  
  
 *btStart*.setBounds(30, 10, 100, 50);  
 *panel*.add(*btStart*);  
 *btStop*.setBounds(170, 10, 100, 50);  
 *panel*.add(*btStop*);  
 *btStop*.setEnabled(false);  
 *current*.setPreferredSize(new Dimension(200,100));  
 *current*.setBounds(50,480,200,100);  
 *panel*.add(*current*);  
 *info*.setBounds(0,70,300,20);  
 *panel*.add(*info*);  
 *timeOn*.setBounds(30,180,200,20);  
 *panel*.add(*timeOn*);  
 *timeOff*.setBounds(30,210,200,20);  
 *panel*.add(*timeOff*);  
 ButtonGroup buttonGroup1 = new ButtonGroup();  
 buttonGroup1.add(*timeOn*);  
 buttonGroup1.add(*timeOff*);  
 *thread1*.setBounds(18,590,140,40);  
 *panel*.add(*thread1*);  
 *threadon1*.setBounds(12,630,50,20);  
 *panel*.add(*threadon1*);  
 *threadoff1*.setBounds(62,630,60,20);  
 *panel*.add(*threadoff1*);  
 ButtonGroup buttonGroup2 = new ButtonGroup();  
 buttonGroup2.add(*threadon1*);  
 buttonGroup2.add(*threadoff1*);  
 *thread2*.setBounds(170,590,140,40);  
 *panel*.add(*thread2*);  
 *threadon2*.setBounds(170,630,50,20);  
 *panel*.add(*threadon2*);  
 *threadoff2*.setBounds(220,630,60,20);  
 *panel*.add(*threadoff2*);  
 ButtonGroup buttonGroup3 = new ButtonGroup();  
 buttonGroup3.add(*threadon2*);  
 buttonGroup3.add(*threadoff2*);  
 *percent1*.setBounds(30,235,300,20);  
 *panel*.add(*percent1*);  
 *slider1*.setBounds(0,260,280,30);  
 *slider1*.setMajorTickSpacing(10);  
 *slider1*.setPaintLabels(true);  
 *slider1*.setSnapToTicks(true);  
 *panel*.add(*slider1*);  
 *percent2*.setBounds(30,300,300,20);  
 *panel*.add(*percent2*);  
 *slider2*.setBounds(0,325,280,30);  
 *slider2*.setMajorTickSpacing(10);  
 *slider2*.setPaintLabels(true);  
 *slider2*.setSnapToTicks(true);  
 *panel*.add(*slider2*);  
 *period1*.setBounds(0,360,230,20);  
 *panel*.add(*period1*);  
 *field1*.setBounds(235,360,60,20);  
 *panel*.add(*field1*);  
 *period2*.setBounds(0,390,230,20);  
 *panel*.add(*period2*);  
 *field2*.setBounds(235,390,60,20);  
 *panel*.add(*field2*);  
  
 *period3*.setBounds(0,420,230,20);  
 *panel*.add(*period3*);  
 *field3*.setBounds(235,420,60,20);  
 *panel*.add(*field3*);  
 *period4*.setBounds(0,450,230,20);  
 *panel*.add(*period4*);  
 *field4*.setBounds(235,450,60,20);  
 *panel*.add(*field4*);  
  
 *pr*.setBounds(27,660,300,20);  
 *panel*.add(*pr*);  
 *jcb1*.setBounds(5,690,140,20);  
 *panel*.add(*jcb1*);  
 *jcb2*.setBounds(155,690,140,20);  
 *panel*.add(*jcb2*);  
  
  
 *jmCommands*.add(*jmiStart*);  
 *jmCommands*.add(*jmiStop*);  
 *jmCommands*.add(*jmiTime*);  
 *jmCommands*.add(*jmiConsol*);  
 *jmCommands*.add(*jmiSave*);  
 *jmCommands*.add(*jmiLoad*);  
 *jMenuBar*.add(*jmCommands*);  
  
 *jToolBar*.add(*jtbStart*);  
 *jToolBar*.add(*jtbStop*);  
 *jToolBar*.add(*jtbTime*);  
 *jToolBar*.setLayout(new FlowLayout(FlowLayout.*LEADING*));  
 *jToolBar*.setBounds(0, 800, 300, 100);  
 *panel*.add(*jToolBar*);  
  
 *frame*.add(*objs*);  
 *frame*.add(*panel*);  
 *frame*.setJMenuBar(*jMenuBar*);  
 *frame*.setResizable(false);  
  
 *timeOn*.setSelected(true);  
 *info*.setSelected(true);  
 *threadon1*.setSelected(true);  
 *threadon2*.setSelected(true);  
  
 Listeners listen=new Listeners();  
  
 *habitat*=new Habitat();  
 }  
  
 static void time\_simulation( ){  
 if(*text*){  
 *textTimeSimulation*.setText((*time*/1000)+" секунд после начала симуляции");  
 }else {  
 *textTimeSimulation*.setText("");  
 }  
 *panel*.repaint();  
 }  
}

Класс Listeners:

package Tutorial;  
  
import javax.swing.\*;  
import javax.swing.text.BadLocationException;  
import java.awt.event.\*;  
import java.io.\*;  
  
public class Listeners  
{  
 Listeners()  
 {  
 Frame.*btStart*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 *func\_B*();  
 Frame.*frame*.requestFocus();  
 }  
 });  
  
 Frame.*frame*.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyPressed(KeyEvent e) {  
  
 if(e.getKeyCode()== KeyEvent.*VK\_T*)  
 {  
 *func\_T*();  
 if(Frame.*text*)  
 Frame.*timeOn*.setSelected(true);  
 else  
 Frame.*timeOff*.setSelected(true);  
 }  
 if(e.getKeyCode()== KeyEvent.*VK\_E*)  
 {  
 if(Frame.*simulation*)  
 *func\_E*();  
 }  
 if(e.getKeyCode()== KeyEvent.*VK\_B*)  
 {  
 if(!Frame.*simulation*)  
 *func\_B*();  
 }  
 }  
 });  
  
 Frame.*field*.addKeyListener( new KeyAdapter(){  
 @Override  
 public void keyPressed(KeyEvent e)  
 {  
 if(e.getKeyCode()== KeyEvent.*VK\_ENTER*)  
 {  
 String s=getcurrentText();  
 Frame.*habitat*.analyser(s);  
 //System.out.println(s);  
 /\*i\*//\*nt ch;  
 InputStream ipSt = textAreaInputStream( Frame.field );  
 try{  
 while( (ch = ipSt.read()) > 0 )  
 opSt.print( (char)ch );  
 }catch(Exception \_) { }\*/  
 }  
 }  
 });  
  
 Frame.*btStop*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 *func\_E*();  
 Frame.*frame*.requestFocus();  
  
 }  
 });  
  
 Frame.*timeOn*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Frame.*text* = true;  
 }  
 });  
  
 Frame.*timeOff*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Frame.*text* = false;  
 }  
 });  
  
 Frame.*threadon1*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Starter.*st*.wakeup();  
 Starter.*st*.stop=false;  
 }  
 });  
  
 Frame.*threadoff1*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Starter.*st*.stop=true;  
 }  
 });  
  
 Frame.*threadon2*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Starter.*wo*.wakeup();  
 Starter.*wo*.stop=false;  
 }  
 });  
  
 Frame.*threadoff2*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Starter.*wo*.stop=true;  
 /\*Iterator<Object> hIterator= Habitat.houses.iterator();  
 while(hIterator.hasNext()) {  
 Object current = hIterator.next();  
 if(current instanceof Wood)  
 {  
 Wood nextwood = (Wood) current;  
 synchronized (nextwood) {  
 try {  
 nextwood.wait();  
 } catch (InterruptedException e1) {  
 }  
 }  
 }  
 }\*/  
 }});  
  
 Frame.*slider1*.addChangeListener(e -> {  
 Frame.*frame*.requestFocus();  
 Habitat.*P2*=Frame.*slider1*.getValue();  
 });  
  
 Frame.*slider2*.addChangeListener(e -> {  
 Frame.*frame*.requestFocus();  
 Habitat.*P1*=Frame.*slider2*.getValue();  
 });  
  
 Frame.*field1*.addFocusListener(new FocusListener() {  
 @Override  
 public void focusGained(FocusEvent e) {  
 Frame.*field1*.setText("");  
 }  
  
 @Override  
 public void focusLost(FocusEvent e) {  
  
 }  
 });  
  
 Frame.*field1*.addActionListener(e -> {  
 Frame.*frame*.requestFocus();  
 int k = Integer.*parseInt*(Frame.*field1*.getText());  
 if(k > 0){  
 Habitat.*N1* = k;  
 JOptionPane.*showMessageDialog*(Frame.*frame*, "Установлен период появления деревянных домов равный "+Habitat.*N1*+" секунд");  
 }else{  
 Habitat.*N1* = 1;  
 JOptionPane.*showMessageDialog*(Frame.*frame*, "Неверное значение" +  
 "\nУстановлено значение по умолчанию = 1.");  
 }  
 }  
 );  
  
 Frame.*field2*.addFocusListener(new FocusListener() {  
 @Override  
 public void focusGained(FocusEvent e) {  
 Frame.*field2*.setText("");  
 }  
  
 @Override  
 public void focusLost(FocusEvent e) {  
  
 }  
 });  
  
 Frame.*field2*.addActionListener(e -> {  
 Frame.*frame*.requestFocus();  
 int k = Integer.*parseInt*(Frame.*field2*.getText());  
 if(k > 0){  
 Habitat.*N2* = k;  
 JOptionPane.*showMessageDialog*(Frame.*frame*, "Установлен период появления каменных домов равный "+Habitat.*N2*+" секунд");  
 }else{  
 Habitat.*N2* = 1;  
 JOptionPane.*showMessageDialog*(Frame.*frame*, "Неверное значение" +  
 "\nУстановлено значение по умолчанию = 1.");  
 }  
 }  
 );  
  
 Frame.*field3*.addFocusListener(new FocusListener() {  
 @Override  
 public void focusGained(FocusEvent e) {  
 Frame.*field3*.setText("");  
 }  
  
 @Override  
 public void focusLost(FocusEvent e) {  
  
 }  
 });  
  
 Frame.*field3*.addActionListener(e -> {  
 Frame.*frame*.requestFocus();  
 int k = Integer.*parseInt*(Frame.*field3*.getText());  
 if(k > 0){  
 Habitat.*K1* = k;  
 JOptionPane.*showMessageDialog*(Frame.*frame*, "Установлен период появления деревянных домов равный "  
 +Habitat.*K1*+" секунд");  
 }else{  
 Habitat.*K1* = 10;  
 JOptionPane.*showMessageDialog*(Frame.*frame*, "Неверное значение" +  
 "\nУстановлено значение по умолчанию = 10.");  
 }  
 }  
 );  
  
 Frame.*field4*.addFocusListener(new FocusListener() {  
 @Override  
 public void focusGained(FocusEvent e) {  
 Frame.*field4*.setText("");  
 }  
  
 @Override  
 public void focusLost(FocusEvent e) {  
  
 }  
 });  
  
 Frame.*field4*.addActionListener(e -> {  
 Frame.*frame*.requestFocus();  
 int k = Integer.*parseInt*(Frame.*field4*.getText());  
 if(k > 0){  
 Habitat.*K2* = k;  
 JOptionPane.*showMessageDialog*(Frame.*frame*, "Установлен период появления каменных домов равный "  
 +Habitat.*K2*+" секунд");  
 }else{  
 Habitat.*K2* = 10;  
 JOptionPane.*showMessageDialog*(Frame.*frame*, "Неверное значение" +  
 "\nУстановлено значение по умолчанию = 10.");  
 }  
 }  
 );  
  
 Frame.*current*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Frame.*simulation*=false;  
 long stop=System.*currentTimeMillis*();  
 Habitat.*currentobject*(stop);  
 Frame.*frame*.requestFocus();  
 }  
 });  
  
 Frame.*jmiStart*.addActionListener(e -> {  
 *func\_B*();  
 Frame.*frame*.requestFocus();  
 });  
 Frame.*jmiStop*.addActionListener(e -> {  
 *func\_E*();  
 Frame.*frame*.requestFocus();  
 });  
 Frame.*jmiTime*.addActionListener(e -> {  
 *func\_T*();  
 Frame.*frame*.requestFocus();  
 if(Frame.*text*)  
 Frame.*timeOn*.setSelected(true);  
 else  
 Frame.*timeOff*.setSelected(true);  
 });  
 Frame.*jmiConsol*.addActionListener(e -> {  
 Frame.*consol*.setVisible(true);  
 Frame.*frame*.requestFocus();  
 });  
 Frame.*jmiSave*.addActionListener(e -> {  
 SavedApp sap=new SavedApp(Habitat.*houses*,Habitat.*ids*,Habitat.*borns*);  
 try {  
 JFileChooser fc = new JFileChooser("src/");  
 fc.showSaveDialog(Frame.*frame*);  
 File sf = fc.getSelectedFile();  
 if(sf!= null) {  
 FileOutputStream fileOutputStream = new FileOutputStream(sf);  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);  
 objectOutputStream.writeObject(sap);  
 fileOutputStream.close();  
 objectOutputStream.close();  
 }  
 }  
 catch(Exception e1){e1.printStackTrace();}  
 });  
 Frame.*jmiLoad*.addActionListener(e -> {  
 try {  
 JFileChooser fc = new JFileChooser("src/");  
 fc.showOpenDialog(Frame.*frame*);  
 File lf = fc.getSelectedFile();  
 if(lf!=null) {  
 if(Frame.*simulation*==true)  
 Frame.*habitat*.clear();  
 FileInputStream fileInputStream = new FileInputStream(lf);  
 ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);  
  
 SavedApp sap = (SavedApp) objectInputStream.readObject();  
 sap.Loading();  
 Frame.*habitat*.resume();  
 }  
 }  
 catch(Exception e1){e1.printStackTrace();}  
  
 });  
  
 Frame.*jtbStart*.addActionListener(e -> {  
 *func\_B*();  
 Frame.*frame*.requestFocus();  
 });  
 Frame.*jtbStop*.addActionListener(e -> {  
 *func\_E*();  
 Frame.*frame*.requestFocus();  
 });  
 Frame.*jtbTime*.addActionListener(e -> {  
 *func\_T*();  
 Frame.*frame*.requestFocus();  
 if(Frame.*text*)  
 Frame.*timeOn*.setSelected(true);  
 else  
 Frame.*timeOff*.setSelected(true);  
 });  
  
 Frame.*jcb1*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JComboBox box = (JComboBox)e.getSource();  
 String item = (String)box.getSelectedItem();  
 Starter.*st*.setPriority(Integer.*parseInt*(item));  
 System.*out*.println(Starter.*st*.getPriority());  
 }  
 });  
 Frame.*jcb2*.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JComboBox box = (JComboBox)e.getSource();  
 String item = (String)box.getSelectedItem();  
 Starter.*wo*.setPriority(Integer.*parseInt*(item));  
 System.*out*.println(Starter.*wo*.getPriority());  
 }  
 });  
 }  
 static void func\_B()  
 {  
 Frame.*frame*.requestFocus();  
 Frame.*btStart*.setEnabled(false);  
 Frame.*btStop*.setEnabled(true);  
 //Wood.woods=Stone.stones=0;  
 //Frame.habitat.clear();  
 Frame.*start*=System.*currentTimeMillis*();  
 Frame.*t*=0;  
 Starter.*wo*.prev=0;  
 Starter.*creat*.prev=0;  
 Starter.*st*.prev=0;  
 Starter.*wo*.moving=true;  
 Starter.*st*.moving=true;  
 Frame.*simulation*=true;  
 }  
 static void func\_E()  
 {  
 Frame.*simulation*=false;  
 long stop=System.*currentTimeMillis*();  
 if(Frame.*info*.isSelected())  
 {  
 *results*();  
 int result= JOptionPane.*showConfirmDialog*(Frame.*frame*,  
 Frame.*TextArea*,  
 "Message",  
 JOptionPane.*OK\_CANCEL\_OPTION*);  
 if(result==JOptionPane.*OK\_OPTION*)  
 {  
 Frame.*habitat*.clear();  
 Frame.*btStop*.setEnabled(false);  
 Frame.*btStart*.setEnabled(true);  
 Frame.*time*=0;  
 }  
 else  
 {  
 Frame.*start*=Frame.*start*+(System.*currentTimeMillis*()-stop);  
 Frame.*simulation*=true;  
 }  
 }  
 else  
 {  
 Frame.*btStop*.setEnabled(false);  
 Frame.*btStart*.setEnabled(true);  
 Frame.*time*=0;  
 Frame.*habitat*.clear();  
 }  
 }  
  
 static void func\_T()  
 {  
 Frame.*text*=!Frame.*text*;  
 }  
  
 static void results()  
 {  
 Frame.*TextArea*[0]=("Количество деревянных домов " + Wood.*woods*);  
 Frame.*TextArea*[1]=("Количество каменных домов " + Stone.*stones*);  
 Frame.*TextArea*[2]=("Всего объектов " + (Stone.*stones* + Wood.*woods*));  
 Frame.*TextArea*[3]=("Время симуляции " + (Frame.*time* / 1000));  
 }  
  
 public String getcurrentText()  
 {  
 try {  
 int offset=Frame.*field*.getLineOfOffset(Frame.*field*.getCaretPosition());  
 int start=Frame.*field*.getLineStartOffset(offset);  
 int end=Frame.*field*.getLineEndOffset(offset);  
 return Frame.*field*.getText(start, (end-start));  
 } catch (BadLocationException ex) {  
 System.*out*.println(ex.getMessage());  
 return "-1";  
 }  
 }  
}

Класс Creator:

package Tutorial;  
  
public class Creator extends Thread {  
 long prev;  
 synchronized public void run() {  
 while(true)  
 {  
 Frame.*time\_simulation*();  
 if(Frame.*simulation*==true)  
 {  
 Frame.*time*=System.*currentTimeMillis*()-Frame.*start*;  
 long step=Frame.*time*-prev;  
 if(step>=1000)  
 {  
 Frame.*t* = (int) (Frame.*time* / 1000);  
 Frame.*habitat*.update(Frame.*t*);  
 this.prev = Frame.*time*;  
 }  
 }  
 try{  
 Thread.*sleep*(10);  
 }  
 catch (InterruptedException e){}  
 }  
 }  
}

Класс Starter:

package Tutorial;  
  
public class Starter {  
 static Creator *creat*=new Creator();  
 static WoodAI *wo*=new WoodAI();  
 static StoneAI *st*=new StoneAI();  
 static void run()  
 {  
 *st*.start();  
 *creat*.start();  
 *wo*.start();  
 *st*.setPriority(1);  
 *wo*.setPriority(1);  
 }  
}

Класс BaseAI:

package Tutorial;  
  
public abstract class BaseAI extends Thread {  
 boolean moving=false;  
 long prev;  
 int V=400;  
 boolean stop=false;  
  
 synchronized public void run()  
 {  
 while(true)  
 {  
 if(this.stop)  
 {  
 try{  
 this.wait();  
 }  
 catch (InterruptedException e){}  
 }  
 if(this.moving)  
 {  
 long step=Frame.*time*-prev;  
 if(step>=20)  
 {  
 this.mover();  
 this.prev = Frame.*time*;  
 }  
 }  
 try{  
 Thread.*sleep*(1);  
 }  
 catch (InterruptedException e){}  
  
 }  
 }  
 void mover()  
 { }  
}

Класс WoodAI:

package Tutorial;  
  
import java.util.Iterator;  
  
public class WoodAI extends BaseAI {  
 synchronized void mover() {  
 Iterator<Object> hIterator = Habitat.*houses*.iterator();  
 while (hIterator.hasNext()) {  
 Object current = hIterator.next();  
 if (current instanceof Wood) {  
 Wood nextwood = (Wood) current;  
 nextwood.movexy(V);  
 }  
 }  
 }  
 synchronized public void wakeup()  
 {  
 synchronized (Wood.class)  
 {  
 notify();  
 }  
 }  
}

Класс StoneAI:

package Tutorial;  
  
import java.util.Iterator;  
  
public class StoneAI extends BaseAI {  
 synchronized void mover() {  
 Iterator<Object> hIterator = Habitat.*houses*.iterator();  
 while (hIterator.hasNext()) {  
 Object current = hIterator.next();  
 if (current instanceof Stone) {  
 Stone nextstone = (Stone) current;  
 nextstone.movexy(V);  
 }  
 }  
 }  
 synchronized public void wakeup()  
 {  
 synchronized (Wood.class)  
 {  
 notify();  
 }  
 }  
}