МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра вычислительной техники

Отчет по лабораторной работе №1-2

по дисциплине «Технология программирования»

Вариант №7

Выполнил:

студент группы АВТ-716

Имамов В.М.

Проверил:

Михайленко Д.А.

Новосибирск, 2019 г.

#### Лабораторная работа 1

**Темы**: ***Основы программирования на Java.***

***Обработка событий. Механизм делегирования событий.***

#### Практические задания

1. Познакомиться с особенностями технологии Java и изучить синтаксис языка Java.
2. Изучить основные понятия и термины обработки событий в модели делегирования событий на Java (учебно-методическое пособие, главы 1 и 2).
3. Разработать учебную программу. Основная ее задача – разработка упрощенной имитации поведения объектов (все последующие лабораторные работы будут расширять это задание). Объекты реализуются через наследование: абстрактный класс + интерфейс → наследники.

Рабочий цикл программы:

* запускается процесс симуляции по клавише, генерируются объекты классов согласно заданию;
* симуляция завершается по другой клавише, выводится статистическая информация.

1. Для решения задачи:

* Разработать абстрактный класс объекта, согласно варианту индивидуального задания.
* Создать интерфейс IBehaviour, задающий поведение объекта (методы: move(), getx(), gety(), sety(), setx() и другие. Далее будут реализоваться алгоритмы движения объектов в окне программы).
* Реализовать иерархию классов, определяющих объекты по варианту и реализующие интерфейс IBehaviour.
* Создать класс Habitat (среда), определяющий размер рабочей области и хранящий массив объектов, с параметрами, заданными вариантом. Предусмотреть в классе метод Update, вызывающийся по таймеру и получающий на вход время, прошедшее от начала симуляции. В данном методе должны генерироваться новые объекты и помещаться в поле визуализации в случайном месте. Визуализация объекта - использовать готовые небольшие картинки;

Рабочее окно программы – область визуализации среды обитания объектов;

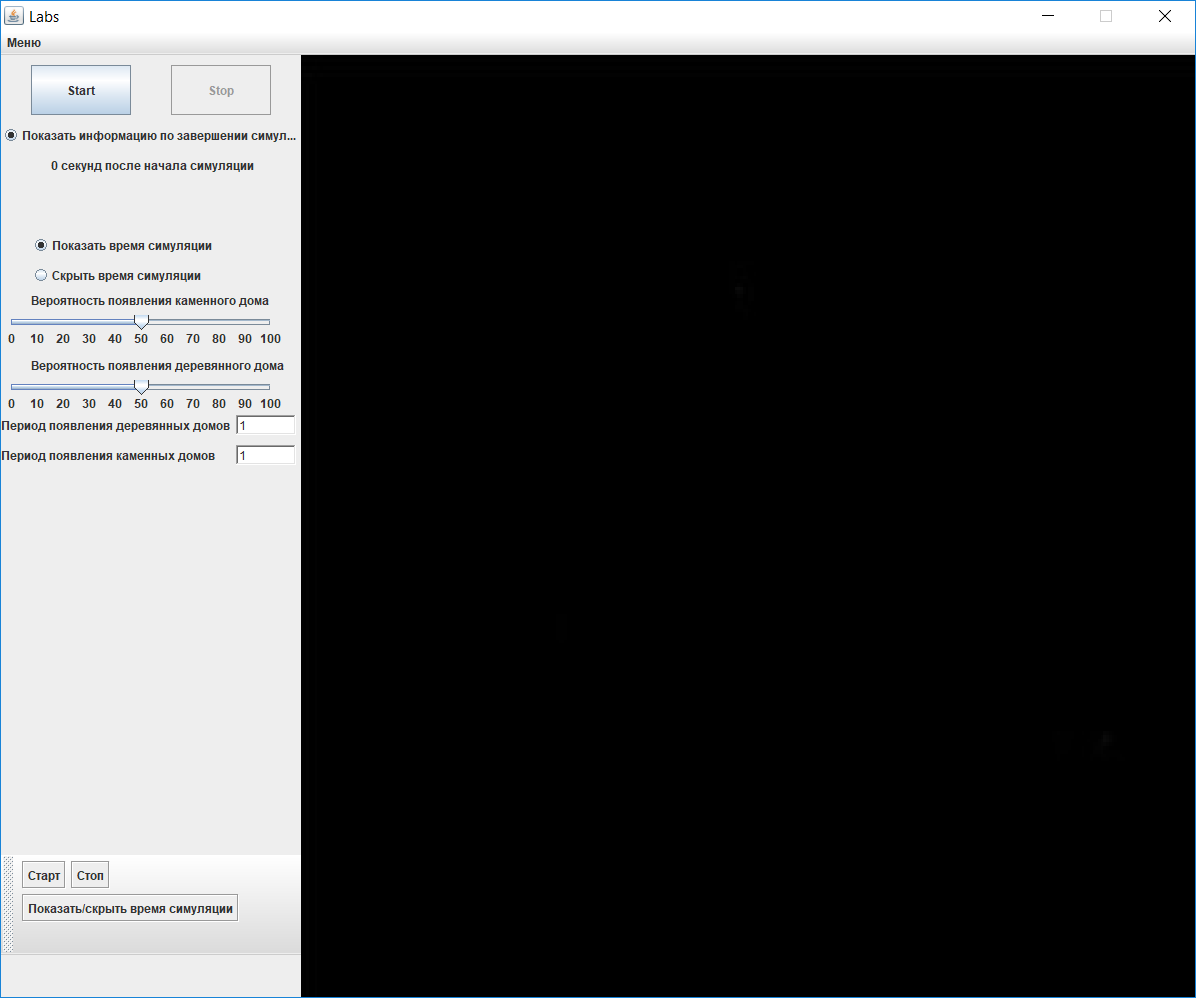
1. Симуляция должна запускаться по клавише **B** и останавливаться по клавише **E**. При остановке симуляции список очищается. Время симуляции должно отображаться текстом в области визуализации и скрываться/показываться по клавише **T**;
2. По завершению симуляции область отображения объектов очищается и в поле визуализации выводится информация о количестве и типе сгенерированных объектов, а также время симуляции. Текст должен быть форматирован, т.е. выводиться с использованием разных шрифтов и цветов.
3. Параметры симуляции задаются в классе Habitat.

***Вариант 7***

Список жилых домов города состоит из двух типов: капитальный, деревянный. Капитальные дома генерируются каждые N1 секунд с вероятностью P1. Деревянные дома генерируются каждые N2 секунд с вероятностью P2.

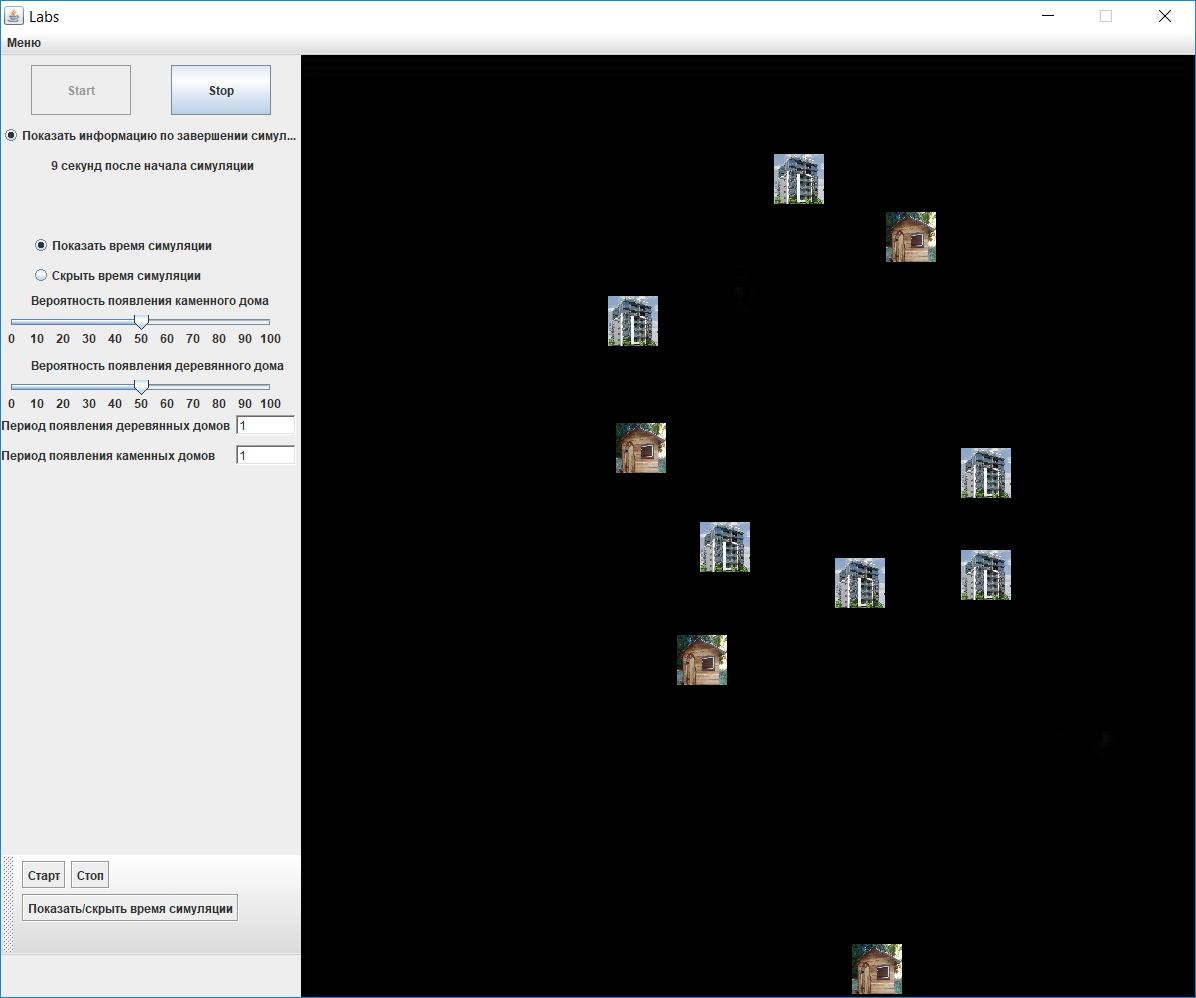
**Результаты работы программы:**

При запуске программы пользователь видит данное окно.

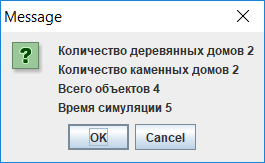


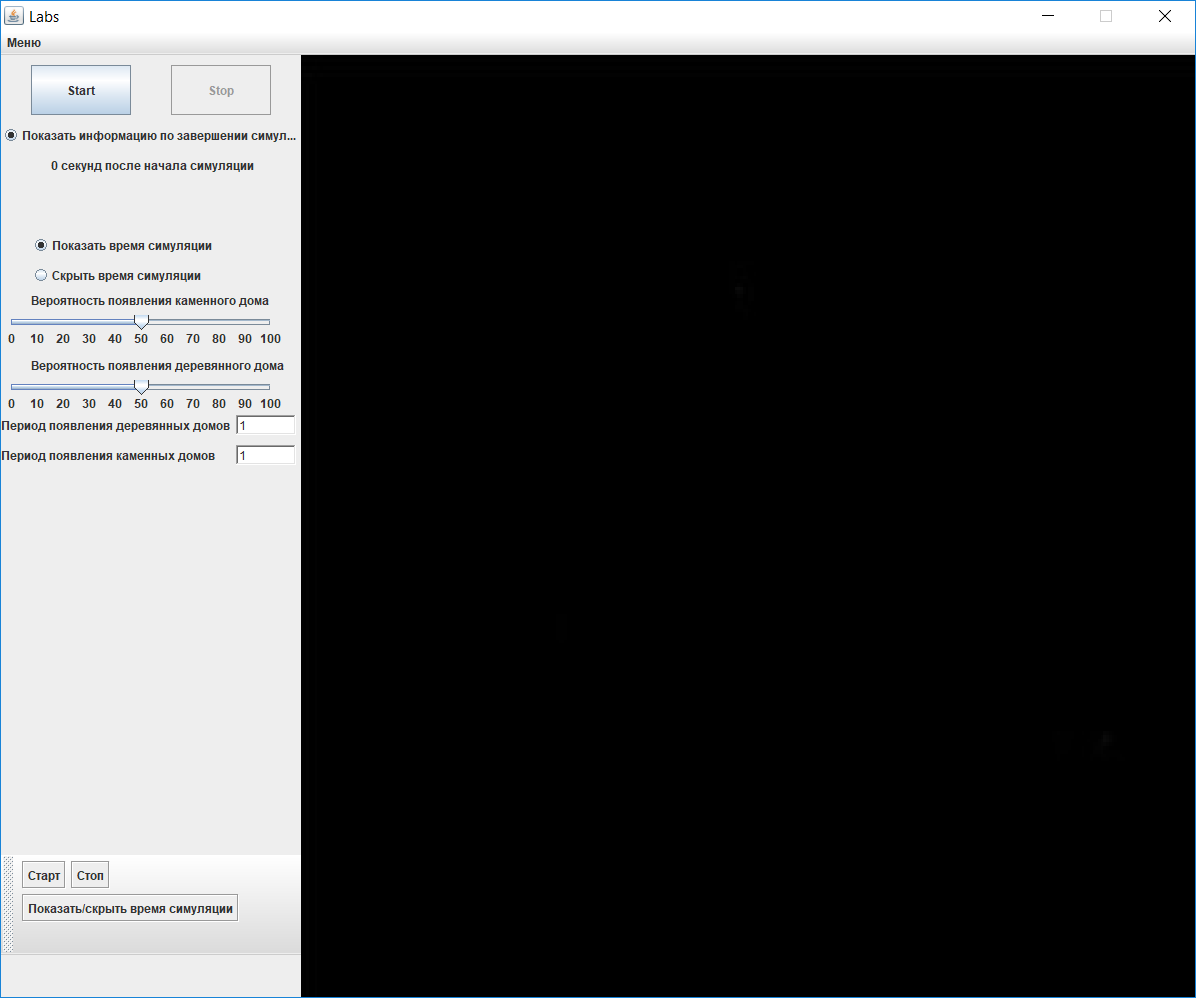
В рамках работы были созданы слушатели клавиш B,E и T, которые отвечают за определенные заданием функции.

Например, при нажатии на кнопку B происходит следующее



А при нажатии на клавишу Е происходит очистка окна визуализации и выводится информация о симуляции





#### Лабораторная работа 2

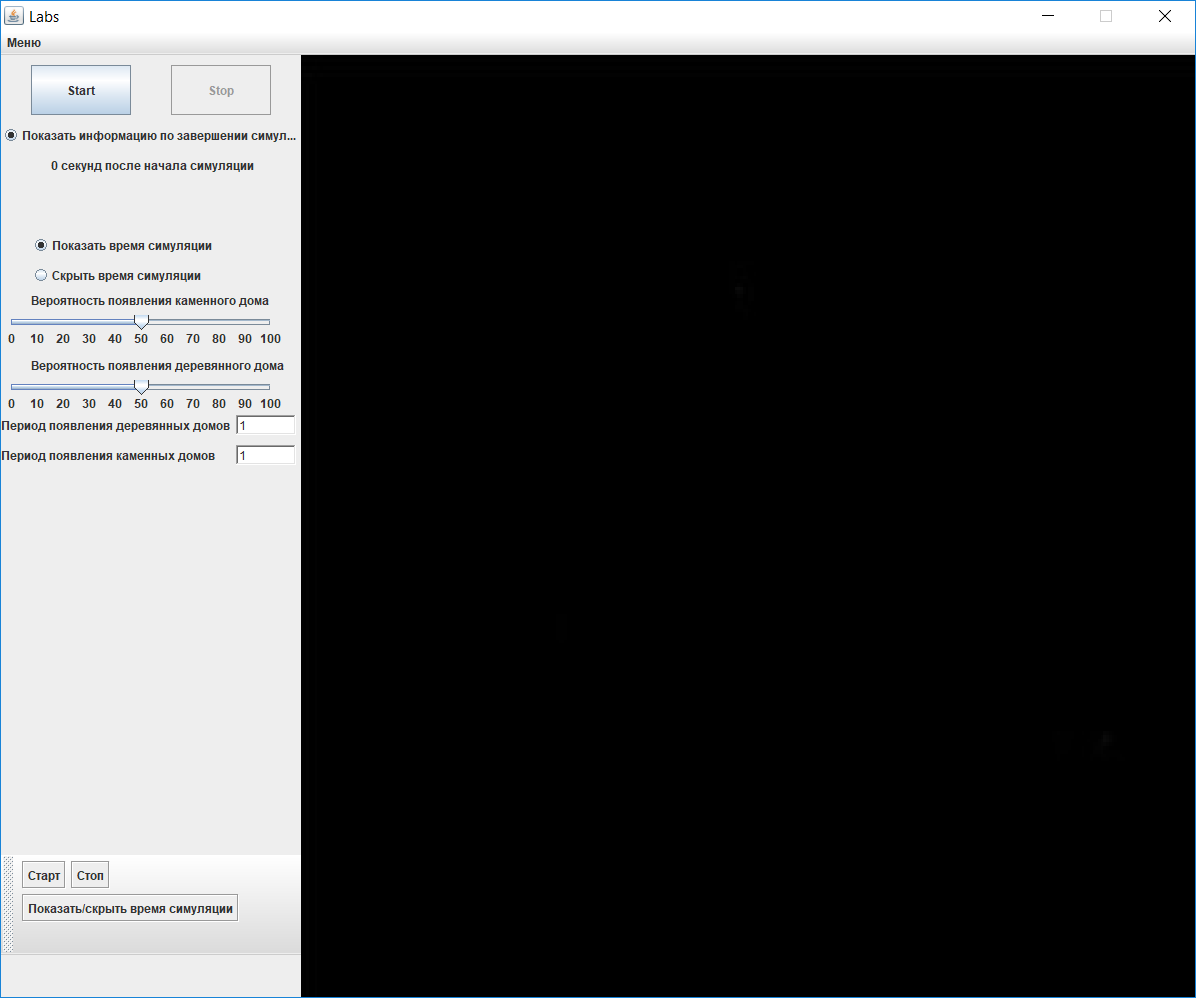
**Темы**: ***Разработка графического интерфейса приложения.***

#### Практические задания

1. Познакомиться с основными графическими библиотеками Java - AWT и Swing и их основными компонентами. Изучить классы менеджеров компоновки, классы создания меню приложения, основных и диалоговых окон и т.д.
2. Для хранения генерируемых объектов использовать динамический массив объектов.
3. Доработать программу, созданную в лабораторной работе № 1:
   1. поделить рабочую область окна приложения на 2 части. Визуализация переносится в одну часть окна, панель управления в другую;
   2. добавить кнопки «Старт» и «Стоп» в панель управления. Они должны запускать и останавливать симуляцию соответственно. Если симуляция остановлена, то кнопка «Стоп» должна блокироваться. Если симуляция идет, то блокируется кнопка «Старт». Клавиши **B** и **E** должны функционировать по-прежнему;
   3. добавить переключатель «Показывать информацию», который разрешает отображение модального диалога из 7 пункта задания;
   4. добавить группу из 2 исключающих переключателей: «Показывать время симуляции» и «Скрывать время симуляции». Клавиша **T** должна функционировать по-прежнему;
   5. используя различные менеджеры компоновки реализовать интерфейс пользователя согласно индивидуальному заданию;
   6. добавить в программу главное меню и панель инструментов, в которых продублировать основные команды интерфейса пользователя;
   7. при остановке симуляции должно появляться модальное диалоговое окно (при условии, что оно разрешено) с информацией о количестве и типе сгенерированных объектов, а также времени симуляции. Вся информация выводится в элементе TextArea, недоступном для редактирования. В диалоговом окне должно быть 2 кнопки: «ОК» и «Отмена». При нажатии на «ОК» симуляции полностью останавливается, а при нажатии на «Отмена», соответственно продолжается;
   8. предусмотреть проверку данных вводимых пользователем. При вводе неверного значения обрабатывать исключительную ситуацию: выставлять значение по умолчанию и выводить диалоговое окно с сообщением об ошибке;
   9. Реализовать следующие элементы управления:
   * Периоды рождения объектов – текстовые поля;
   * Для задания вероятностей рождения объектов - JComboBox и JSlider  (шаг значений 10);
   * Дополнить интерфейс поясняющими метками.

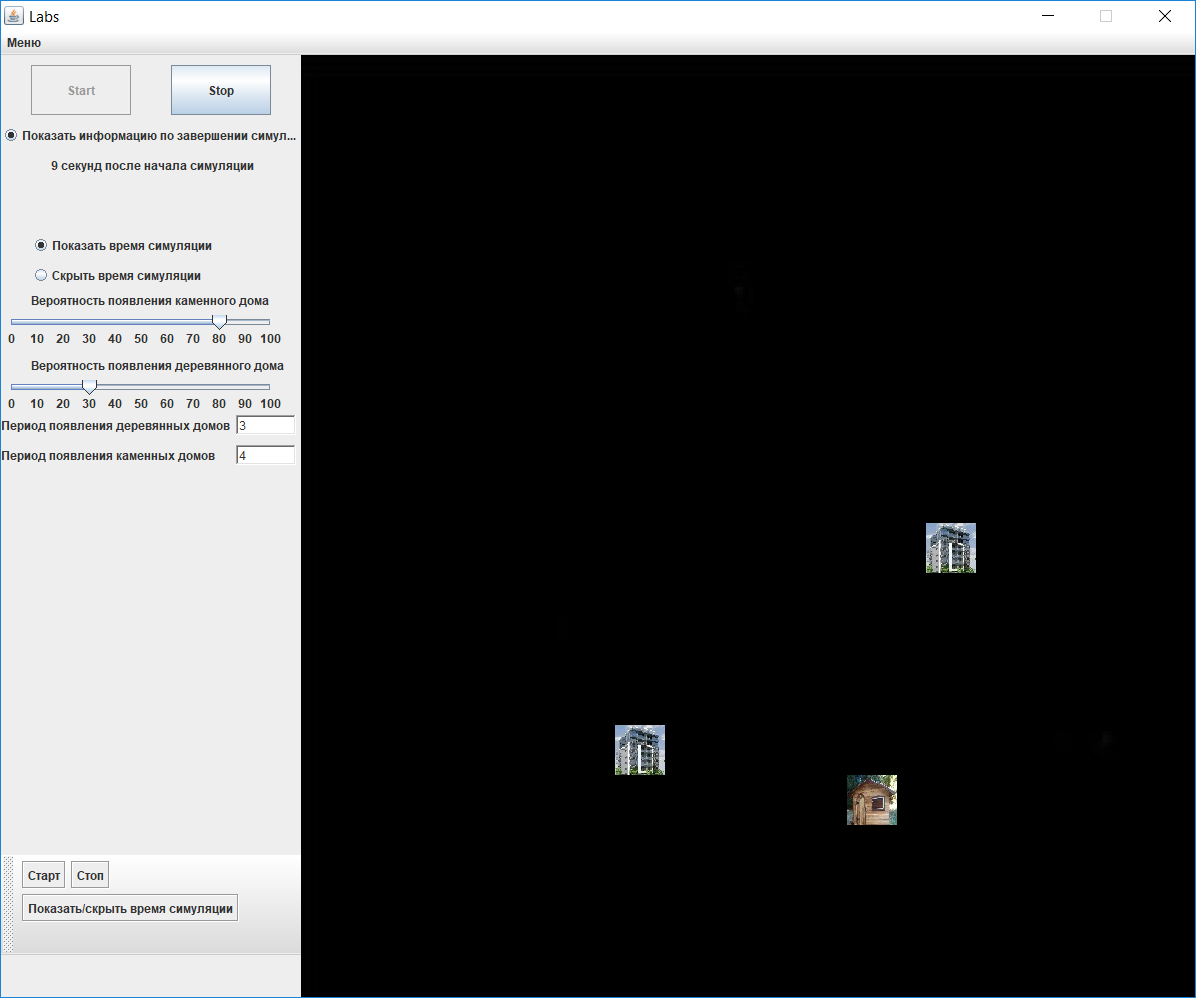
**Результаты работы программы**

В программу был добавлен GUI , с кнопками указанными в задании, которые отвечают за функции, которые объявлены заданием.



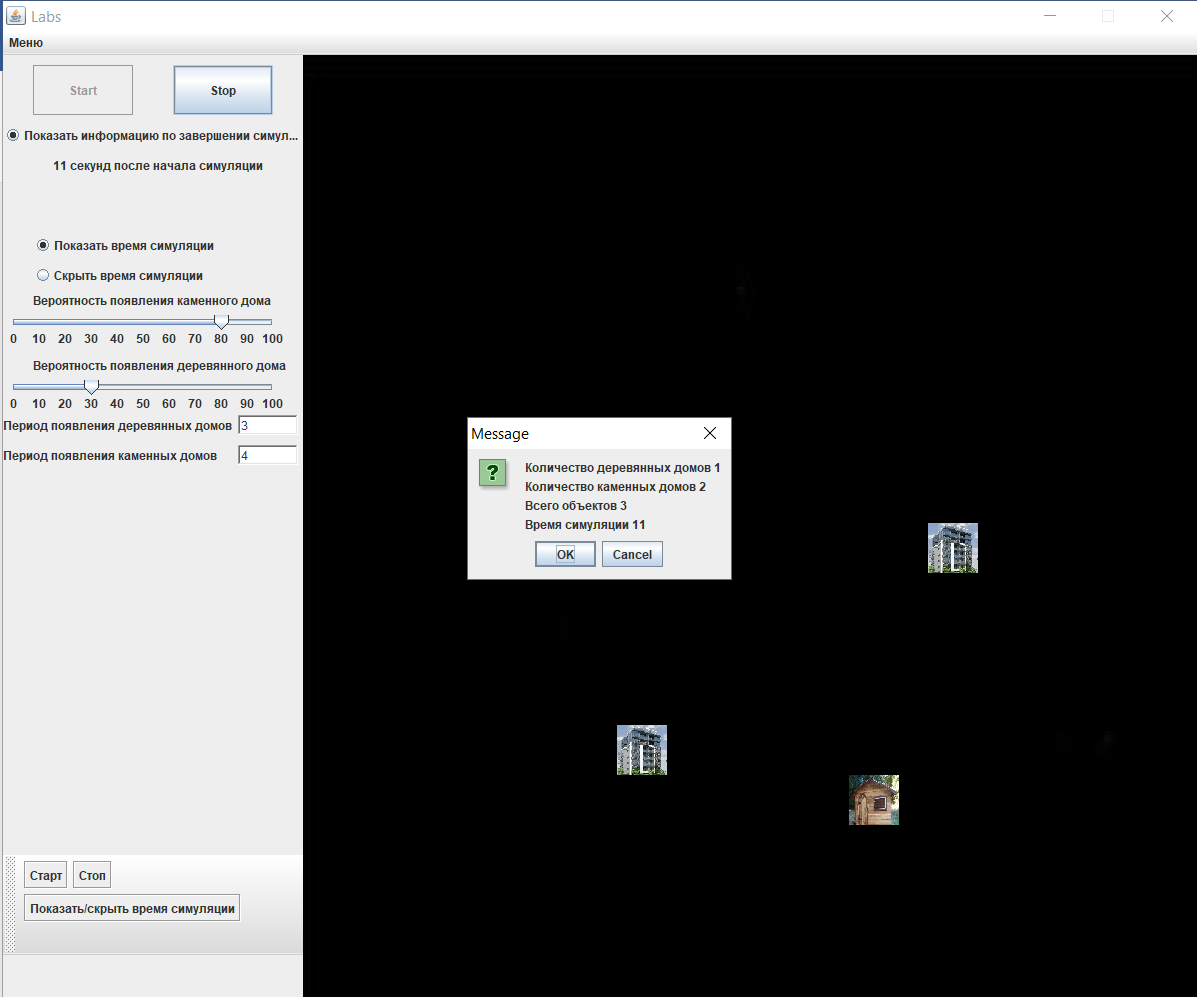
Как видно на скриншоте, есть поля JTextField и JSlider для изменения параметров симуляции. А также JRadioButton для переключения вывода определенной информации.

При нажатии на кнопку Start происходи следующее

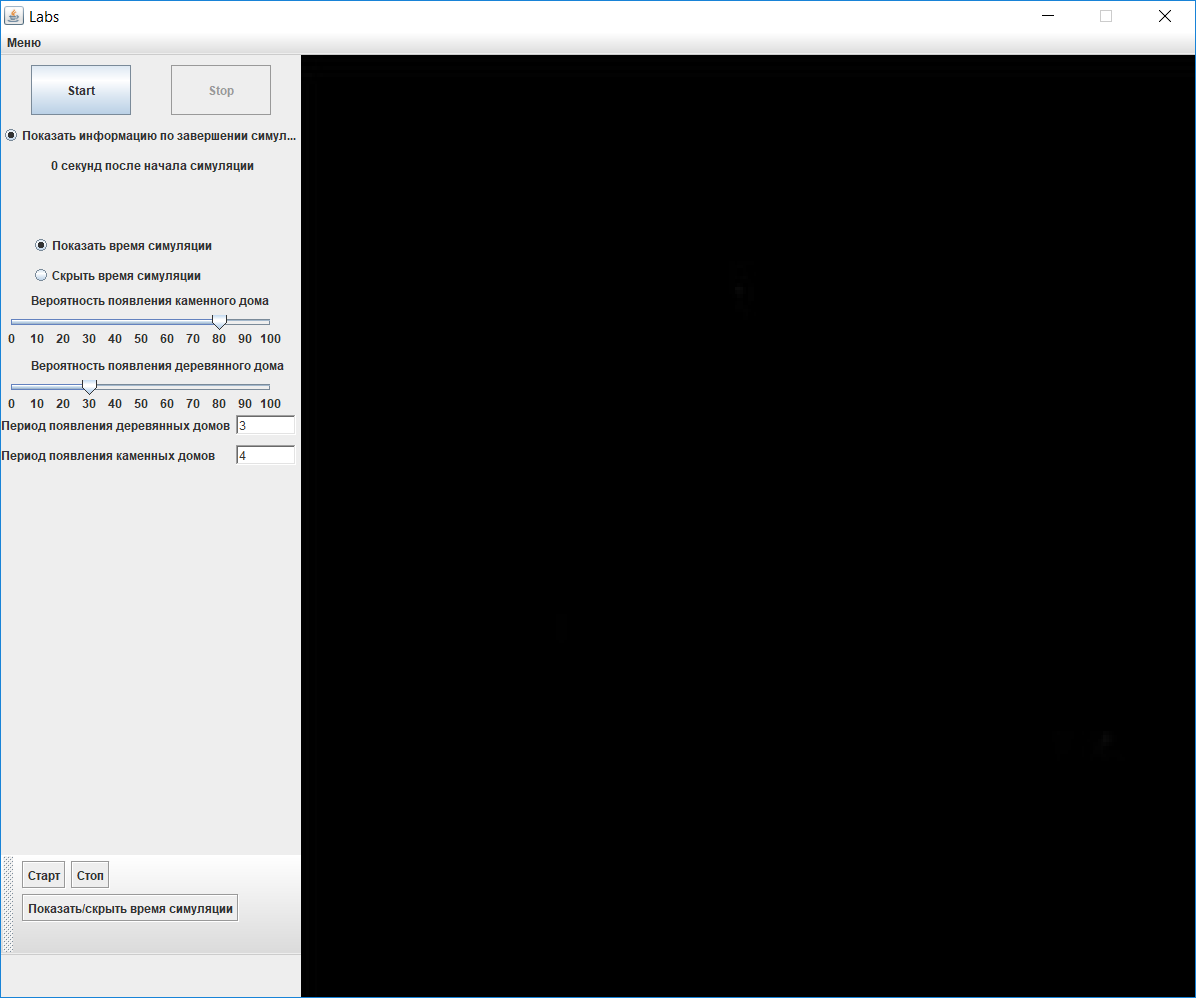


Как видно изменив параметры симуляции, мы получили изменения в поле визуализации.

В данный момент при нажатии на кнопку Stop, будет создано модальное диалоговое окно, которое покажет информацию о симуляции.



При нажатии на кнопку Cancel симуляция продолжится, а при нажатии на кнопку Ok очистится поле визуализации, нажмём на Ok и посмотрим



**Листинг программы:**

House.java

package Tutorial;  
  
import javax.swing.\*;  
import java.util.Random;  
  
public abstract class House extends JPanel implements IBehaviour  
{  
  
 int x,y;  
 JLabel jl;  
 int k=0;  
 @Override  
 public void movexy(int x, int y) {}  
 @Override  
 public void setx(int x) {this.x = x;}  
 @Override  
 public void sety(int y) {this.y = y;}  
 @Override  
 public int getx() {return this.x;}  
 @Override  
 public int gety() {return this.y;}  
 }

Habitat.java

package Tutorial;  
  
import java.util.\*;  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Habitat {  
  
 static int *N1*=1,*N2*=1,*P1*=50,*P2*=50;  
 static int *K1*=10,*K2*=10;  
 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)  
 {  
 Random random=new Random();  
 if(t%*N1*==0)  
 {  
 int i=random.nextInt(101);  
 if(i<=*P1*)  
 {  
 int x=random.nextInt(501)+300;  
 int y=random.nextInt(901);  
 Wood wood= new Wood(x,y);  
 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(501)+300;  
 int y=random.nextInt(901);  
 Stone stone=new Stone(x,y);  
 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;  
 nextwood.dead(t);  
 if(nextwood.isDead())  
 {  
 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;  
 nextstone.dead(t);  
 if(nextstone.isDead())  
 {  
 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()+1];  
 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 clear()  
 {  
 *ids*.clear();  
 *borns*.clear();  
 *houses*.clear();  
 Frame.*objs*.removeAll();  
 new Habitat();  
 }  
}

Wood.java

package Tutorial;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Wood extends House {  
 static int *woods*=0;  
 ImageIcon img=new ImageIcon(new javax.swing.ImageIcon("src/wood.jpg").getImage().  
 getScaledInstance(50,50, Image.*SCALE\_DEFAULT*));  
 Wood(int x,int y)  
 {  
 jl=new JLabel();  
 jl.setBounds(x,y,img.getIconWidth(),img.getIconHeight());  
 jl.setIcon(img);  
 Frame.*objs*.add(jl,0,0);  
 Frame.*objs*.repaint();  
 setId();  
 }  
}

Stone.java

package Tutorial;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Stone extends House {  
 static int *stones*=0;  
 ImageIcon img =new ImageIcon(new javax.swing.ImageIcon("src/stone.jpg").getImage()  
 .getScaledInstance(50,50, Image.*SCALE\_DEFAULT*));  
 Stone(int x, int y)  
 {  
 jl=new JLabel();  
 jl.setBounds(x,y,img.getIconWidth(),img.getIconHeight());  
 jl.setIcon(img);  
 Frame.*objs*.add(jl,0,0);  
 Frame.*objs*.repaint();  
 setId();  
 }  
}

IBehaviour.java

package Tutorial;  
  
public interface IBehaviour {  
  
 void movexy(int x,int y);  
 void setx(int x);  
 void sety(int y);  
 int getx();  
 int gety();  
}

Frame.java

package Tutorial;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
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,*prev*=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");  
 static TextField *field1*=new TextField("1"),*field2*=new TextField("1"),*field3*=new TextField("10"),*field4*=new TextField("10");  
 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");  
 static JSlider *slider1*= new JSlider(0,100,50),*slider2*=new JSlider(0,100,50);  
  
 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");  
 static JToolBar *jToolBar* = new JToolBar();  
 static JButton *jtbStart* = new JButton("Старт"), *jtbStop* = new JButton("Стоп"), *jtbTime* = new JButton("Показать/скрыть время симуляции");  
  
 Frame()  
 {  
 *frame*.setLayout(null);  
 *frame*.setSize(*WIDTH*,*HEIGHT*);  
 *frame*.setDefaultCloseOperation(*frame*.*EXIT\_ON\_CLOSE*);  
 *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*);  
 *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(50,500,200,100);  
 *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 buttonGroup = new ButtonGroup();  
 buttonGroup.add(*timeOn*);  
 buttonGroup.add(*timeOff*);  
 *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*);  
  
  
 *jmCommands*.add(*jmiStart*);  
 *jmCommands*.add(*jmiStop*);  
 *jmCommands*.add(*jmiTime*);  
 *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);  
  
 Listeners listen=new Listeners();  
  
 *habitat*=new Habitat();  
 }  
 static void run() {  
 while(true)  
 {  
 *time\_simulation*();  
 /\*try {  
 Thread.sleep(500/60);  
 } catch (InterruptedException e) {  
 }\*/  
 if(*simulation*==true)  
 {  
 *time*=System.*currentTimeMillis*()-*start*;  
 long step=*time*-*prev*;  
 if(step>=1000)  
 {  
 *t* = (int) (*time* / 1000);  
 *habitat*.update(*t*);  
 *prev* = *time*;  
 }  
 }  
 }  
 }  
  
 static void time\_simulation( ){  
 if(*text*){  
 *textTimeSimulation*.setText((*time*/1000)+" секунд после начала симуляции");  
 }else {  
 *textTimeSimulation*.setText("");  
 }  
 *panel*.repaint();  
 }  
}

Listeners.java

package Tutorial;  
  
import javax.swing.\*;  
import java.awt.event.\*;  
import java.util.Iterator;  
  
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 keyTyped(KeyEvent e) {  
  
 if(String.*valueOf*(e.getKeyChar()).equals("t")||String.*valueOf*(e.getKeyChar()).equals("T"))  
 {  
 *func\_T*();  
 if(Frame.*text*)  
 Frame.*timeOn*.setSelected(true);  
 else  
 Frame.*timeOff*.setSelected(true);  
 }  
 if(String.*valueOf*(e.getKeyChar()).equals("e")||String.*valueOf*(e.getKeyChar()).equals("E"))  
 {  
 if(Frame.*simulation*)  
 *func\_E*();  
 }  
 if(String.*valueOf*(e.getKeyChar()).equals("b")||String.*valueOf*(e.getKeyChar()).equals("и")||String.*valueOf*(e.getKeyChar()).equals("B")||String.*valueOf*(e.getKeyChar()).equals("И"))  
 {  
 if(!Frame.*simulation*)  
 *func\_B*();  
 }  
 }  
 });  
  
 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.*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.*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);  
 });  
 }  
 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;  
 Frame.*prev*=0;  
 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));  
 }  
}

Main.java

package Tutorial;  
  
public class Main {  
  
  
 public static void main(String[] args) {  
 Frame frm=new Frame();  
 frm.*run*();  
 }  
}