

Лабораторная работа №2

Технология программирования

*«Разработка графического интерфейса приложения»*

Студент: Воликов М.В.

Группа: АВТ-910

Вариант: 4

Преподаватель: Дубков И.С.



НОВОСИБИРСК 2021

Оглавление

[Задание 3](#_Toc38167352)

[Ход работы 4](#_Toc38167353)

[Демонстрация работы программы 4](#_Toc38167354)

[Структурный паттерн Singleton 4](#_Toc38167355)

[Разделение рабочей области окна 4](#_Toc38167356)

[Создание объектов интерфейса 5](#_Toc38167357)

[Размещение объектов интерфейса 5](#_Toc38167358)

[Кнопки старт и стоп 5](#_Toc38167359)

[Другие объекты интерфейса 7](#_Toc38167360)

[Главное меню 8](#_Toc38167361)

[Другие классы 9](#_Toc38167362)

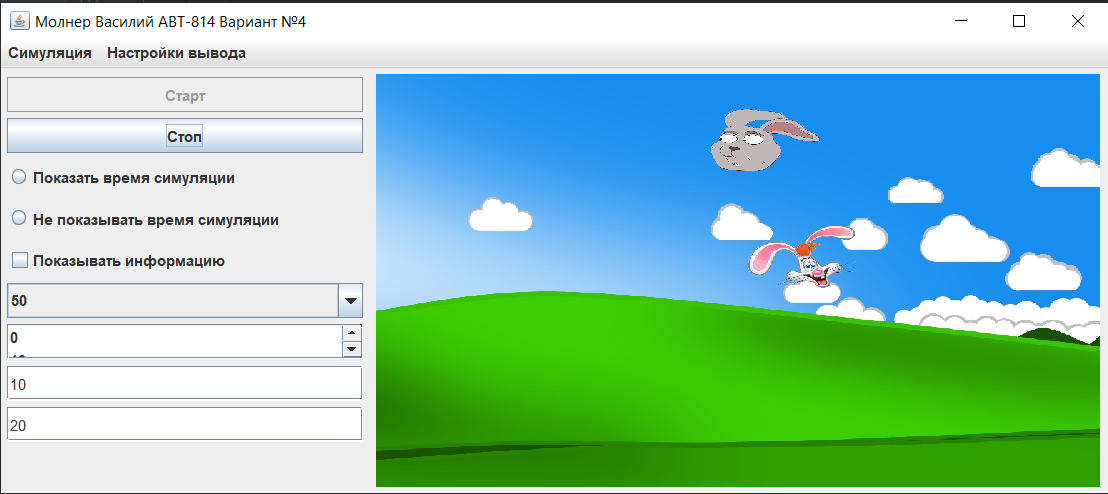
[Вывод 9](#_Toc38167363)

# Задание

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. Для обеспечения доступа всем элементам приложения и создания только одного экземпляра массива объектов используйте структурный паттерн Singleton.
3. При реализации шаблона Singleton используется класс Singleton. В этом классе определяется закрытый конструктор, имеется закрытая статическая переменная, в которой хранится ссылка на единственный экземпляр данного класса, а также определен статический метод доступа, возвращающий ссылку на этот экземпляр. Так же класс содержит объект коллекции и открытые методы доступа к ней. Таким образом можно в любом месте приложения получить ссылку на объект класса Singleton и доступ к коллекции объектов.
4. Доработать программу, созданную в лабораторной работе № 1:
5. поделить рабочую область окна приложения на 2 части. Визуализация переносится в одну часть окна, панель управления в другую;
6. 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;
7. добавить переключатель «Показывать информацию», который разрешает отображение модального диалога из 7 пункта задания;
8. добавить группу из 2 исключающих переключателей: «Показывать время симуляции» и «Скрывать время симуляции». Клавиша **T** должна функционировать по-прежнему;
9. используя различные менеджеры компоновки, реализовать интерфейс пользователя согласно индивидуальному заданию;
10. добавить в программу главное в меню и панель инструментов, в которых продублировать основные команды вашего интерфейса пользователя;
11. 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 останавливается, а при нажатии на «Отмена», соответственно продолжается;
12. предусмотреть проверку данных вводимых пользователем. При вводе неверного значения обрабатывать исключительную ситуацию: выставлять значение по умолчанию и выводить диалоговое окно с сообщением об ошибке;
13. Реализовать следующие элементы управления:
14. Периоды рождения объектов – текстовые поля;
15. Для задания вероятностей рождения объектов комбобокс и список (шаг значений 10%);

# Ход работы

## Демонстрация работы программы

Рисунок - Главное окно

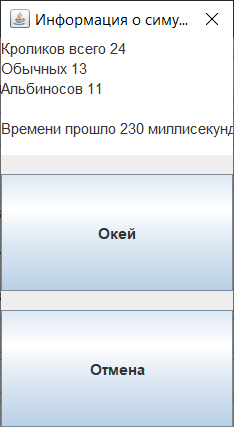


Рисунок - Модальное диалоговое окно

## Структурный паттерн Singleton

Класс Singleton предоставляет доступ к массиву объектов в любой точке программы.

public class Singleton {  
 private Singleton(){}  
 private static ArrayList<Rabbit> *link* = new ArrayList<>();  
 public static ArrayList<Rabbit> getLink() {  
 return *link*;  
 }  
}

## Разделение рабочей области окна

Для разделения рабочей области был использован менеджер компоновки GridBagLayouts.

container.setComponentOrientation(ComponentOrientation.*LEFT\_TO\_RIGHT*);  
container.setLayout(new GridBagLayout());  
  
GridBagConstraints constraints = new GridBagConstraints();  
constraints.insets = new Insets(5,5,5,5);  
constraints.gridy = 0;  
constraints.gridx = 0;  
constraints.fill = GridBagConstraints.*BOTH*;  
constraints.weightx = 0.1;  
constraints.weighty = 1;  
container.add(jPanelControl, constraints);  
  
constraints.gridx = 1;  
constraints.weightx = 0.9;  
container.add(jPanelImage, constraints);

## Создание объектов интерфейса

Все объекты интерфейса были созданы в глобальной области класса GUI.

JPanel jPanelImage = new JPanel(), jPanelControl = new JPanel();  
JButton jButtonStart = new JButton("Старт"), jButtonStop = new JButton("Стоп");  
JLabel jLabel = new JLabel();  
JDialog jDialog = new JDialog(this,"Информация о симуляции", true);  
JTextArea jTextArea = new JTextArea();  
JButton jButtonOk = new JButton("Окей"), jButtonCancel = new JButton("Отмена");  
JComboBox jComboBox = new JComboBox();  
JList jList = new JList();  
JTextField jTextFieldNOrdinary = new JTextField();  
JTextField jTextFieldNAlbino = new JTextField();

## Размещение объектов интерфейса

Объекты добавлялись в панель контроля (левая панель) при помощи менеджера компоновки GridLayout.

jPanelControl.setLayout(new GridLayout(10,1,5,5));  
jPanelControl.add(jButtonStart);  
jPanelControl.add(jButtonStop);  
jPanelControl.add(jRadioButtonShowTime);  
jPanelControl.add(jRadioButtonNoShowTime);  
jPanelControl.add(jCheckBoxShowInf);  
jPanelControl.add(jComboBox);  
jPanelControl.add(jPanelList);  
jPanelControl.add(jTextFieldNOrdinary);  
jPanelControl.add(jTextFieldNAlbino);  
jPanelControl.add(jLabel);

## Кнопки старт и стоп

Для каждой из кнопок был создан анонимный класс слушателя действия, который вызывал соответствующий ему метод.

jButtonStart.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 ActionStart();  
 }  
});  
  
jButtonStop.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 ActionStop();  
 }  
});

Методы ActionStart() и ActionStop():

private void ActionStart() {  
 Habitat.*POrdinary* = (double) Integer.*parseInt*((String) jComboBox.getSelectedItem()) / 100;  
 Habitat.*KAlbino* = (double) Integer.*parseInt*((String) jList.getSelectedValue()) / 100;  
 try {  
 Habitat.*NOrdinary* = Integer.*parseInt*(jTextFieldNOrdinary.getText());  
 Habitat.*NAlbino* = Integer.*parseInt*(jTextFieldNAlbino.getText());  
 }  
 catch (NumberFormatException ex) {  
 JDialog jDialogEx = new JDialog(this,"Ошибка!",true);  
 jDialogEx.setSize(320, 240);  
 jDialogEx.setLayout(new GridLayout(2,1));  
  
 JButton jButtonExOk = new JButton("Я исправлю");  
 JLabel jLabelExInf = new JLabel("Неверный ввод данных");  
 jDialogEx.add(jLabelExInf);  
 jDialogEx.add(jButtonExOk);  
  
 jTextFieldNOrdinary.setText("10");  
 jTextFieldNAlbino.setText("20");  
  
 jButtonExOk.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 jDialogEx.setVisible(false);  
 }  
 });  
  
 jDialogEx.setVisible(true);  
 return;  
 }  
 if (!flag\_is\_work) {  
 flag\_is\_work = true;  
 timer = new Timer();  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 Habitat.*Update*(cumulativePeriod);  
 if (RadioButtonBoolean) {  
 ActionShowText();  
 }  
 cumulativePeriod += period;  
 }  
 }, 0, period);  
 }  
 jButtonStop.setEnabled(true);  
 jButtonStart.setEnabled(false);  
}  
  
private void ActionStop() {  
 timer.cancel();  
 flag\_is\_work = false;  
 Singleton.*getLink*().clear();  
 jButtonStop.setEnabled(false);  
 jButtonStart.setEnabled(true);  
 jTextArea.setText("Кроликов всего " + Rabbit.*getAllQuantity*() +  
 "\nОбычных " + Ordinary.*getOrdinaryQuantity*() +  
 "\nАльбиносов " + Albino.*getAlbinoQuantity*() +  
 "\n\nВремени прошло " + cumulativePeriod + " миллисекунд");  
 cumulativePeriod = 0;  
 Rabbit.*AllQuantity* = 0;  
 Ordinary.*OrdinaryQuantity* = 0;  
 Albino.*AlbinoQuantity* = 0;  
 jDialog.setVisible(true);  
}

## Другие объекты интерфейса

Для остальных объектов также были созданы слушатели.

jCheckBoxShowInf.addItemListener(new ItemListener() {  
 @Override  
 public void itemStateChanged(ItemEvent e) {  
 timer.cancel();  
 flag\_is\_work = false;  
 jTextArea.setText("Кроликов всего " + Rabbit.*getAllQuantity*() +  
 "\nОбычных " + Ordinary.*getOrdinaryQuantity*() +  
 "\nАльбиносов " + Albino.*getAlbinoQuantity*() +  
 "\n\nВремени прошло " + cumulativePeriod + " миллисекунд");  
 jDialog.setVisible(true);  
 }  
});  
  
jRadioButtonShowTime.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 RadioButtonBoolean = true;  
 }  
});  
  
jRadioButtonNoShowTime.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 RadioButtonBoolean = false;  
 jLabel.setVisible(false);  
 }  
});  
  
jComboBox.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Habitat.*POrdinary* = (double)Integer.*parseInt*((String)jComboBox.getSelectedItem()) / 100;  
 }  
});  
  
jList.addListSelectionListener(new ListSelectionListener() {  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 Habitat.*KAlbino* = (double) Integer.*parseInt*((String)jList.getSelectedValue()) / 100;  
 }  
});  
  
jButtonCancel.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 jDialog.setVisible(false);  
 ActionStart();  
 }  
});  
  
jButtonOk.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 jDialog.setVisible(false);  
 cumulativePeriod = 0;  
 Singleton.*getLink*().clear();  
 jButtonStop.setEnabled(false);  
 jButtonStart.setEnabled(true);  
 }

## Главное меню

Для создания меню был выделен отдельный метод, который вызывается из конструктора класса GUI. Для наглядности иерархии пунктов меню в коде используется табуляция.

private void creatMenuBar() {  
 JMenuBar menuBar = new JMenuBar();  
  
 JMenu SimulationMenu = new JMenu("Симуляция");  
  
 JMenuItem StartItem = new JMenuItem("Старт");  
 SimulationMenu.add(StartItem);  
 StartItem.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 ActionStart();  
 }  
 });  
  
 JMenuItem StopItem = new JMenuItem("Стоп");  
 SimulationMenu.add(StopItem);  
 StopItem.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 ActionStop();  
 }  
 });  
  
 JMenuItem exitItem = new JMenuItem("Выход");  
 SimulationMenu.add(exitItem);  
 exitItem.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 System.*exit*(0);  
 }  
 });  
  
 JMenu SettingsMenu = new JMenu("Настройки вывода");  
  
 JMenu ShowTimeMenu = new JMenu("Показывать время");  
 SettingsMenu.add(ShowTimeMenu);  
  
 JMenuItem YesItem = new JMenuItem("Да");  
 ShowTimeMenu.add(YesItem);  
 YesItem.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 RadioButtonBoolean = true;  
 }  
 });  
  
 JMenuItem NoItem = new JMenuItem("Нет");  
 ShowTimeMenu.add(NoItem);  
 NoItem.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 RadioButtonBoolean = false;  
 jLabel.setVisible(false);  
 }  
 });  
  
 JMenuItem ShowInfItem = new JMenuItem("Информация о симуляции");  
 SettingsMenu.add(ShowInfItem);  
 ShowInfItem.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 timer.cancel();  
 flag\_is\_work = false;  
 jTextArea.setText("Кроликов всего " + Rabbit.*getAllQuantity*() +  
 "\nОбычных " + Ordinary.*getOrdinaryQuantity*() +  
 "\nАльбиносов " + Albino.*getAlbinoQuantity*() +  
 "\n\nВремени прошло " + cumulativePeriod + " миллисекунд");  
 jDialog.setVisible(true);  
 }  
 });  
  
 menuBar.add(SimulationMenu);  
 menuBar.add(SettingsMenu);  
  
 setJMenuBar(menuBar);  
}

## Другие классы

Остальные классы не были изменены в ходе выполнения данной лабораторной работы.

# Вывод

В ходе выполнения работы был создан графический интерфейс для программы, разработанной в первой лабораторной работе. Также был изучен структурный паттерн Singleton и применён на практике.