МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

НГТУ

Лабораторная работа №2

по предмету «Технология программирования»

на тему «Разработка графического интерфейса приложения»

Факультет: АВТ Преподаватель:

Группа: АВТ-809 Михайленко Д.А.

Студент: Швыдко Е.А.

Новосибирск 2020

Цели работы:

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:
   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 в меню и панель инструментов, в которых продублировать основные команды вашего 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 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 комбобокс и список (шаг значений 10%);
   * Дополнить интерфейс поясняющими метками.

Ход разработки:

* Статический массив объектов был заменен динамическим массивом Vector
* Был реализован паттерн Singleton

**package** com.company;

**import** java.util.Vector;

**public** **class** Singleton {

**private** **static** Singleton singleton;

**private** **static** **Vector**<Animal> vector = **new** **Vector**<>();

**public** **static** **synchronized** Singleton getVector(){

**if**(singleton == **null**) singleton = **new** Singleton();

**return** singleton;

}

**public** **Vector**<Animal> getArray(){

**return** vector;

}

}

* Были добавлены следующие элементы управления: кнопки «Start», «Stop» и «Showtime»; добавлен не исключающий переключатель «Show info»; добавлены два исключающих переключателя «Show time of simulation» и «Hide time of simulation»; добавлен MenuBar «Process»; разработан и добавлено модальное диалоговое окно, показывающее статистическую информацию о симуляции на момент появления диалога; добавлены текстовые поля, позволяющие редактировать периоды рождения объектов вовремя симуляции; добавлен ComboBox и JList, позволяющие редактировать шансы рождения объектов каждого типа

showObjects = new JCheckBox("Показать диалог", false);

add(showObjects);

showObjects.setBounds(5, 45,140,13);

ButtonGroup group = new ButtonGroup();

showTimeSim = new JRadioButton("Показать время", true);

group.add(showTimeSim);

add(showTimeSim);

showTimeSim.setBounds(5,60,120,13);

hideTimeSim = new JRadioButton("Скрыть время", false);

group.add(hideTimeSim);

add(hideTimeSim);

hideTimeSim.setBounds(5,75,120,13);

showTimeSim.addItemListener(new ItemListener() {

**@Override**

**public** void itemStateChanged(ItemEvent itemEvent) {

timerLabel.setVisible(**true**);

}

});

hideTimeSim.addItemListener(new ItemListener() {

**@Override**

**public** void itemStateChanged(ItemEvent itemEvent) {

timerLabel.setVisible(**false**);

}

});

JMenuBar Bar = new JMenuBar();

JMenu mainMenu = new JMenu("�"лавная");

JMenuItem menuStart = new JMenuItem("Старт");

JMenuItem menuStop = new JMenuItem("Стоп");

JMenuItem menuShowTime = new JMenuItem("Показать время");

mainMenu.add(menuStart);

mainMenu.add(menuStop);

mainMenu.add(menuShowTime);

Bar.add(mainMenu);

setJMenuBar(Bar);

menuStart.addActionListener(new ActionListener() {

**@Override**

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

startButton.doClick();

}

});

menuStop.addActionListener(new ActionListener() {

**@Override**

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

stopButton.doClick();

}

});

menuShowTime.addActionListener(new ActionListener() {

**@Override**

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

showTimeButton.doClick();

}

});

catDelayLabel = new JLabel("Пер. рождения кошек");

add(catDelayLabel);

catDelayLabel.setBounds(250,10,130,13);

dogDelayLabel = new JLabel("Пер. рождения собак");

add(dogDelayLabel);

dogDelayLabel.setBounds(385,10,130,13);

catDelayField = new JTextField(10);

add(catDelayField);

catDelayField.setBounds(250,27,130,16);

catDelayField.addActionListener(new ActionListener() {

**@Override**

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

double kk = **Double**.parseDouble(catDelayField.getText());

**if**(kk >= 1.0) Habitat.catDelay = kk;

**else** {

Habitat.catDelay = 1.0;

catDelayField.setText("1.0");

}

}

});

dogDelayField = new JTextField(10);

add(dogDelayField);

dogDelayField.setBounds(385,27,130,16);

dogDelayField.addActionListener(new ActionListener() {

**@Override**

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

double kk = **Double**.parseDouble(dogDelayField.getText());

**if**(kk >= 1.0) Habitat.dogDelay = kk;

**else** {

Habitat.dogDelay = 1.0;

dogDelayField.setText("1.0");

}

}

});

catProbLabel = new JLabel("�'ер. рождения кошек");

add(catProbLabel);

catProbLabel.setBounds(520,10,130,13);

dogProbLabel = new JLabel("�'ер. рождения собак");

add(dogProbLabel);

dogProbLabel.setBounds(655,10,130,13);

String[] dogValues = {"0","10","20","30","40","50","60","70","80","90","100"};

dogListProb = new JList(dogValues);

dogListProb.setLayoutOrientation(JList.HORIZONTAL\_WRAP);

JScrollPane scrollList = new JScrollPane(dogListProb);

add(scrollList);

scrollList.setBounds(655,25,130,50);

dogListProb.addListSelectionListener(new ListSelectionListener() {

**@Override**

**public** void valueChanged(ListSelectionEvent e) {

**if**(dogListProb.getValueIsAdjusting()) **return**;

double kk = **Double**.parseDouble(dogListProb.getSelectedValue().toString());

Habitat.dogProb = kk/100;

Habitat.catProb = 1 - kk/100;

}

});

String[] catValues = {"0","10","20","30","40","50","60","70","80","90","100"};

catComboProb = new JComboBox(catValues);

catComboProb.setSelectedIndex(5);

add(catComboProb);

catComboProb.setBounds(520,25,130,15);

catComboProb.addActionListener(new ActionListener() {

**@Override**

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

double kk = **Double**.parseDouble((**String**)catComboProb.getSelectedItem());

Habitat.catProb = kk/100;

Habitat.dogProb = 1 - kk/100;

}

});

* Были добавлены поясняющие метки.

Выводы:

В результате выполнения работы был положен старт изучению графической библиотеки Swing, были изучены диалоговые окна и стандартные компоненты графического интерфейса.