НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Автоматики и Вычислительной Техники

Кафедра Вычислительной Техники

**ЛАБОРАТОРНАЯ РАБОТА № 2**

по дисциплине «Технология программирования»

*Основы программирования на Java. Обработка событий. Механизм делегирования событий.*

**Группа:** АВТ-710

**Студент:** Шелудько В.

Бурдинский И.

**Вариант:** 8

**Преподаватель:** Васюткина И.А.

# 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.
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 10);
   * Дополнить интерфейс поясняющими метками.

# Описание структуры классов:

Был дополнен класс HabitatView всеми необходимыми элементами пользовательского интерфейса.

# Описание функционала:

* + - 1. Вспомогательные методы, контролирующие состояние симуляции

private void continueSim(){ \_simulating = true; }

private void pauseSim() { \_simulating = false; }

private void startSim() {  
 for(Component tmp : \_panelUI.getComponents())  
 tmp.setEnabled(true);  
 btnStart.setEnabled(false);  
 startMenu.setEnabled(false);  
 \_firstRun = false;  
 \_simulating = true;  
 \_habitat = new Habitat();  
 \_timer = new Timer();  
 \_timer.schedule(new SimulationLoop(), 0, 10);  
 Employee.*Amount* = 0;  
 Manager.*Amount* = 0;  
 Worker.*Amount* = 0;  
}

private void stopSim() {  
 for(Component tmp : \_panelUI.getComponents())  
 tmp.setEnabled(false);  
 btnStart.setEnabled(true);  
 startMenu.setEnabled(true);  
 \_simulating = false;  
 \_graphics = this.getGraphics();  
 clearViewScreen();  
 if(!\_firstRun) {  
 int midWidth = *Width* / 2 - 100;  
 int midHeight = *Height* / 2;  
 \_graphics.setFont(new Font("Consolas", Font.*PLAIN*, 20));  
 \_graphics.setColor(new Color(0, 255, 255, 255));  
 \_graphics.drawString(String.*format*("Emp amount: %d", Employee.*Amount*), midWidth, midHeight);  
 \_graphics.setFont(new Font("Arial", Font.*ITALIC*, 18));  
 \_graphics.setColor(new Color(255, 255, 255, 255));  
 \_graphics.drawString(String.*format*("Wrk amount: %d", Worker.*Amount*), midWidth, midHeight + 50);  
 \_graphics.setFont(new Font("Consolas", Font.*BOLD*, 22));  
 \_graphics.setColor(new Color(255, 0, 255, 255));  
 \_graphics.drawString(String.*format*("Mng amount: %d", Manager.*Amount*), midWidth, midHeight + 100);  
 \_graphics.setFont(new Font("Arial", Font.*BOLD* | Font.*ITALIC*, 30));  
 \_graphics.setColor(new Color(200, 170, 30, 255));  
 \_graphics.drawString(String.*format*("%.3f", *ElapsedTime*), midWidth, midHeight + 150);  
 \_timer.cancel();  
 }  
 else {  
 \_graphics.drawString("Press Start", getWidth()/2, getHeight()/2);  
 }  
}

Методы для старта, запуска, паузы, продолжения анимации по событию с клавиатуры или с кнопок на панели управления или меню.

* + - 1. Демонстрация статистики текущей симуляции

public void showStats(){  
 pauseSim();  
 int result = JOptionPane.*showConfirmDialog*(this,  
 String.*format*("Emp amount: %d\nWrk amount: %d\nMng amount: %d\nelapsed time: %.3f", Employee.*Amount*, Worker.*Amount*, Manager.*Amount*, *ElapsedTime*),  
 "Statistic", JOptionPane.*OK\_CANCEL\_OPTION*);  
 if (result == JOptionPane.*OK\_OPTION*)  
 continueSim();  
 else if (result == JOptionPane.*CANCEL\_OPTION*)  
 stopSim();  
}

Демонстрация статистики симуляции через модальное окно.

* + - 1. Слушатель событий ввода в текстовые поля.
* private class ByTextFieldNameActionListener implements ActionListener {  
   public void actionPerformed(ActionEvent e) {  
   JTextField tmp = (JTextField) e.getSource();  
   float fieldValue;  
   try {  
   fieldValue = Float.*parseFloat*(tmp.getText());  
   } catch (NumberFormatException ee) {  
   fieldValue = 0;  
   }  
   if (fieldValue <= 0) fieldValue = 1;  
   switch (tmp.getName()) {  
   case ("SpawnTimeWorker"):  
   Worker.*TimeBetweenSpawn* = fieldValue;  
   break;  
   case ("SpawnTimeManager"):  
   Manager.*TimeBetweenSpawn* = fieldValue;  
   break;  
   case ("LifeTimeWorker"):  
   Worker.*LifeTime* = fieldValue;  
   break;  
   case ("LifeTimeManager"):  
   Manager.*LifeTime* = fieldValue;  
   break;  
   }  
   tmp.setText(String.*valueOf*(fieldValue));  
   }  
  }

Изменяемое значение определяется по имени элемента (нужно задать элементу имя). Если ввод был не корректен, то переменной присваивается 1.

# Выводы:

В ходе лабораторной работы научились работать со слушателями событий различных графических элементов и различными менеджерами компоновки. Научились обрабатывать исключительные ситуации методами языка Java.