Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Новосибирский государственный технический университет»

Кафедра вычислительной техники

Лабораторная работа № 1

По дисциплине «Технология программирования»

На тему: «Основы программирования на Java; обработка событий; механизм делегирования событий»

Факультет: АВТФ

Группа: АВТ-808

Студент: Хайрулина Елизавета

Вариант 12

Преподаватель: Михайленко Дмитрий Анатольевич

Новосибирск 2020

Цель

Познакомиться с особенностями технологии Java и изучить синтаксис языка Java. Изучить основные понятия и термины обработки событий в модели делегирования событий на Java. Разобрать приведенные примеры и разработать программу в соответствии с вариантом.

Задание

Разработать программу. Основная 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.

***Вариант 12***

Объекты – птицы. Бывают 2 видов: птенцы и взрослые птицы. Взрослые птицы генерируются каждые N1 секунд с вероятностью P1. Птенцы генерируются каждые N2 секунд при условии, что их количество менее K% от общего числа взрослых птиц, в противном случае – не генерируются.

Описание структуры программы

*interface IBehaviour*

Реализуется абстрактным классом и его наследниками.

*public abstract class Bird implements IBehaviour*

Абстрактный класс, реализующий интерфейс *IBehaviour.* Позволяет задавать изображение и расположение(координаты) некоторой птицы.

*public class Big extends Bird; public class Small extends Bird*

Реализация объектов через наследование: абстрактный класс + интерфейс → наследники. Определение объектов по варианту: определяется конкретное изображение взрослой птицы или птенца.

*public interface AbstractFactory*

Интерфейс для создания объектов классов. Содержит методы, позволяющие вернуть число сгенерированных птиц (Return\_the\_Number\_of\_animals()), создать очередную птицу (Bird Luntik(Integer time, Point place)) и обнулить имеющиеся счетчики (void Total\_destruction()).

*public class ConcreteFactory implements AbstractFactory*

Класс, реализующий интерфейс *AbstractFactory* и возвращающий нужный объект: взрослую птицу или птенца. Класс содержит переменные, хранящие исходные данные (time\_birth\_small; time\_birth\_big; chance\_birth\_big; percent;) позволяющие собирать статистику о сгенерированных объектах (private Integer number\_of\_birds; number\_of\_Big; number\_of\_Small).

*public class Habitat extends JFrame*

Класс «среда», определяющий размер рабочей области и хранящий массив объектов, с параметрами, заданными вариантом. Класс предусматривает метод void Update(), контролирующий таймер и время, прошедшее от начала симуляции. В данном методе генерируются новые объекты и помещаются в поле визуализации в случайном месте. Рабочее окно программы = область визуализации среды обитания объектов.

Класс содержит метод (public void Stop()), позволяющий остановить симуляцию и сохранить – вывести на экран необходимые данные: информация о количестве и типе сгенерированных объектов, время симуляции. А также метод отображающий текстом время в области визуализации, которое скрывается/показывается по клавише Т (public void Clock\_yes\_no(Boolean isShown)).

*public class ToBeContinued extends TimerTask*

Класс, содержащий переопределенный метод public void run(), который, в свою очередь, вызывает метод void Update(), позволяющий производить необходимую отрисовку среды.

*public class Depict\_a\_bird extends JPanel*

Класс, хранящий массив сгенерированных объектов, отвечающий за отрисовку этих самых объектов.

*public class AnimalTour extends KeyAdapter*

Класс, который запускает работу программы и перехватывает события, реализованные при помощи клавиатуры.

* Нажатие клавиши B – запускает симуляцию
* Нажатие клавиши E – останавливает симуляцию
* Нажатие клавиши T – отображение таймера

Результат работы программы

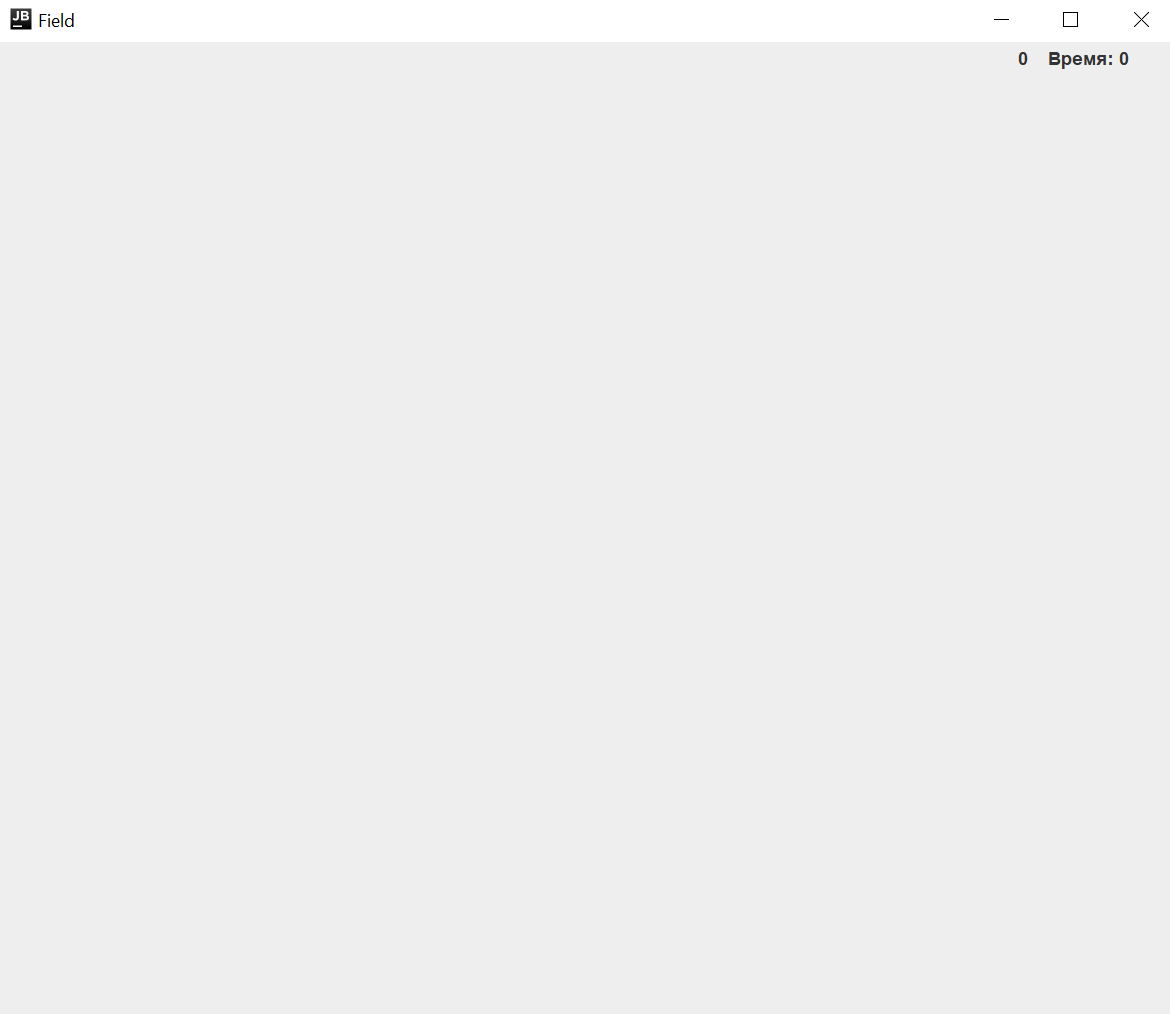
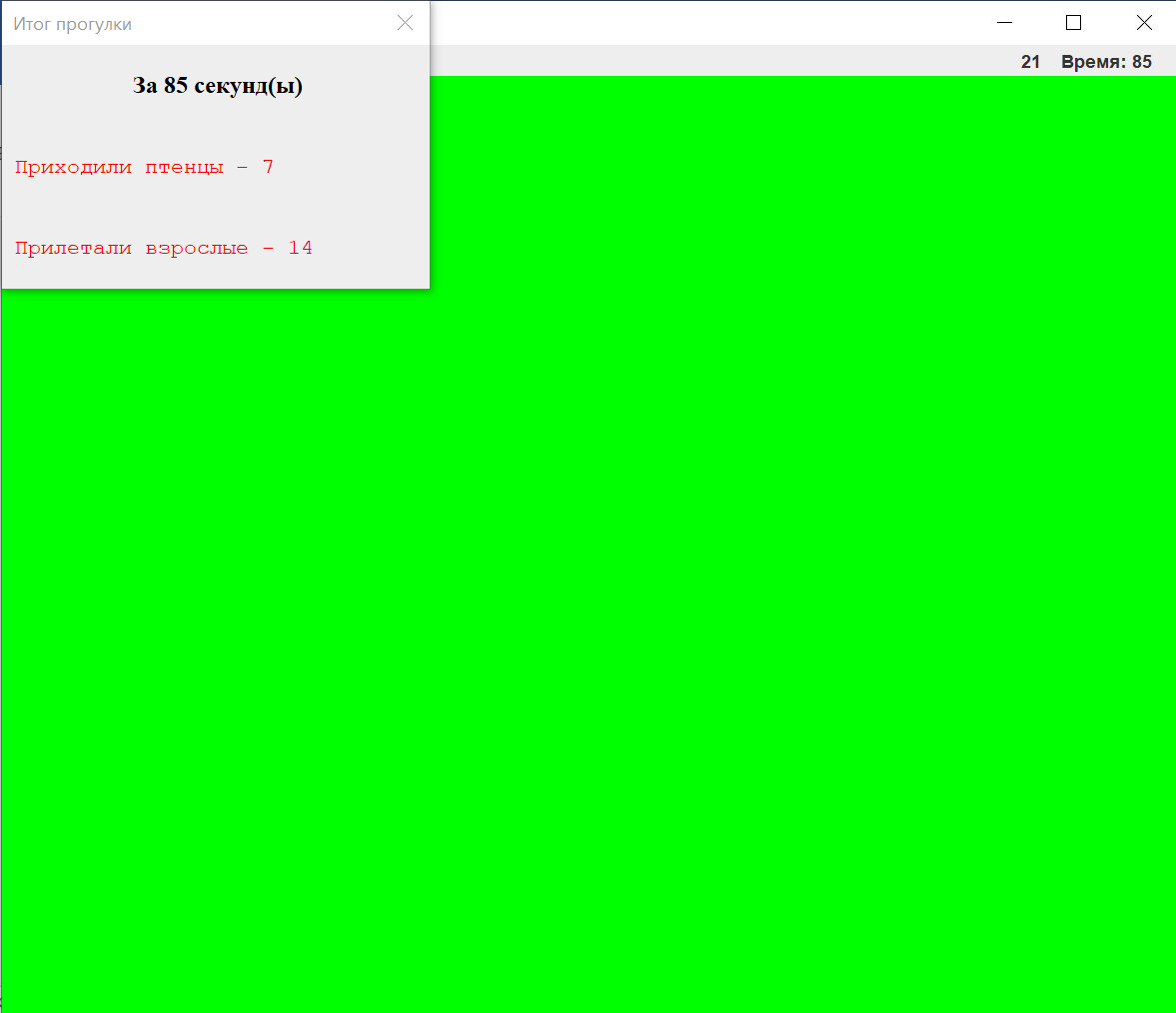
 Рис. 1 – Окно до запуска симуляции

 Рис. 2 - Симуляция

 Рис. 3 – Окончание симуляции

Выводы

В ходе выполнения данной лабораторной работы была достигнута её цель - знакомство с особенностями технологии Java и изучение синтаксиса языка Java. Для достижения данной цели были изучены основные понятия и термины обработки событий в модели делегирования событий на Java, разобраны приведенные примеры и разработана программа в соответствии с вариантом. Данная программа является разработкой упрощенной имитации поведения объектов (птиц: взрослых и птенцов). Программа реализует запуск процесса симуляции по клавише, генерацию объектов (птиц), завершение симуляции по другой клавише и вывод статистической информации.

Листинг программы

package Behav;  
  
import java.awt.\*;  
  
public interface IBehaviour {  
 Image getImage();  
 Point getPlace();  
}

package DrawPanel;  
  
import Object.\*;  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Depict\_a\_bird extends JPanel {  
  
 Bird[] bird\_s;  
 int Mass;  
  
 public Depict\_a\_bird(Bird bird\_s[], int Mass) {  
 this.bird\_s = bird\_s;  
 this.Mass = Mass;  
 }  
  
 @Override  
 protected void paintComponent(Graphics graphics)  
 {  
 super.paintComponent(graphics);  
  
 if (bird\_s[0] == null)  
 return;  
  
 for (int i=0; bird\_s [i] != null; i++)  
 if (i < Mass) {  
 setBackground(Color.*GREEN*);  
 graphics.drawImage(bird\_s[i].getImage(),bird\_s[i].getPlace().x,bird\_s[i].getPlace().y,50,50,null);  
 }  
 }  
}

package Fact;  
  
import Object.Bird;  
  
import java.awt.\*;  
import java.io.IOException;  
  
  
public interface AbstractFactory  
{  
 Integer Return\_the\_Number\_of\_animals();  
 Bird Luntik(Integer time, Point place) throws IOException; // Создание птицы (той или другой)  
 void Total\_destruction();  
}

package Fact;  
  
import Object.\*;  
  
import java.awt.\*;  
import java.io.IOException;  
import java.util.Random;  
  
  
public class ConcreteFactory implements AbstractFactory  
{  
  
 private Integer number\_of\_birds;  
 private Integer number\_of\_Big;  
 private Integer number\_of\_Small;  
  
 private final Integer time\_birth\_small;  
 private final Integer time\_birth\_big;  
 private final Float chance\_birth\_big;  
 private final Float percent;  
  
 public ConcreteFactory(Integer time\_birth\_small, Integer time\_birth\_big, Float chance\_birth\_big, Float percent)  
 {  
 number\_of\_birds = 0;  
 number\_of\_Big = 0;  
 number\_of\_Small = 0;  
  
 this.time\_birth\_small = time\_birth\_small;  
 this.time\_birth\_big = time\_birth\_big;  
 this.chance\_birth\_big = chance\_birth\_big;  
 this.percent = percent;  
 }  
  
 @Override  
 public Integer Return\_the\_Number\_of\_animals() {  
 return number\_of\_birds;  
 }  
  
  
 @Override  
 public void Total\_destruction() {  
 number\_of\_birds = 0;  
 number\_of\_Big = 0;  
 number\_of\_Small = 0;  
 }  
  
 @Override  
 public Bird Luntik(Integer time, Point place) throws IOException {  
  
 if (time % time\_birth\_big == 0)   
 {  
 Random random = new Random();  
 float chance = random.nextFloat();  
 if (chance <= chance\_birth\_big)  
 {  
 Bird fowl = new Big(place);  
 number\_of\_birds++;  
 number\_of\_Big++;  
 return fowl;  
 }  
 }  
  
 if (time % time\_birth\_small == 0)  
 {  
 if(number\_of\_Small < number\_of\_Big\*percent)  
 {  
 Bird fowl = new Small(place);  
 number\_of\_birds++;  
 number\_of\_Small++;  
 return fowl;  
 }  
 }  
 return null;  
 }  
  
}

package Habit;  
  
import DrawPanel.Depict\_a\_bird;  
import Fact.\*;  
import Object.\*;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.util.Random;  
  
public class Habitat extends JFrame  
{  
 Bird bird\_s[];  
 private Integer i = 0;  
 private Integer Mass = 100;   
  
 private final JLabel timer;  
  
 private final AbstractFactory factory;  
 private Integer time;  
  
 private final Depict\_a\_bird depict\_a\_bird;  
 private Integer number\_of\_Big;  
 private Integer number\_of\_Small;  
  
 public Habitat(String title, Integer time\_birth\_small, Integer time\_birth\_big, Float chance\_birth\_big, Float percent) {   
  
 super(title);   
 int width = 800;  
 int height = 700;  
 setSize(width, height);  
 setResizable(true);   
 setLayout(null);   
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);   
  
 this.time = 0;  
 bird\_s = new Bird [Mass];  
 factory = new ConcreteFactory(time\_birth\_small,time\_birth\_big,chance\_birth\_big,percent);  
  
  
 timer = new JLabel(factory.Return\_the\_Number\_of\_animals() + " Время: " + time);   
  
 JPanel labels = new JPanel();  
 labels.setLayout(new GridLayout(1,1));   
  
 labels.add(timer);   
 labels.setBounds(width-120,0, width,20);   
  
 depict\_a\_bird = new Depict\_a\_bird(bird\_s, Mass); // extends JPanel  
 depict\_a\_bird.setBounds(0,21, width, height);  
  
 add(labels);   
 add(depict\_a\_bird);  
  
 number\_of\_Big=0;  
 number\_of\_Small = 0;  
 }  
  
  
 public void Stop()  
 {  
 JLabel Result\_time = new JLabel("За " + time + " секунд(ы)");   
  
 Result\_time.setFont(new Font("Times New Roman", Font.*BOLD*,16));   
 Result\_time.setForeground(Color.*BLACK*);  
 Result\_time.setHorizontalAlignment(SwingConstants.*CENTER*);   
  
 JLabel Result\_small = new JLabel(" Приходили птенцы - " + number\_of\_Small + " ");  
 Result\_small.setFont(new Font("Courier New", Font.*PLAIN*,14));  
 Result\_small.setForeground(Color.*RED*);  
 Result\_small.setHorizontalAlignment(SwingConstants.*LEFT*);  
  
 JLabel Result\_big = new JLabel(" Прилетали взрослые - " + number\_of\_Big + " ");  
 Result\_big.setFont(new Font("Courier New", Font.*PLAIN*,14));  
 Result\_big.setForeground(Color.*RED*);  
 Result\_big.setHorizontalAlignment(SwingConstants.*LEFT*);  
  
 JDialog dialog = new JDialog(this, "Итог прогулки", false);   
 dialog.setLayout(null);   
 dialog.setResizable(false);   
 dialog.setDefaultCloseOperation(WindowConstants.*DISPOSE\_ON\_CLOSE*);   
  
 dialog.setSize(300, 200);  
 dialog.setLayout(new GridLayout(3,1));  
  
  
 dialog.add(Result\_time);  
 dialog.add(Result\_small);  
 dialog.add(Result\_big);  
 dialog.setVisible(true);   
  
 // Обнуление всего  
 factory.Total\_destruction();  
 number\_of\_Small = 0;  
 number\_of\_Big = 0;  
 for (int j=0; j<Mass; j++) {  
 bird\_s[j] = null;  
 }  
 i = 0;  
 time = 0;  
  
 depict\_a\_bird.repaint();  
 }  
  
  
 public void Clock\_yes\_no(Boolean isShown){   
 if (!isShown)  
 timer.setVisible(false);  
 else  
 timer.setVisible(true);   
 }  
  
 void Update()  
 {  
 time++;  
 timer.setText(factory.Return\_the\_Number\_of\_animals() + " Время: " + time);  
  
 Random coord = new Random();  
 int x\_Coord = coord.nextInt(depict\_a\_bird.getWidth()-50);  
 int y\_Coord = coord.nextInt(depict\_a\_bird.getHeight()-50);  
  
 Point coordinates = new Point(x\_Coord, y\_Coord);  
 try  
 {  
 Bird bird = factory.Luntik(time, coordinates); // Возвращает какую-то из птиц или нет  
 timer.setText(factory.Return\_the\_Number\_of\_animals() + " Время: " + time);  
  
 if(bird != null)  
 {  
 if (bird.getClass() == Big.class) {  
 number\_of\_Big++;  
 }  
 else {  
 number\_of\_Small++;  
 }  
  
 bird\_s[i] = bird;  
 i++;  
  
 depict\_a\_bird.repaint();  
 }  
 }  
 catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
}

package Habit;  
  
import java.util.TimerTask;  
  
public class ToBeContinued extends TimerTask {  
  
 private final Habitat habitat;  
  
 public ToBeContinued(Habitat habitat){  
 this.habitat = habitat;  
 }  
  
 @Override  
 public void run() {   
 habitat.Update();  
 }  
}

package Object;  
  
import java.awt.\*;  
import java.io.\*;  
import javax.imageio.\*;  
  
public class Big extends Bird  
{  
 public Big(Point place) throws IOException   
 {  
 super(place);  
 this.ris = ImageIO.*read*(getClass().getResource("/duck2.png"));   
 }  
  
}

package Object;   
  
import Behav.IBehaviour;  
import java.awt.\*;  
  
public abstract class Bird implements IBehaviour  
{  
 private Point place;  
 protected Image ris;  
  
 protected Bird(Point place) {  
 this.place = place;  
 }  
  
 @Override  
 public Image getImage() {  
 return ris;  
 }  
  
 @Override  
 public Point getPlace() {  
 return place;  
 }  
}

package Object;  
  
import java.awt.\*;  
import java.io.\*;  
import javax.imageio.\*;  
  
  
public class Small extends Bird  
{  
 public Small(Point place) throws IOException  
 {  
 super(place);  
 this.ris = ImageIO.*read*(getClass().getResource("/Small.png"));  
 }  
}

import Habit.\*;  
import java.awt.event.\*;  
import java.util.Timer;  
  
public class AnimalTour extends KeyAdapter {  
 private Timer timer;  
 private ToBeContinued habitatTask;  
 private final Habitat habitat;  
 private Boolean isStarted;  
 private Boolean isShown;  
  
 private final Integer time\_birth\_small = 2;  
 private final Integer time\_birth\_big = 5;  
 private final Float chance\_birth\_big = 0.8f;  
 private final Float percent = 0.5f;  
  
 private AnimalTour()  
 {  
 habitat = new Habitat("Field", time\_birth\_small, time\_birth\_big,chance\_birth\_big,percent);  
 habitat.setVisible(true);   
 habitat.addKeyListener(this);   
 isStarted = false;  
 isShown = true;  
 }  
  
 @Override  
 public void keyPressed(KeyEvent keyEvent)  
 {  
  
 if (keyEvent.getKeyCode() == KeyEvent.*VK\_B*)   
 {   
 isStarted = true;  
 timer = new Timer();  
 habitatTask = new ToBeContinued(habitat);  
 timer.schedule(habitatTask,0,1000);   
 }  
  
 if (keyEvent.getKeyCode() == KeyEvent.*VK\_E*){  
 timer.cancel();  
 habitatTask.cancel();  
 timer.purge();  
 habitat.Stop();  
 isStarted = false;  
 }  
  
 if(keyEvent.getKeyCode() == KeyEvent.*VK\_T*) {  
 isShown = !isShown;  
 habitat.Clock\_yes\_no(isShown);  
 }  
 }  
  
 public static void main(String[] args) {   
 new AnimalTour();  
 }  
}