МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**Лабораторная работа №1**

**по дисциплине «Технология программирования»**

Группа: АВТ-910

Студент: Воликов М.В.

Преподаватель: Дубков И.С.

Вариант: 4

НОВОСИБИРСК 2021

**Задание:**

Объект – кролик. Бывают 2 видов: обыкновенный и альбинос. Обыкновенные кролики рождаются каждые N1 секунд с вероятностью P1. Альбиносы рождаются каждые N2 секунд, при условии, что их количество менее K% от общего числа кроликов, в противном случае – не рождаются вовсе.

**Ход работы:**

Создаём интерфейс, определяющий поведение кроликов.

public interface IBehaviour {  
  
 void setX(double X);  
 double getX();  
 void setY(double Y);  
 double getY();  
  
}

Создаём абстрактный класс Кролик, согласно заданию варианта и имплементируем его от интерфейса поведения.

import javax.imageio.ImageIO;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
  
public abstract class Rabbit implements IBehaviour {  
  
 private static int *AllQuantity* = 0;  
 protected double X, Y;  
 protected BufferedImage img;  
  
 protected void openImage(String pathname) {  
 try {  
 img = ImageIO.*read*(new File(pathname));  
 } catch (IOException Ex) {  
 System.*out*.println(Ex);  
 }  
 };  
  
 public BufferedImage getImg() {  
 return img;  
 }  
  
 Rabbit() { *AllQuantity*++; }  
  
 Rabbit(double X, double Y) {  
 *AllQuantity*++;  
 this.X = X;  
 this.Y = Y;  
 }  
  
 public static int getAllQuantity() {  
 return *AllQuantity*;  
 }  
  
 @Override  
 public void setX(double X) { this.X = X; }  
 @Override  
 public double getX() { return X; }  
 @Override  
 public void setY(double Y) { this.Y = Y; }  
 @Override  
 public double getY() { return Y; }  
  
}

Создаём два класса наследника от супер-класса Кролик: кролик обычный и кролик альбинос.

public class Ordinary extends Rabbit {  
  
 private static int *OrdinaryQuantity* = 0;  
  
 Ordinary() { openImage("./images/Ordinary.png"); *OrdinaryQuantity*++; }  
 Ordinary(double X, double Y) {  
 super(X, Y);  
 openImage("./images/Ordinary.png");  
 *OrdinaryQuantity*++;  
 }  
  
 public static int getOrdinaryQuantity() {  
 return *OrdinaryQuantity*;  
 }  
  
}

public class Albino extends Rabbit {  
  
 private static int *AlbinoQuantity* = 0;  
  
 Albino() { openImage("./images/Albino.png"); *AlbinoQuantity*++; }  
 Albino(double X, double Y) {  
 super(X, Y);  
 openImage("./images/Albino.png");  
 *AlbinoQuantity*++;  
 }  
  
 public static int getAlbinoQuantity() {  
 return *AlbinoQuantity*;  
 }  
  
}

Для создания новых кроликов будем использовать паттерн абстрактная фабрика. Сначала создаём абстрактную фабрику.

public interface AbstactFactory {  
  
 Ordinary creatOrdinary();  
 Ordinary creatOrdinary(double X, double Y);  
  
 Albino creatAlbino();  
 Albino creatAlbino(double X, double Y);  
  
}

Создаём класс, имплементирующий данный интерфейс. Это будет фабрика кроликов.

public class RabbitFactory implements AbstactFactory {  
  
 @Override  
 public Ordinary creatOrdinary() { return new Ordinary(); }  
  
 @Override  
 public Ordinary creatOrdinary(double X, double Y) { return new Ordinary(X, Y); }  
  
 @Override  
 public Albino creatAlbino() { return new Albino(); }  
  
 @Override  
 public Albino creatAlbino(double X, double Y) { return new Albino(X, Y); }  
  
}

Теперь создаём класс среда обитания. Разместим в нём фабрику, слушатель событий и фрейм для визуализации имитации.

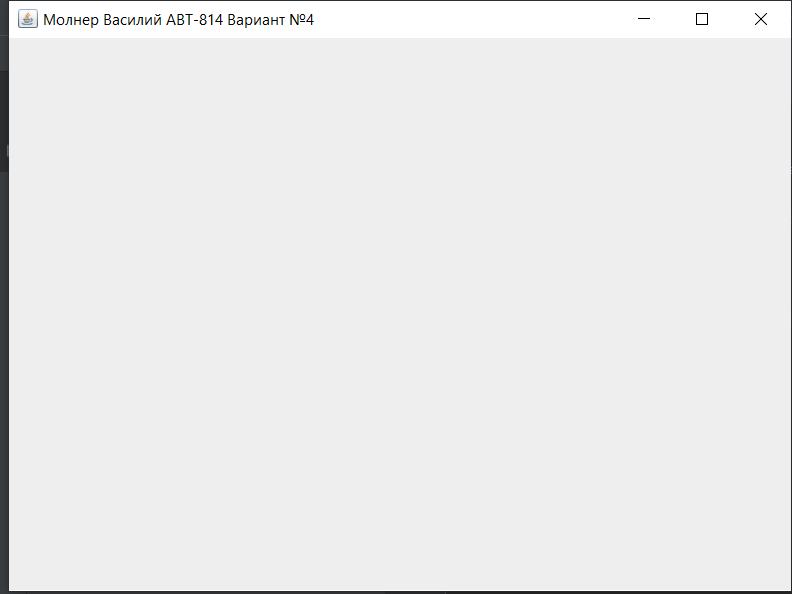
import javax.imageio.ImageIO;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class Habitat {  
  
 BufferedImage offScreenImage;  
  
 private int quantity = 0;  
 private Rabbit[] rabbit = new Rabbit[1000];  
 private JFrame frame = new JFrame();  
 private Timer timer;  
 private static BufferedImage *BackgroundImg*;  
 private long NOrdinary = 1000, NAlbino = 2000; // milliseconds  
 private double KAlbino = 0.7, POrdinary = 0.5; // [0, 1]  
 private final double RabbitSize = 0.15; // [0, 1]  
  
 private long cumulativePeriod = 0;  
 private final long period = 1000;  
  
 private AbstactFactory factory = new RabbitFactory();  
  
 Habitat(int Width, int Height) {  
 try {  
 *BackgroundImg* = ImageIO.*read*(new File("./Images/Field.jpg"));  
 } catch (IOException Ex) {  
 System.*out*.println(Ex);  
 }  
 frame.setTitle("Воликов Максим АВТ-910 Вариант №4");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setPreferredSize(new Dimension(Width, Height));  
 frame.pack();  
 frame.setVisible(true);  
 frame.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyTyped(KeyEvent e) {  
 super.keyTyped(e);  
 switch (e.getKeyChar()) {  
 case 'b':  
 timer = new Timer();  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 Update(cumulativePeriod);  
 cumulativePeriod += period;  
 }  
 }, 0, period);  
 break;  
 case 'e':  
 timer.cancel();  
 rabbit = new Rabbit[1000];  
 quantity = 0;  
 break;  
 case 't':  
 ShowText();  
 cumulativePeriod = 0;  
 break;  
 default:  
 break;  
 }  
 }  
 });  
 }  
  
 public void Update(long time) {  
 if (time % NOrdinary == 0 && Math.*random*() < POrdinary) rabbit[quantity++] = factory.creatOrdinary(Math.*random*(), Math.*random*());  
 if (time % NAlbino == 0 && (double)Albino.*getAlbinoQuantity*() / Rabbit.*getAllQuantity*() < KAlbino) rabbit[quantity++] = factory.creatAlbino(Math.*random*(), Math.*random*());  
 Render();  
 }  
  
 private void Render() {  
 //Двойная буфферизация. Рисую в BufferedImage, и только готовое вывожу в фрейм (устраняю мерцание)  
 int w = frame.getWidth()\*2, h = frame.getHeight()\*2; //умножаю на 2, потому что падает качество  
 offScreenImage = new BufferedImage(w, h, BufferedImage.*TYPE\_INT\_RGB*);  
 Graphics offScreenGraphics= offScreenImage.getGraphics();  
  
 offScreenGraphics.drawImage(*BackgroundImg*, 0,0, w, h,null);  
 for (int i = 0; i < quantity; i++) {  
 offScreenGraphics.drawImage(rabbit[i].getImg(), (int)(rabbit[i].getX()\*w),(int)(rabbit[i].getY()\*h), (int)(RabbitSize\*w), (int)(RabbitSize\*h), null);  
 }  
  
 //вывод изображение на фрейм  
 frame.getGraphics().drawImage(offScreenImage,0,0,w/2,h/2,null); //делю на 2, чтобы вернуться к размерам фрейма  
 }  
  
 public void ShowText() {  
 JTextArea jTextArea = new JTextArea();  
 jTextArea.setFont(new Font("TimesRoman", Font.*BOLD*, 30));  
 String string = new String("Информация о симуляции:" +  
 "\nКроликов всего " + Rabbit.*getAllQuantity*() +  
 "\nОбычных " + Ordinary.*getOrdinaryQuantity*() +  
 "\nАльбиносов " + Albino.*getAlbinoQuantity*() +  
 "\n\nВремени прошло " + cumulativePeriod + " миллисекунд");  
 jTextArea.setText(string);  
 frame.add(jTextArea);  
 frame.setVisible(true);  
 }  
}

В main создаём среду обитания.

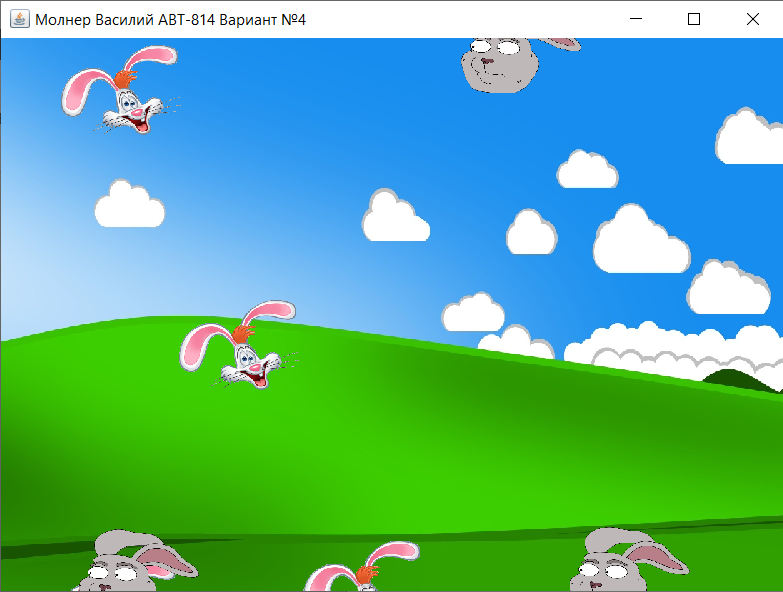
public class Main {  
  
 public static void main(String[] args) {  
  
 Habitat h = new Habitat(640,480);  
  
 }  
}

**Пример работы программы:**

Сразу после запуска.

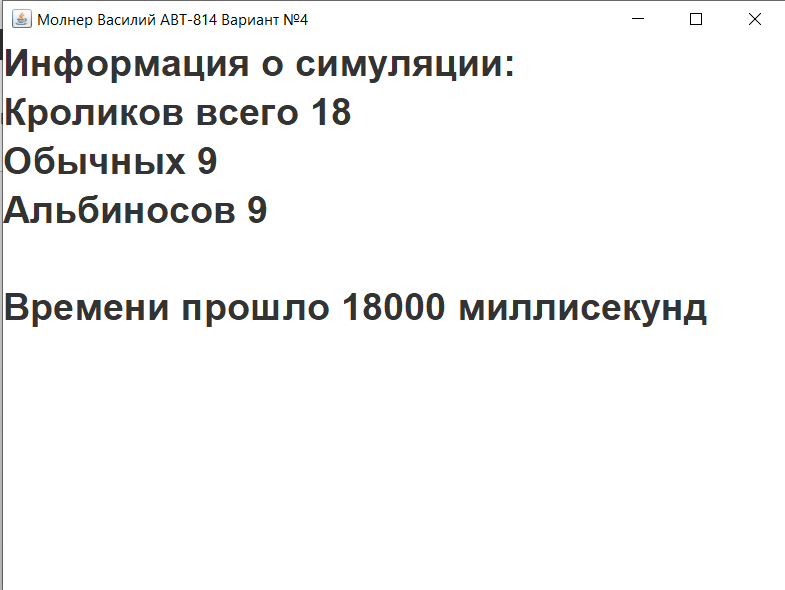


После нажатия на клавишу b.



После нажатия на клавишу e все останавливается.

По клавише t выводится информация об имитации.



**Вывод:**

В процессе выполнения работы появилась проблема в виде мигания картинки. Данную проблему решила двойная буферизация.