Кафедра вычислительной техники



**ЛАБОРАТОРНАЯ РАБОТА №2**

**по дисциплине:** *технология программирования на Java*

**на тему:** *Разработка графического интерфейса приложения.*

Факультет: АВТФ

Группа: АВТ-808

Студент: Сенотрусов С.Е.

Вариант 12

Преподаватель: Михайленко Дмитрий Анатольевич

# **ЦЕЛЬ И ЗАДАНИЕ**

# **Цель работы:**

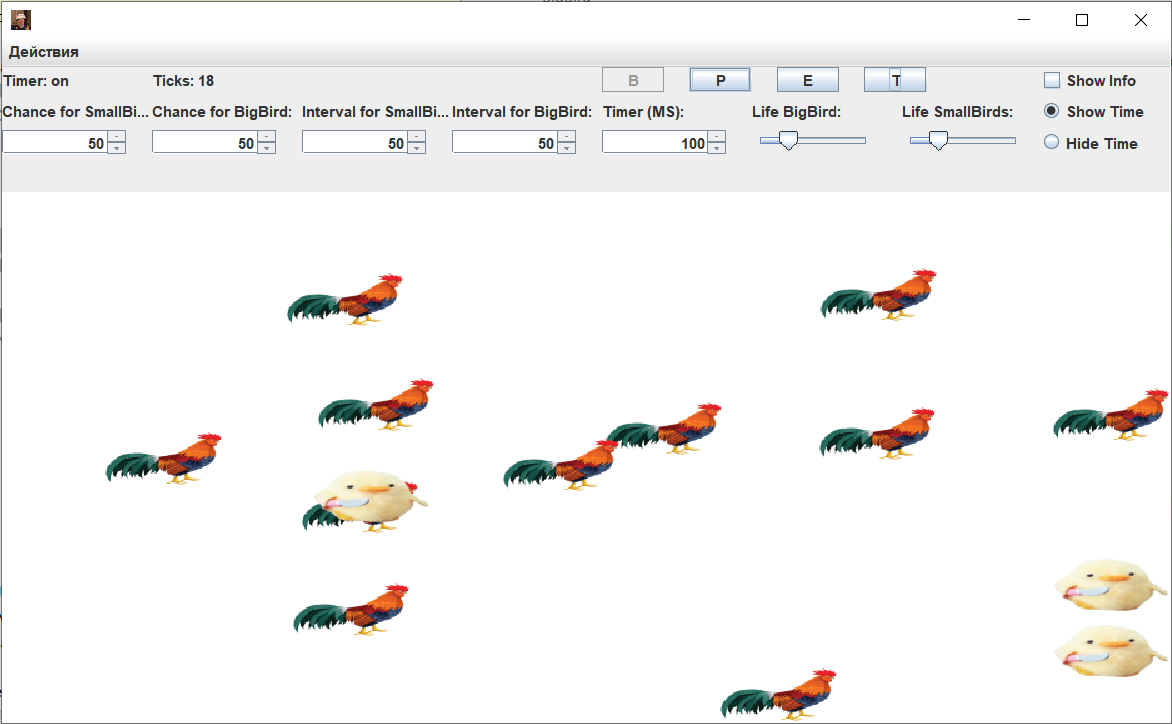
1. Познакомиться с основными графическими библиотеками Java - AWT и Swing и их основными компонентами. Изучить классы менеджеров компоновки, классы создания меню приложения, основных и диалоговых окон и т.д.
2. Для хранения генерируемых объектов использовать динамический массив объектов.
3. Доработать программу, созданную в лабораторной работе № 1:

**Вариант 12**

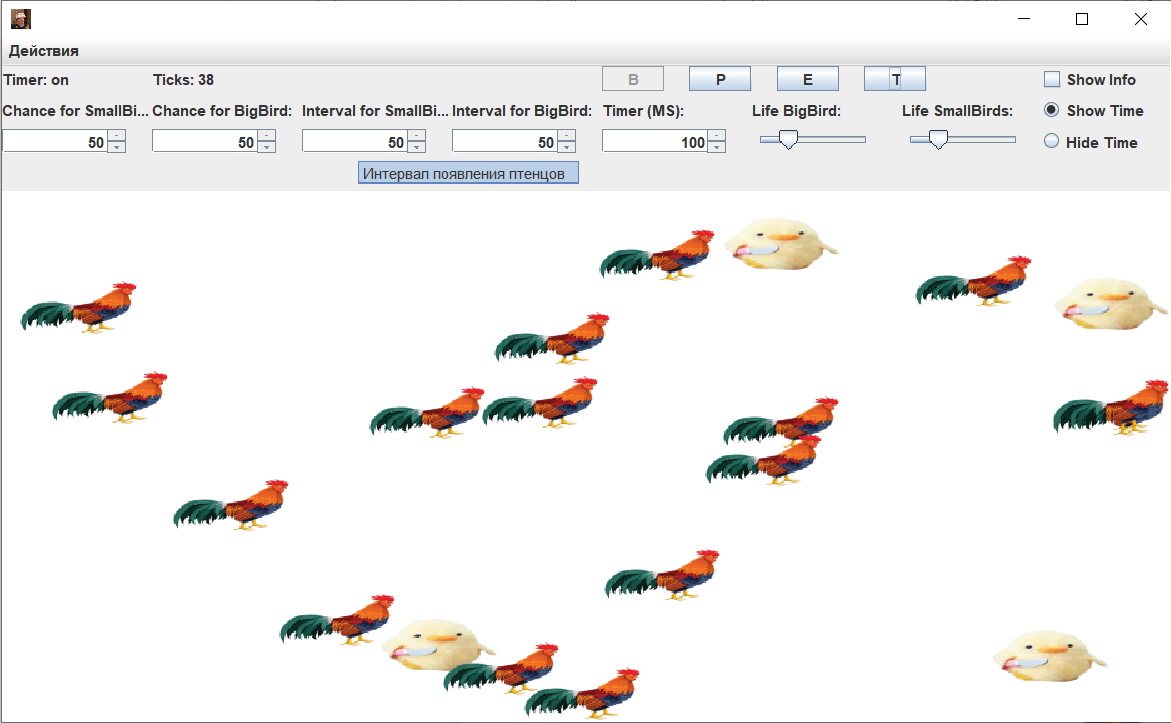
Объекты – птицы. Бывают 2 видов: птенцы и взрослые птицы. Взрослые птицы генерируются каждые N1 секунд с вероятностью P1. Птенцы генерируются каждые N2 секунд при условии, что их количество менее K% от общего числа взрослых птиц, в противном случае – не генерируются.

# **Задание**

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 значений)
10. Дополнить интерфейс поясняющими метками.



*Рис. 1. Результат выполнения функции кнопки «старт» (начало симуляции).*



*Рис. 2.Результат выполнения метода* setToolTipText

# Ход работы

В ходе 2 лабораторной работы были внесены изменения:

Использованы различные менеджеры компоновки:

- setLayout(new BorderLayout());

- setLayout (new FlowLayout());

- setLayout (new BoxLayout());

Созданы JComboBox, JSlider:

private JSpinner SmallBirdChance **=** **new** JSpinner**(new** SpinnerNumberModel**(**50.0**,** 1.0**,** 99.0**,** 1.0**));**

private JSpinner BigBirdChance **=** **new** JSpinner**(new** SpinnerNumberModel**(**50.0**,** 1.0**,** 99.0**,** 1.0**));**

private JSpinner textIntervalBig **=** **new** JSpinner**(new** SpinnerNumberModel**(**3.0**,** 1.0**,** 10.0**,** 1.0**));**

private JSpinner textIntervalLit **=** **new** JSpinner**(new** SpinnerNumberModel**(**2.0**,** 1.0**,** 10.0**,** 1.0**));**

private JSpinner textTimerInterval **=** **new** JSpinner**(new** SpinnerNumberModel**(**100**,** 10**,** 1000**,** 10**));**

private JSlider textLifeBig **=** **new** JSlider**(**0**,** 80**,** 20**);**

private JSlider textLifeLittle **=** **new** JSlider**(**0**,** 80**,** 20**);**

Добавлены поясняющие метки:

timerEnabled**.**setToolTipText**(**"Таймер"**);**

ticks**.**setToolTipText**(**"Время симуляции"**);**

cLit**.**setToolTipText**(**"Кол-во птенцов"**);**

cBig**.**setToolTipText**(**"Кол-во птиц"**);**

run**.**setToolTipText**(**"Запуск симуляции"**);**

pause**.**setToolTipText**(**"Пауза"**);**

stop**.**setToolTipText**(**"Остановка симуляции, очищение рабочей области"**);**

showTime**.**setToolTipText**(**"Скрыть/Показать время симуляции"**);**

showInfo**.**setToolTipText**(**"Окно информации"**);**

SmallBirdChance**.**setToolTipText**(**"Шанс появления птенцов"**);**

BigBirdChance**.**setToolTipText**(**"Шанс появления птиц"**);**

textIntervalLit**.**setToolTipText**(**"Интервал появления птенцов"**);**

textIntervalBig**.**setToolTipText**(**"Интервал появления птиц"**);**

textTimerInterval**.**setToolTipText**(**"Шаг таймера"**);**

textLifeLittle**.**setToolTipText**(**"Время жизни птенцов"**);**

textLifeBig**.**setToolTipText**(**"Время жизни птиц"**);**

# Вывод:

Была разработана программа упрощенной имитации появления на экране двух видов существ: птенцов и птиц. В ходе лабораторной работы были изучены особенности технологий Java и синтаксис языка. А также основные понятия и термины обработки событий в модели делегирования событий.

# Листинг:

/\*Package lab\*/

//\_\_\_\_\_Main\_\_\_\_\_//

package lab**;**

**import** lab**.**habitat**.**Habitat**;**

**import** lab**.**habitat**.**creatures**.**birds**.**BigBird**;**

**import** lab**.**habitat**.**creatures**.**birds**.**SmallBird**;**

**import** javax**.**swing**.\*;**

public class Main **{**

final static Habitat h **=** **new** Habitat**();**

final static lab**.**frames**.**MainFrames g **=** **new** lab**.**frames**.**MainFrames**(**h**);**

public static void main**(**String**[]** args**)** **{**

**try** **{**

h**.**addCreatureTypes**(**SmallBird**.**class**,** BigBird**.**class**);**

g**.**setBounds**(**100**,**100**,** 910**,** 524**);**

g**.**setDefaultCloseOperation**(**JFrame**.**EXIT\_ON\_CLOSE**);**

g**.**setVisible**(true);**

g**.**requestFocusInWindow**();**

**}**

**catch** **(**Exception ex**)** **{**

ex**.**printStackTrace**();**

**}**

**}**

**}**

/\*Package frames\*/

//\_\_\_\_\_Class\_MainFrames\_\_\_\_\_//

package lab**.**frames**;**

**import** lab**.**habitat**.**ICreature**;**

**import** lab**.**habitat**.**creatures**.**birds**.**BigBird**;**

**import** lab**.**habitat**.**creatures**.**birds**.**SmallBird**;**

**import** lab**.**habitat**.**Habitat**;**

**import** javax**.**swing**.\*;**

**import** java**.**awt**.\*;**

**import** java**.**awt**.**event**.**KeyAdapter**;**

**import** java**.**awt**.**event**.**KeyEvent**;**

**import** java**.**time**.**Duration**;**

**import** java**.**time**.**Instant**;**

**import** java**.**util**.**HashMap**;**

**import** java**.**util**.**Timer**;**

**import** java**.**util**.**TimerTask**;**

**import** static java**.**awt**.**Image**.**SCALE\_SMOOTH**;**

**import** static java**.**awt**.**event**.**KeyEvent**.\*;**

public class MainFrames **extends** JFrame **{**

private JLabel timerEnabled **=** **new** JLabel**(**"Timer: Off"**);**

private JLabel pauseEnabled **=** **new** JLabel**(**"Pause: Off"**);**

private JLabel cLit **=** **new** JLabel**(**"Count SmallBirds: 0"**);**

private JLabel cBig **=** **new** JLabel**(**"Count BigBird: 0"**);**

private JLabel ticks **=** **new** JLabel**(**"Ticks: 0"**);**

private JLabel optChLit **=** **new** JLabel**(**"Chance for SmallBirds: "**);**

private JLabel optChBig **=** **new** JLabel**(**"Chance for BigBird: "**);**

private JLabel optIntLit **=** **new** JLabel**(**"Interval for SmallBirds: "**);**

private JLabel optIntBig **=** **new** JLabel**(**"Interval for BigBird: "**);**

private JLabel optTimerInterval **=** **new** JLabel**(**"Timer (MS): "**);**

private JLabel optLifeBig **=** **new** JLabel**(**"Life BigBird: "**);**

private JLabel optLifeLittle **=** **new** JLabel**(**"Life SmallBirds: "**);**

private JSpinner SmallBirdChance **=** **new** JSpinner**(new** SpinnerNumberModel**(**50.0**,** 1.0**,** 99.0**,** 1.0**));**

private JSpinner BigBirdChance **=** **new** JSpinner**(new** SpinnerNumberModel**(**50.0**,** 1.0**,** 99.0**,** 1.0**));**

private JSpinner textIntervalBig **=** **new** JSpinner**(new** SpinnerNumberModel**(**3.0**,** 1.0**,** 10.0**,** 1.0**));**

private JSpinner textIntervalLit **=** **new** JSpinner**(new** SpinnerNumberModel**(**2.0**,** 1.0**,** 10.0**,** 1.0**));**

private JSpinner textTimerInterval **=** **new** JSpinner**(new** SpinnerNumberModel**(**100**,** 10**,** 1000**,** 10**));**

private JSlider textLifeBig **=** **new** JSlider**(**0**,** 80**,** 20**);**

private JSlider textLifeLittle **=** **new** JSlider**(**0**,** 80**,** 20**);**

private JButton run **=** **new** JButton**(**"B"**);**

private JButton pause **=** **new** JButton**(**"P"**);**

private JButton stop **=** **new** JButton**(**"E"**);**

private JButton showTime **=** **new** JButton**(**"T"**);**

private JButton showInfo **=** **new** JButton**(**"I"**);**

private JPanel creatureBox **=** **new** JPanel**();**

private HashMap**<**ICreature**,** JComponent**>** creatureBoxComponents **=** **new** HashMap**<>();**

private Habitat habitat**;**

private boolean running **=** **false;**

private Timer timer**;**

private Instant startPoint**,** stopPoint**,** pausePoint**;**

private Duration pauseTime **=** Duration**.**ZERO**;**

public MainFrames**(**Habitat h**)** **{**

habitat **=** h**;**

timerEnabled**.**setBounds**(**0**,**0**,**120**,**20**);**

pauseEnabled**.**setBounds**(**0**,**0**,**120**,**120**);**

ticks**.**setBounds**(**120**,**0**,**120**,**20**);**

cLit**.**setBounds**(**240**,**0**,**120**,**20**);**

cBig**.**setBounds**(**360**,**0**,**120**,**20**);**

optChLit**.**setBounds**(**0**,**25**,**120**,**20**);**

optChBig**.**setBounds**(**120**,**25**,**120**,**20**);**

optIntLit**.**setBounds**(**240**,**25**,**120**,**20**);**

optIntBig**.**setBounds**(**360**,**25**,**120**,**20**);**

optTimerInterval**.**setBounds**(**480**,** 25**,** 120**,** 20**);**

optLifeBig**.**setBounds**(**600**,** 25**,** 120**,** 20**);**

optLifeLittle**.**setBounds**(**720**,** 25**,** 120**,** 20**);**

SmallBirdChance**.**setBounds**(**0**,** 50**,** 100**,**20**);**

BigBirdChance**.**setBounds**(**120**,** 50**,** 100**,**20**);**

textIntervalLit**.**setBounds**(**240**,** 50**,** 100**,**20**);**

textIntervalBig**.**setBounds**(**360**,** 50**,** 100**,**20**);**

textTimerInterval**.**setBounds**(**480**,** 50**,** 100**,** 20**);**

textLifeBig**.**setBounds**(**600**,** 50**,** 100**,** 20**);**

textLifeLittle**.**setBounds**(**720**,** 50**,** 100**,** 20**);**

run**.**setBounds**(**480**,** 0**,** 50**,** 20**);**

pause**.**setBounds**(**550**,**0**,**50**,**20**);**

stop**.**setBounds**(**620**,**0**,**50**,**20**);**

showTime**.**setBounds**(**690**,**0**,**50**,**20**);**

showInfo**.**setBounds**(**760**,**0**,**50**,**20**);**

pause**.**setEnabled**(false);**

stop**.**setEnabled**(false);**

showInfo**.**setEnabled**(false);**

creatureBox**.**setLayout**(null);**

creatureBox**.**setBounds**(**0**,** 100**,** 840**,** 424**);**

timerEnabled**.**setToolTipText**(**"Таймер"**);**

ticks**.**setToolTipText**(**"Время симуляции"**);**

cLit**.**setToolTipText**(**"Кол-во птенцов"**);**

cBig**.**setToolTipText**(**"Кол-во птиц"**);**

run**.**setToolTipText**(**"Запуск симуляции"**);**

pause**.**setToolTipText**(**"Пауза"**);**

stop**.**setToolTipText**(**"Остановка симуляции, очищение рабочей области"**);**

showTime**.**setToolTipText**(**"Скрыть/Показать время симуляции"**);**

showInfo**.**setToolTipText**(**"Окно информации"**);**

SmallBirdChance**.**setToolTipText**(**"Шанс появления птенцов"**);**

BigBirdChance**.**setToolTipText**(**"Шанс появления птиц"**);**

textIntervalLit**.**setToolTipText**(**"Интервал появления птенцов"**);**

textIntervalBig**.**setToolTipText**(**"Интервал появления птиц"**);**

textTimerInterval**.**setToolTipText**(**"Шаг таймера"**);**

textLifeLittle**.**setToolTipText**(**"Время жизни птенцов"**);**

textLifeBig**.**setToolTipText**(**"Время жизни птиц"**);**

setFocusable**(true);**

setLayout**(null);**

setIconImage**(new** ImageIcon**(**"src/lab/assets/icon256.png"**).**getImage**());**

add**(**timerEnabled**);**

add**(**pauseEnabled**);**

add**(**ticks**);**

add**(**cLit**);**

add**(**cBig**);**

add**(**optChLit**);**

add**(**optChBig**);**

add**(**optIntLit**);**

add**(**optIntBig**);**

add**(**SmallBirdChance**);**

add**(**BigBirdChance**);**

add**(**textIntervalBig**);**

add**(**textIntervalLit**);**

add**(**optTimerInterval**);**

add**(**optLifeBig**);**

add**(**optLifeLittle**);**

add**(**run**);** // Кнопка старта

add**(**pause**);** // Кнопка паузы

add**(**stop**);** // Кнопка стоп

add**(**showTime**);** // Кнопка скрытия cSec

add**(**showInfo**);** // Кнопка информаци

add**(**textTimerInterval**);**

add**(**textLifeBig**);**

add**(**textLifeLittle**);**

add**(**creatureBox**);**

**for** **(**Component c **:** getComponents**())** **{**

c**.**setFocusable**(false);**

**}**

**((**JSpinner**.**DefaultEditor**)** SmallBirdChance**.**getEditor**()).**getTextField**().**setFocusable**(false);**

**((**JSpinner**.**DefaultEditor**)** BigBirdChance**.**getEditor**()).**getTextField**().**setFocusable**(false);**

**((**JSpinner**.**DefaultEditor**)** textIntervalBig**.**getEditor**()).**getTextField**().**setFocusable**(false);**

**((**JSpinner**.**DefaultEditor**)** textIntervalLit**.**getEditor**()).**getTextField**().**setFocusable**(false);**

**((**JSpinner**.**DefaultEditor**)** textTimerInterval**.**getEditor**()).**getTextField**().**setFocusable**(false);**

addKeyListener**(new** KeyAdapter**()** **{**

@Override

public void keyPressed**(**KeyEvent e**)** **{** // Бинд кнопок

**switch** **(**e**.**getKeyCode**())** **{**

**case** VK\_B**:** **{**

**if(**run**.**isEnabled**())**

run**.**doClick**();**

**break;**

**}**

**case** VK\_E**:** **{**

**if(**stop**.**isEnabled**())**

stop**.**doClick**();**

**break;**

**}**

**case** VK\_T**:** **{**

**if(**showTime**.**isEnabled**())**

showTime**.**doClick**();**

**break;**

**}**

**case** VK\_P**:** **{**

**if(**pause**.**isEnabled**())**

pause**.**doClick**();**

**break;**

**}**

**case** VK\_I**:** **{**

**if(**showInfo**.**isEnabled**())**

showInfo**.**doClick**();**

**break;**

**}**

**}**

**}**

**});**

// B

run**.**addActionListener**(**e **->** **{**

startTimer**();**

startPoint **=** Instant**.**now**();**

running **=** **true;**

run**.**setEnabled**(false);**

pause**.**setEnabled**(true);**

stop**.**setEnabled**(true);**

showInfo**.**setEnabled**(true);**

timerEnabled**.**setText**(**"Timer: on"**);**

**});**

// P

pause**.**addActionListener**(**e **->** **{**

**if(**running**)** **{**

stopTimer**();**

pausePoint **=** Instant**.**now**();**

**}**

**else** **{**

startTimer**();**

pauseTime**.**plus**(**Duration**.**between**(**pausePoint**,** Instant**.**now**()));**

**}**

timerEnabled**.**setText**(**"Timer: " **+** **(**running **?** "on" **:** "pause"**));**

running **=** **!**running**;**

**});**

// E

stop**.**addActionListener**(**e **->** **{**

stopTimer**();**

stopPoint **=** Instant**.**now**();**

**switch** **(**

JOptionPane**.**showOptionDialog**(this,**

"<html>" **+**

"<p>Кол-во Chick: " **+** h**.**getMetrics**().**getCreatureCountByType**(**SmallBird**.**class**)** **+** "</p>" **+**

"<p>Кол-во Adult: " **+** h**.**getMetrics**().**getCreatureCountByType**(**BigBird**.**class**)** **+** "</p>" **+**

"<p>Время симуляции:: " **+**

Duration**.**between**(**startPoint**,** stopPoint**).**minus**(**pauseTime**).**getSeconds**()** **+** "с" **+**

"</p>" **+**

"</html>"**,**

"Общая информация"**,**

JOptionPane**.**OK\_CANCEL\_OPTION**,**

JOptionPane**.**INFORMATION\_MESSAGE**,**

**new** ImageIcon**(**"src/lab/assets/icon64.png"**),**

**new** String**[]** **{**"Ok"**,**"Cancel" **},**

"Ok"

**))**

**{**

**case** JOptionPane**.**CLOSED\_OPTION**:**

pause**.**doClick**();**

**break;**

**}**

timer**.**cancel**();**

h**.**reset**();**

creatureBox**.**removeAll**();**

creatureBoxComponents**.**clear**();**

creatureBox**.**updateUI**();**

run**.**setEnabled**(true);**

pause**.**setEnabled**(false);**

stop**.**setEnabled**(false);**

timerEnabled**.**setText**(**"Timer: off"**);**

ticks**.**setText**(**"Sec: 0"**);**

**});**

// T

showTime**.**addActionListener**(**e **->**

ticks**.**setVisible**(!**ticks**.**isVisible**())**

**);**

// I

showInfo**.**addActionListener**(**e **->** **{**

stopTimer**();**

stopPoint **=** Instant**.**now**();**

**switch** **(**

JOptionPane**.**showOptionDialog**(this,**

"<html>" **+**

"<p>Кол-во Chick: " **+** h**.**getMetrics**().**getCreatureCountByType**(**SmallBird**.**class**)** **+** "</p>" **+**

"<p>Кол-во Adult: " **+** h**.**getMetrics**().**getCreatureCountByType**(**BigBird**.**class**)** **+** "</p>" **+**

"<p>Время симуляции:: " **+**

Duration**.**between**(**startPoint**,** stopPoint**).**minus**(**pauseTime**).**getSeconds**()** **+** "с" **+**

"</p>" **+**

"</html>"**,**

"Общая информация"**,**

JOptionPane**.**OK\_CANCEL\_OPTION**,**

JOptionPane**.**INFORMATION\_MESSAGE**,**

**new** ImageIcon**(**"src/lab/assets/icon64.png"**),**

**new** String**[]** **{**"Ok"**,**"Cancel" **},**

"Ok"

**))** **{**

**case** JOptionPane**.**OK\_OPTION**:**

stop**.**doClick**();** // to stop simulation. Maybe switch to separate method?

**break;**

**case** JOptionPane**.**YES\_NO\_CANCEL\_OPTION**:**

**case** JOptionPane**.**CLOSED\_OPTION**:**

pause**.**doClick**();**

**break;**

**}**

**});**

h**.**addCreatureCreateListener**((**creature**)** **->** **{**

JComponent component **=** **new** JComponent**()** **{**

**{**

setBounds**(**

**(**int**)(**creatureBox**.**getWidth**()** **\*** **(**creature**.**getX**()** **>** 0.9 **?** 0.9 **:** creature**.**getX**())),**

**(**int**)(**creatureBox**.**getHeight**()** **\*** **(**creature**.**getY**()** **>** 0.9 **?** 0.9 **:** creature**.**getY**())),**

**(**int**)(**creatureBox**.**getWidth**()** **\*** 0.1**),**

**(**int**)(**creatureBox**.**getHeight**()** **\*** 0.1**)**

**);**

setOpaque**(true);**

**}**

@Override

protected void paintComponent**(**Graphics g**)** **{**

**super.**paintComponent**(**g**);**

**try{**

g**.**drawImage**(**

creature**.**getImage**().**getScaledInstance**(**

getWidth**(),**

getHeight**(),**

SCALE\_SMOOTH

**),**

0**,** 0**,** **null**

**);**

**}**

**catch** **(**Exception ex**)** **{**

**}**

**}**

**};**

creatureBox**.**add**(**component**);**

creatureBoxComponents**.**put**(**creature**,** component**);**

creatureBox**.**updateUI**();**

**});**

h**.**addCreatureDieListener**((**creature**)** **->** **{**

creatureBox**.**remove**(**creatureBoxComponents**.**get**(**creature**));**

creatureBoxComponents**.**remove**(**creature**);**

creatureBox**.**updateUI**();**

**});**

**}**

private void startTimer**()** **{**

timer **=** **new** Timer**();**

timer**.**scheduleAtFixedRate**(new** TimerTask**()** **{**

public void run**()** **{**

habitat**.**update**();**

ticks**.**setText**(**"Ticks: " **+** habitat**.**getMetrics**().**getTime**());**

**}**

**},** 0**,** 100**);**

**}**

private void stopTimer**()** **{**

timer**.**cancel**();**

**}**

**}**

/\*Package habitat\*/

//\_\_\_\_\_Class\_Habitat\_\_\_\_\_//

package lab**.**habitat**;**

**import** lab**.**habitat**.**creatures**.**birds**.**BigBird**;**

**import** lab**.**habitat**.**creatures**.**birds**.**SmallBird**;**

**import** java**.**util**.\*;**

public class Habitat **{**

private LinkedList**<**CreatureEventListener**>** creatureCreateListeners **=** **new** LinkedList**<>();**

private LinkedList**<**CreatureEventListener**>** creatureDieListeners **=** **new** LinkedList**<>();**

private LinkedList**<**Class**<?** **extends** ICreature**>>** creatureTypes **=** **new** LinkedList**<>();**

private HabitatMetrics metrics **=** **new** HabitatMetrics**();**

private int ticks **=** 0**;**

private LinkedList**<**ICreature**>** creatures **=** **new** LinkedList**<>();**

private HashMap**<**Integer**,** LinkedList**<**ICreature**>>** creaturesTTLs **=** **new** HashMap**();**

public void update**()** **{**

ticks**++;**

metrics**.**incrementTime**();**

creatureTypes**.**forEach**(**T **->** **{**

**try** **{**

**if((**Boolean**)**T**.**getMethod**(**"isCreationAllowed"**,** IHabitatMetrics**.**class**).**invoke**(null,** metrics**))** **{**

// TODO: search for params constructor?

ICreature c **=** T**.**getConstructor**().**newInstance**();**

c**.**setX**(**Math**.**random**());**

c**.**setY**(**Math**.**random**());**

creatures**.**add**(**c**);**

LinkedList**<**ICreature**>** cs **=** creaturesTTLs**.**getOrDefault**(**ticks **+** c**.**getTTL**(),** **new** LinkedList**<>());**

cs**.**add**(**c**);**

creaturesTTLs**.**put**(**ticks **+** c**.**getTTL**(),** cs**);**

metrics**.**incrementCreatureCount**(**T**);**

creatureCreateListeners**.**forEach**(**e **->** e**.**onEvent**(**c**));**

**}**

**}**

**catch(**Exception ex**)** **{**

ex**.**printStackTrace**();**

**}**

**});**

LinkedList**<**ICreature**>** cs **=** creaturesTTLs**.**getOrDefault**(**ticks**,** **null);**

**if(**cs **!=** **null)** **{**

cs**.**forEach**(**c **->** **{**

creatures**.**removeFirstOccurrence**(**c**);**

metrics**.**decrementCreatureCount**(**c**.**getClass**());**

creatureDieListeners**.**forEach**(**e **->** e**.**onEvent**(**c**));**

**});**

creaturesTTLs**.**remove**(**ticks**);**

**}**

**}**

public void reset**()** **{**

ticks **=** 0**;**

metrics **=** **new** HabitatMetrics**();**

creatures**.**clear**();**

creaturesTTLs**.**clear**();**

**}**

public IHabitatMetrics getMetrics**()** **{**

**return** metrics**;**

**}**

public void addCreatureType**(**Class creatureType**)** **throws** IllegalArgumentException **{**

**if(!**ICreature**.**class**.**isAssignableFrom**(**creatureType**))** **{**

**throw** **new** IllegalArgumentException**(**creatureType**.**getName**()** **+** " doesn't implement Creature interface!"**);**

**}**

**if(**creatureTypes**.**contains**(**creatureType**))** **{**

**throw** **new** IllegalArgumentException**(**creatureType**.**getName**()** **+** " already added to current habitat"**);**

**}**

creatureTypes**.**add**(**creatureType**);**

**}**

public void addCreatureTypes**(**Class **...** creatureTypes**)** **throws** IllegalArgumentException **{**

**for** **(**Class creatureType **:** creatureTypes**)** **{**

addCreatureType**(**creatureType**);**

**}**

**}**

public void removeCreatureType**(**Class creatureType**)** **{**

creatureTypes**.**remove**(**creatureType**);**

**}**

public void removeCreatureTypes**(**Class **...** creatureTypes**)** **throws** IllegalArgumentException **{**

**for** **(**Class type **:** creatureTypes**)** **{**

removeCreatureType**(**type**);**

**}**

**}**

public void addCreatureCreateListener**(**CreatureEventListener e**)** **{**

creatureCreateListeners**.**add**(**e**);**

**}**

public void addCreatureDieListener**(**CreatureEventListener e**)** **{**

creatureDieListeners**.**add**(**e**);**

**}**

protected class HabitatMetrics **implements** IHabitatMetrics **{**

private int creatureCount **=** 0**;**

private int ticks **=** 0**;**

private HashMap**<**Class**<?** **extends** ICreature**>,** Integer**>** creatureCountByType **=** **new** HashMap**<>();**

public int getCreatureCount**()** **{**

**return** creatureCount**;**

**}**

public int getCreatureCountByType**(**Class**<?** **extends** ICreature**>** type**)** **{**

**return** creatureCountByType**.**getOrDefault**(**type**,** 0**);**

**}**

public int getTime**()** **{**

**return** ticks**;**

**}**

public void incrementCreatureCount**(**Class**<?** **extends** ICreature**>** type**)** **{**

**++**creatureCount**;**

int count **=** creatureCountByType**.**getOrDefault**(**type**,** 0**);**

creatureCountByType**.**put**(**type**,** **++**count**);**

**}**

public void decrementCreatureCount**(**Class**<?** **extends** ICreature**>** type**)** **{**

**--**creatureCount**;**

int count **=** creatureCountByType**.**get**(**type**);**

creatureCountByType**.**put**(**type**,** **--**count**);**

**}**

public void incrementTime**()** **{**

**++**ticks**;**

**}**

**}**

**}**

//\_\_\_\_\_Interface\_CreatureEventListener\_\_\_\_\_//

package lab**.**habitat**;**

public interface CreatureEventListener **{**

void onEvent**(**ICreature ICreature**);**

**}**

//\_\_\_\_\_Interface\_IBehaviour\_\_\_\_\_//

package lab**.**habitat**;**

public interface IBehaviour **{**

double getX**();**

double getY**();**

void setX**(**double x**);**

void setY**(**double y**);**

**}**

//\_\_\_\_\_Interface\_ICreature\_\_\_\_\_//

package lab**.**habitat**;**

**import** java**.**awt**.**image**.**BufferedImage**;**

public interface ICreature **extends** IBehaviour **{**

static boolean isCreationAllowed**(**IHabitatMetrics metrics**)** **{**

**throw** **new** UnsupportedOperationException**(**"isCreationAllowed must be implemented"**);**

**}**

BufferedImage getImage**();**

int getTTL**();**

**}**

//\_\_\_\_\_Interface\_IHabitatMetrics\_\_\_\_\_//

package lab**.**habitat**;**

public interface IHabitatMetrics **{**

int getCreatureCount**();**

int getCreatureCountByType**(**Class**<?** **extends** ICreature**>** type**);**

int getTime**();**

**}**

/\*Package Creatures\*/

//\_\_\_\_\_Abstract\_class\_Bird\_\_\_\_\_//

package lab**.**habitat**.**creatures**;**

**import** lab**.**habitat**.**IBehaviour**;**

**import** lab**.**habitat**.**ICreature**;**

public abstract class Bird **implements** IBehaviour**,** ICreature **{**

protected double X **=** 0**;**

protected double Y **=** 0**;**

public Bird**()** **{**

**}**

public Bird**(**double x**,** double y**)** **{**

X **=** x**;**

Y **=** y**;**

**}**

public double getX**(){**

**return** **this.**X**;**

**}**

public double getY**(){**

**return** **this.**Y**;**

**}**

public void setX**(**double x**){**

**this.**X **=** x**;**

**}**

public void setY**(**double y**){**

**this.**Y **=** y**;**

**}**

**}**

/\*Package birds\*/

//\_\_\_\_\_Class\_BigBird\_\_\_\_\_//

package lab**.**habitat**.**creatures**.**birds**;**

**import** lab**.**habitat**.**Habitat**;**

**import** lab**.**habitat**.**IHabitatMetrics**;**

**import** lab**.**habitat**.**creatures**.**Bird**;**

**import** javax**.**imageio**.**ImageIO**;**

**import** java**.**awt**.**image**.**BufferedImage**;**

**import** java**.**io**.**File**;**

**import** java**.**io**.**IOException**;**

public class BigBird **extends** Bird **{**

private static BufferedImage image**;**

private static int ttl **=** 20**;**

private static double bornChance **=** 0.2**;**

static **{**

**try** **{**

image **=** ImageIO**.**read**(new** File**(**"src/lab/assets/habitat/creatures/birds/BigBird.png"**));**

**}**

**catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

public static boolean isCreationAllowed**(**IHabitatMetrics metrics**)** **{**

**return** Math**.**random**()** **<** bornChance**;**

**}**

public BigBird**()** **{**

**super();**

**}**

public BigBird**(**double x**,** double y**)** **{**

**super(**x**,** y**);**

**}**

public BufferedImage getImage**()** **{**

**return** image**;**

**}**

public int getTTL**()** **{**

**return** ttl**;**

**}**

**}**

/\*Package birds\*/

//\_\_\_\_\_Class\_SmallBird\_\_\_\_\_//

package lab**.**habitat**.**creatures**.**birds**;**

**import** lab**.**habitat**.**Habitat**;**

**import** lab**.**habitat**.**IHabitatMetrics**;**

**import** lab**.**habitat**.**creatures**.**Bird**;**

**import** javax**.**imageio**.**ImageIO**;**

**import** java**.**awt**.**image**.**BufferedImage**;**

**import** java**.**io**.**File**;**

**import** java**.**io**.**IOException**;**

public class SmallBird **extends** Bird **{**

private static BufferedImage image**;**

private static int ttl **=** 50**;**

private static double critical **=** 0.3**;**

static **{**

**try** **{**

image **=** ImageIO**.**read**(new** File**(**"src/lab/assets/habitat/creatures/birds/SmallBird.png"**));**

**}**

**catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

System**.**exit**(**1**);**

**}**

**}**

public static boolean isCreationAllowed**(**IHabitatMetrics metrics**)** **{**

int SmallBird\_count **=** metrics**.**getCreatureCountByType**(**SmallBird**.**class**);**

**return** SmallBird\_count **==** 0 **||** **(**double**)** metrics**.**getCreatureCount**()** **/** SmallBird\_count **<** critical**;**

**}**

public SmallBird**()** **{**

**super();**

**}**

public SmallBird**(**double x**,** double y**)** **{**

**super(**x**,** y**);**

**}**

public BufferedImage getImage**()** **{**

**return** image**;**

**}**

public int getTTL**()** **{**

**return** ttl**;**

**}**

**}**