Кафедра вычислительной техники



**ЛАБОРАТОРНАЯ РАБОТА №3**

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

**на тему:** *Классы-колекции.*

Факультет: АВТФ

Группа: АВТ-808

Студент: Сенотрусов С.Е.

Вариант 12

Преподаватель: Михайленко Дмитрий Анатольевич

# **ЦЕЛЬ И ЗАДАНИЕ**

# **Цель работы:**

1. Изучить особенности реализации классов-коллекций в Java.
2. Доработать программу, созданную в лабораторной работе № 2:

**Вариант 12**

Коллекция для хранения объектов: LinkedList

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

# **Задание**

1. Добавить генерируемым объектам понятия «время рождения» и «время жизни». Время рождения устанавливается в момент генерации объекта и по значению соответствует времени, прошедшему от начала симуляции. Время жизни – время, через которое объект должен исчезнуть, считая от времени рождения;
2. Вынести установку параметров времени жизни объектов в пользовательский интерфейс. Для каждого типа объекта должно задаваться собственное время. Рекомендуется использовать текстовые поля, но следуют помнить о проверке на ввод некорректных данных;
3. Добавить генерируемым объектам уникальные целочисленные идентификаторы (случайные числа), которые назначаются при генерации объекта. Для хранения сгенерированных идентификаторов используйте коллекцию удобную для поиска по варианту;
4. Использовать коллекции по варианту. При генерации объекта происходит добавление его в коллекцию (в класс добавить поле идентификатора), а во вторую коллекцию: идентификаторы существующих объектов, в третью идентификатор + время рождения. При возникновении события по таймеру обойдите коллекцию и удалите все объекты, время жизни которых истекло, а также все данные во вспомогательных коллекциях;
5. Добавьте в панель управления кнопку «Текущие объекты». По нажатию на эту кнопку появляется модальное диалоговое окно, содержащее список всех «живых» объектов на момент нажатия со временем их рождения (тип объекта, время рождения, идентификатор). В класс диалогового окна должна передаваться коллекция с хранением объектов по времени рождения. Типы коллекций задаются вариантом.

# **Ход работы**

Были доработаны следующие классы:

Habitat – класс среды, отвечает за генерацию объектов;

**private** LinkedList<ICreature> creatures = **new** LinkedList<>(); - список для хранения существ.

**private** TreeSet<Integer> creaturesIDs = **new** TreeSet<>(); - для хранения уникальных идентификаторов питомцев.

**private** HashMap<Integer, LinkedList<ICreature>> creaturesTTLs = **new** HashMap(); - для хранения времени рождения объектов.

Реализация ICreature:

creatureTypes.forEach(T -> {

**try** {

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

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());

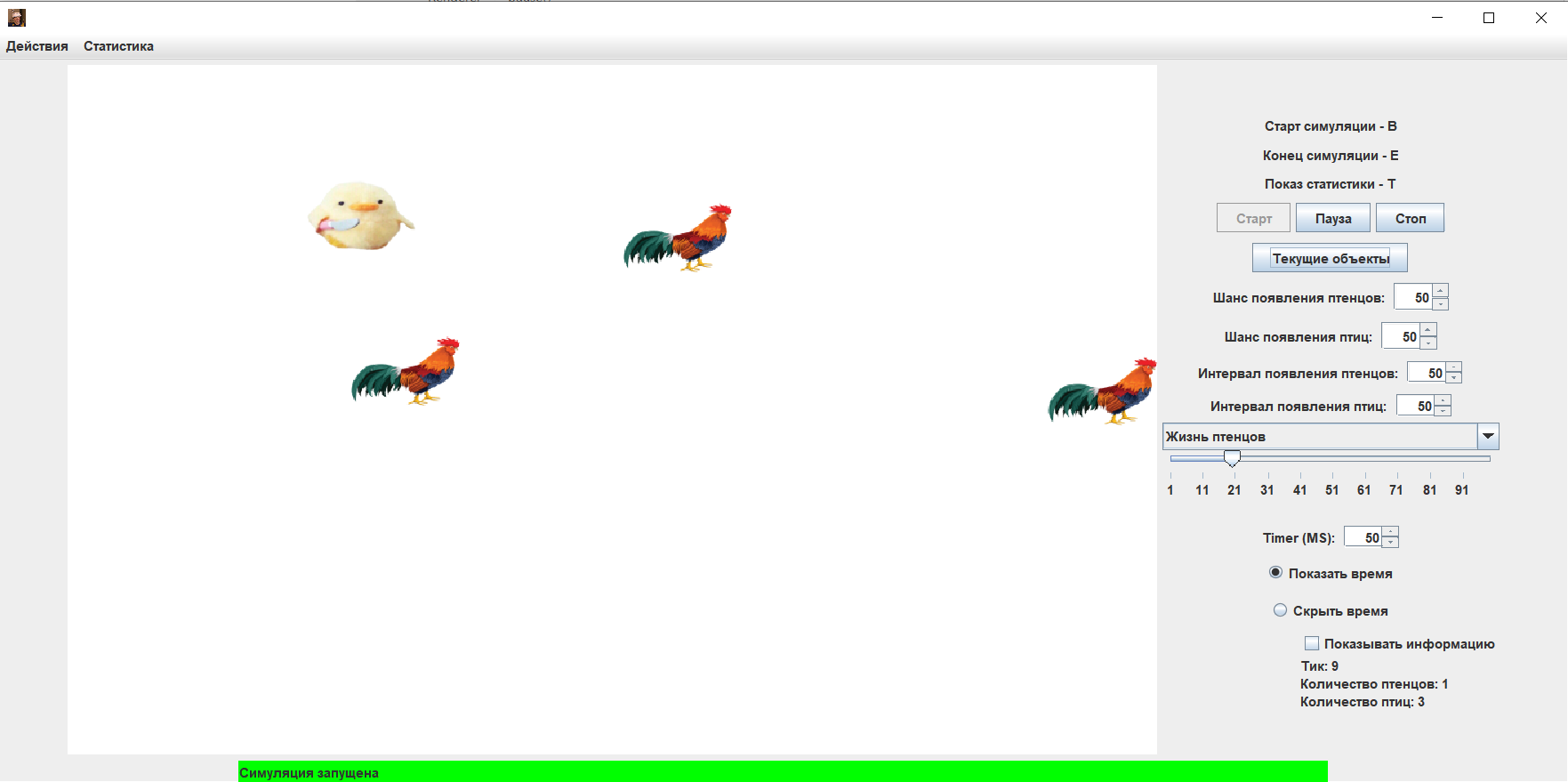
creatureDieListeners.forEach(e -> e.onEvent(c));

});

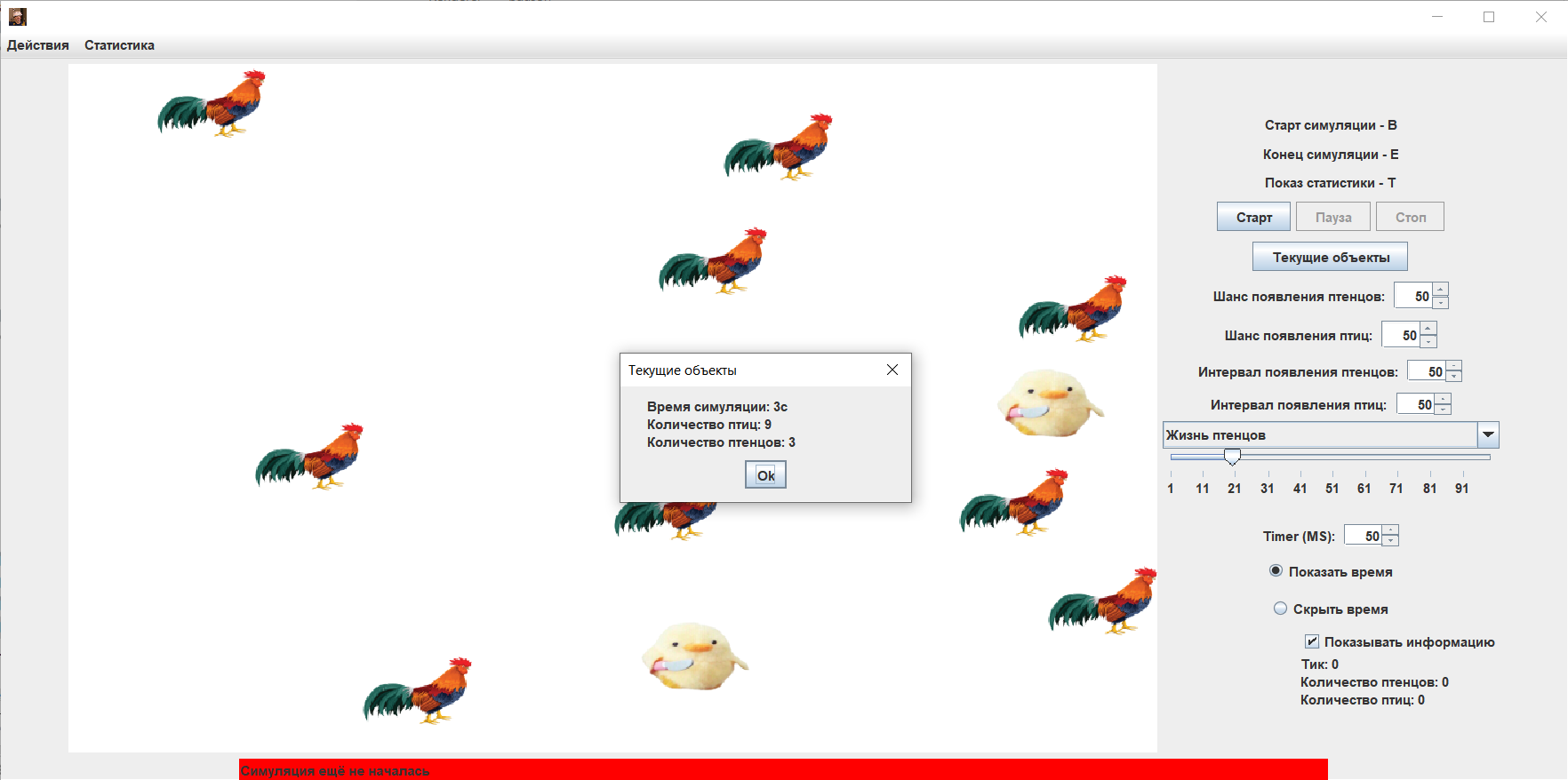
creaturesTTLs.remove(ticks);

}

}



*Рис. 1. Процесс нахождения симуляции после остановки, кнопка “E”*



*Рис. 2.. Процесс симуляции, кнопка “B”*

# Вывод:

В ходе лабораторной работы были изучены особенности реализации классов-коллекций в Java.

# Листинг:

/\*Package lab\*/

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

package lab**;**

**import** lab**.**frames**.**MainForm**;**

**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 MainForm g **=** **new** MainForm**(**h**);**

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

**try** **{**

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

g**.**setBounds**(**100**,**100**,** 1425**,** 710**);**

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

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

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

**}**

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

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

**}**

**}**

**}**

//\_\_\_\_\_Renderer\_\_\_\_\_//

package lab**;**

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

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

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

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

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

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

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

public class Renderer **extends** Thread **implements** Runnable **{**

private JPanel surface**;**

private Habitat habitat**;**

boolean active **=** **false;**

Timer timer**;**

public Renderer**(**JPanel s**,** Habitat h**)** **{**

surface **=** s**;**

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

**}**

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

active **=** **true;**

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

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

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

render**();**

**}**

**},** 0**,** 200**);**

**}**

public void pause**(**boolean b**)** **{**

active **=** **!**b**;**

**}**

public void render**()** **{**

**if(**active**)** **{**

surface**.**removeAll**();**

**for(**ICreature c **:** habitat**.**getCreatures**())** **{**

surface**.**add**(new** JPanel**()** **{**

**{**

setBounds**(**

**(**int**)** **(**surface**.**getWidth**()** **\*** **(**c**.**getX**()** **>** 0.9 **?** 0.9 **:** c**.**getX**())),**

**(**int**)** **(**surface**.**getHeight**()** **\*** **(**c**.**getY**()** **>** 0.9 **?** 0.9 **:** c**.**getY**())),**

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

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

**);**

setOpaque**(true);**

**}**

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

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

c**.**getImage**().**getScaledInstance**(**

getWidth**(),**

getHeight**(),**

SCALE\_SMOOTH

**),**

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

**);**

**}**

**});**

**}**

surface**.**updateUI**();**

**}**

**}**

**}**

/\*Package frames\*/

//\_\_\_\_\_Class\_MainForm\_\_\_\_\_//

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

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

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

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

**import** java**.**io**.\*;**

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

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

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

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

**import** lab**.**Renderer**;**

**import** lab**.**frames**.**components**.**ControlBox**;**

**import** lab**.**frames**.**components**.**StatusBar**;**

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

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

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

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

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

**import** static lab**.**Checked**.**CheckedActionListener**;**

public class MainForm **extends** JFrame **{**

private JMenuBar menuBar **=** **new** JMenuBar**();**

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

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

private ControlBox controlBox **=** **new** ControlBox**();**

private StatusBar statusBar **=** **new** StatusBar**();**

private Habitat habitat**;**

private Timer timer**;**

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

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

private Renderer renderer**;**

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

private boolean showingInfo **=** **true;**

private boolean showingStats **=** **false;**

public MainForm**(**Habitat h**)** **{**

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

renderer **=** **new** Renderer**(**creatureBox**,** habitat**);**

// habitat settings

habitat**.**addCreatureCreateListener**(**e **->** **{**

**if(**e **instanceof** BigBird**)**

controlBox**.**cBig**.**setText**(**"Количество птиц: " **+** h**.**getMetrics**().**getCreatureCountByType**(**BigBird**.**class**));**

**if(**e **instanceof** SmallBird**)**

controlBox**.**cLit**.**setText**(**"Количество птенцов: " **+** h**.**getMetrics**().**getCreatureCountByType**(**SmallBird**.**class**));**

**});**

habitat**.**addCreatureDieListener**(**e **->** **{**

**if(**e **instanceof** BigBird**)**

controlBox**.**cBig**.**setText**(**"Количество птиц: " **+** h**.**getMetrics**().**getCreatureCountByType**(**BigBird**.**class**));**

**if(**e **instanceof** SmallBird**)**

controlBox**.**cLit**.**setText**(**"Количество птенцов: " **+** h**.**getMetrics**().**getCreatureCountByType**(**SmallBird**.**class**));**

**});**

KeyboardFocusManager**.**getCurrentKeyboardFocusManager**().**addKeyEventPostProcessor**(**

e **->** **{**

**if(**KeyEvent**.**KEY\_PRESSED **==** e**.**getID**())**

keyPressed**(**e**);**

**return** **false;**

**}**

**);**

// frame settings

setFocusable**(true);**

setFocusTraversalKeysEnabled**(false);**

setLayout**(new** BorderLayout**());**

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

setJMenuBar**(**menuBar**);**

add**(**frame**,** BorderLayout**.**CENTER**);**

// frame settings

frame**.**add**(**creatureBox**,** BorderLayout**.**CENTER**);**

frame**.**add**(**controlBox**,** BorderLayout**.**EAST**);**

frame**.**add**(**statusBar**,** BorderLayout**.**SOUTH**);**

// statusBar settings

statusBar**.**setStatus**(**"Симуляция ещё не началась"**,** Color**.**RED**);**

statusBar**.**setPreferredSize**(new** Dimension**(**980**,**20**));**

// creatureBox settings

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

creatureBox**.**setBackground**(**Color**.**WHITE**);**

creatureBox**.**setPreferredSize**(new** Dimension**(**980**,**620**));**

// controlBox settings

controlBox**.**run**.**addActionListener**(**e **->** runSimulation**());**

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

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

**if(**running**)** pauseSimulation**();** **else** resumeSimulation**();**

**});**

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

controlBox**.**stop**.**addActionListener**(**e **->** stopSimulation**());**

controlBox**.**showTimeOn**.**setSelected**(true);**

controlBox**.**showTimeOn**.**addActionListener**(**e **->** controlBox**.**ticks**.**setVisible**(true));**

controlBox**.**showTimeOff**.**addActionListener**(**e **->** controlBox**.**ticks**.**setVisible**(false));**

controlBox**.**BigBirdChance**.**addChangeListener**(**e **->** BigBird**.**setBornChance**((**Integer**)** controlBox**.**BigBirdChance**.**getValue**()** **/** 100.0**));**

controlBox**.**SmallBirdChance**.**addChangeListener**(**e **->** SmallBird**.**setBornChance**((**Integer**)** controlBox**.**BigBirdChance**.**getValue**()** **/** 100.0**));**

controlBox**.**sliderLifeBig**.**addChangeListener**(**e **->** BigBird**.**setTTL**(**controlBox**.**sliderLifeBig**.**getValue**()));**

controlBox**.**sliderLifeLit**.**addChangeListener**(**e **->** SmallBird**.**setTTL**(**controlBox**.**sliderLifeLit**.**getValue**()));**

controlBox**.**showStats**.**addActionListener**(**e **->** showingStats **=** **!**showingStats**);**

controlBox**.**lifecreaturesBox**.**addActionListener**(**e**->** **{**

**if** **(**controlBox**.**lifecreaturesBox**.**getSelectedIndex**()** **==** 0**)** **{**

controlBox**.**sliderLifeLit**.**setVisible**(true);**

controlBox**.**sliderLifeBig**.**setVisible**(false);**

**}**

**if** **(**controlBox**.**lifecreaturesBox**.**getSelectedIndex**()** **==** 1**)** **{**

controlBox**.**sliderLifeLit**.**setVisible**(false);**

controlBox**.**sliderLifeBig**.**setVisible**(true);**

**}**

**});**

controlBox**.**current**.**addActionListener**(**e **->** **{**

pauseSimulation**();**

StringBuffer sb **=** **new** StringBuffer**();**

var creatureTTLs **=** habitat**.**getCreaturesTTLs**();**

**for** **(**Integer time **:** creatureTTLs**.**keySet**())** **{**

**for** **(**ICreature c **:** creatureTTLs**.**get**(**time**))** **{**

sb**.**append**(**String**.**format**(**"<p>Тип: %s, Умрет: %d</p>"**,** c**.**getClass**().**getSimpleName**(),** time**));**

**}**

**}**

String**[]** options **=** **{** "Ok" **};**

JOptionPane**.**showOptionDialog**(**

**this,**

"<html>" **+** sb **+** "</html>"**,**

"Текущие объекты"**,**

JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**,**

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

options**,**

options**[**0**]**

**);**

resumeSimulation**();**

**});**

JMenu actionMenu **=** **new** JMenu**(**"Действия"**);**

actionMenu**.**add**(new** JMenuItem**(**"Запустить симуляцию"**){{**

addActionListener**(**CheckedActionListener**(**

e **->** **{**

setText**(**running **?** "Остановить симуляцию" **:** "Запустить симуляцию"**);**

**if(**running**)** stopSimulation**();** **else** runSimulation**();**

**}**

**));**

**}});**

menuBar**.**add**(**actionMenu**);**

JMenu statMenu **=** **new** JMenu**(**"Статистика"**);**

statMenu**.**add**(new** JMenuItem**(**"Скрыть"**){{**

addActionListener**(**CheckedActionListener**(**

e **->** **{**

setText**(**showingInfo **?** "Показать" **:** "Скрыть"**);**

controlBox**.**showInfoPanel**.**setVisible**(**showingInfo **=** **!**showingInfo**);**

**}**

**));**

**}});**

menuBar**.**add**(**statMenu**);**

renderer**.**run**();**

**}**

private void runSimulation**()** **{**

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

habitat**.**reset**();**

startTimer**();**

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

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

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

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

statusBar**.**setStatus**(**"Симуляция запущена"**,** Color**.**GREEN**);**

**}**

private void pauseSimulation**()** **{**

running **=** **false;**

stopTimer**();**

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

statusBar**.**setStatus**(**"Симуляция на паузе"**,** Color**.**YELLOW**);**

**}**

private void resumeSimulation**()** **{**

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

startTimer**();**

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

statusBar**.**setStatus**(**"Симуляция запущена"**,** Color**.**GREEN**);**

**}**

private void stopSimulation**()** **{**

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

running **=** **false;**

stopTimer**();**

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

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

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

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

controlBox**.**ticks**.**setText**(**"Тик: 0"**);**

controlBox**.**cBig**.**setText**(**"Количество птиц: 0"**);**

controlBox**.**cLit**.**setText**(**"Количество птенцов: 0"**);**

statusBar**.**setStatus**(**"Симуляция ещё не началась"**,** Color**.**RED**);**

**if(**showingStats**)** showStatistics**();**

habitat**.**reset**();**

**}**

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

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

**case** VK\_B**:** **{** // Бинд кнопки "В"

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

stopSimulation**();**

runSimulation**();**

**}**

**break;**

**}**

**case** VK\_E**:** **{** // Бинд кнопки "Е"

stopSimulation**();**

**break;**

**}**

**case** VK\_T**:** **{** // Бинд кнопки "Т"

controlBox**.**showInfoPanel**.**setVisible**(**showingInfo **=** **!**showingInfo**);**

**break;**

**}**

**case** VK\_P**:** **{** // Бинд кнопки "Р"

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

pauseSimulation**();**

**else**

resumeSimulation**();**

**break;**

**}**

**}**

**}**

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

**if(**timer **!=** **null)**

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

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

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

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

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

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

**}**

**},** 0**,** 200**);**

**}**

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

**if(**timer **==** **null)**

**return;**

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

**}**

private void showStatistics**()** **{**

long secs **=** Duration**.**between**(**startPoint**,** stopPoint**).**minus**(**pauseTime**).**getSeconds**();**

StringBuffer sb **=** **new** StringBuffer**();**

int BigBirdCount **=** habitat**.**getMetrics**().**getCreatureCountByType**(**BigBird**.**class**);**

int SmallBirdCount **=** habitat**.**getMetrics**().**getCreatureCountByType**(**SmallBird**.**class**);**

sb**.**append**(**String**.**format**(**"<p> Время симуляции: %dс </p>"**,** secs**));**

sb**.**append**(**String**.**format**(**"<p> Количество птиц: %d </p>"**,** BigBirdCount**));**

sb**.**append**(**String**.**format**(**"<p> Количество птенцов: %d </p>"**,** SmallBirdCount**));**

String**[]** options **=** **{** "Ok" **};**

JOptionPane**.**showOptionDialog**(**

**this,**

"<html>" **+** sb **+** "</html>"**,**

"Текущие объекты"**,**

JOptionPane**.**DEFAULT\_OPTION**,**

JOptionPane**.**PLAIN\_MESSAGE**,**

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

options**,**

options**[**0**]**

**);**

**}**

**}**

/\*Package components\*/

//\_\_\_\_\_Class\_ControlBox\_\_\_\_\_//

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

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

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

public class ControlBox **extends** JPanel **{**

private JLabel runText **=** **new** JLabel**(**"Старт симуляции - B"**);**

private JLabel stopText **=** **new** JLabel**(**"Конец симуляции - E"**);**

private JLabel showText **=** **new** JLabel**(**"Показ статистики - T"**);**

private JLabel optChLit **=** **new** JLabel**(**"Шанс появления птенцов: "**);**

private JLabel optChBig **=** **new** JLabel**(**"Шанс появления птиц: "**);**

private JLabel optIntLit **=** **new** JLabel**(**"Интервал появления птенцов: "**);**

private JLabel optIntBig **=** **new** JLabel**(**"Интервал появления птиц: "**);**

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

public JPanel showInfoPanel **=** **new** JPanel**();**

public JLabel ticks **=** **new** JLabel**(**"Тик: 0"**);**

public JLabel cLit **=** **new** JLabel**(**"Количество птенцов: 0"**);**

public JLabel cBig **=** **new** JLabel**(**"Количество птиц: 0"**);**

public JButton run **=** **new** JButton**(**"Старт"**);**

public JButton pause **=** **new** JButton**(**"Пауза"**);**

public JButton stop **=** **new** JButton**(**"Стоп"**);**

public JButton current **=** **new** JButton**(**"Текущие объекты"**);**

public JButton showObjectsInfo **=** **new** JButton**(**"C"**);**

public JSpinner SmallBirdChance **=** **new** JSpinner**(new** SpinnerNumberModel**(**50**,** 0**,** 100**,** 10**));**

public JSpinner BigBirdChance **=** **new** JSpinner**(new** SpinnerNumberModel**(**50**,** 0**,** 100**,** 10**));**

public JSpinner textIntervalBig **=** **new** JSpinner**(new** SpinnerNumberModel**(**50**,** 0**,** 100**,** 10**));**

public JSpinner textIntervalLit **=** **new** JSpinner**(new** SpinnerNumberModel**(**50**,** 0**,** 100**,** 10**));**

public JSpinner textTimerInterval **=** **new** JSpinner**(new** SpinnerNumberModel**(**50**,** 0**,** 100**,** 10**));**

String**[]** itemsLit **=** **{**

"Жизнь птенцов"**,**

"Жизнь птиц"

**};**

public JComboBox lifecreaturesBox **=** **new** JComboBox**(**itemsLit**);**

public JSlider sliderLifeBig **=** **new** JSlider**(**1**,** 100**,** 40**);**

public JSlider sliderLifeLit **=** **new** JSlider**(**1**,** 100**,** 20**);**

public JRadioButton showTimeOn **=** **new** JRadioButton**(**"Показать время"**);**

public JRadioButton showTimeOff **=** **new** JRadioButton**(**"Скрыть время"**);**

private ButtonGroup showTimeGroup **=** **new** ButtonGroup**();**

public JCheckBox showStats **=** **new** JCheckBox**(**"Показывать информацию"**);**

private JLabel textLifeLit **=** **new** JLabel**(**"Жизнь птенцов"**);**

private JLabel textLifeBig **=** **new** JLabel**(**"Жизнь птиц"**);**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

public ControlBox**()** **{**

setLayout**(new** BoxLayout**(this,** BoxLayout**.**Y\_AXIS**));**

add**(**runTextPanel**);**

add**(**stopTextPanel**);**

add**(**showTextPanel**);**

add**(**BtnPanel**);**

add**(**CurrentPanel**);**

add**(**ChancePanelLit**);**

add**(**ChancePanelBig**);**

add**(**IntervalPanelLit**);**

add**(**IntervalPanelBig**);**

add**(**lifecreaturesBox**);**

add**(**sliderLifeLit**);**

add**(**sliderLifeBig**);**

add**(**LifeLitPanel**);**

add**(**LifeBigPanel**);**

add**(**TimerIntervalPanel**);**

add**(**CheckPanelON**);**

add**(**CheckPanelOFF**);**

add**(**showInfoPanel**);**

sliderLifeBig**.**setVisible**(false);**

runTextPanel**.**add**(**runText**);**

stopTextPanel**.**add**(**stopText**);**

showTextPanel**.**add**(**showText**);**

BtnPanel**.**add**(**run**);**

BtnPanel**.**add**(**pause**);**

BtnPanel**.**add**(**stop**);**

CurrentPanel**.**add**(**current**);**

ChancePanelLit**.**add**(**optChLit**);**

ChancePanelLit**.**add**(**SmallBirdChance**);**

SmallBirdChance**.**setPreferredSize**(new** Dimension**(**50**,**25**));**

ChancePanelBig**.**add**(**optChBig**);**

ChancePanelBig**.**add**(**BigBirdChance**);**

BigBirdChance**.**setPreferredSize**(new** Dimension**(**50**,**25**));**

IntervalPanelLit**.**add**(**optIntLit**);**

IntervalPanelLit**.**add**(**textIntervalLit**);**

IntervalPanelBig**.**add**(**optIntBig**);**

IntervalPanelBig**.**add**(**textIntervalBig**);**

TimerIntervalPanel**.**add**(**optTimerInterval**);**

TimerIntervalPanel**.**add**(**textTimerInterval**);**

showTimeGroup**.**add**(**showTimeOn**);**

showTimeGroup**.**add**(**showTimeOff**);**

CheckPanelON**.**add**(**showTimeOn**);**

CheckPanelOFF**.**add**(**showTimeOff**);**

showInfoPanel**.**add**(**showStats**);**

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

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

showStats**.**setEnabled**(true);**

showObjectsInfo**.**setEnabled**(true);**

sliderLifeLit**.**setPaintTicks**(true);**

sliderLifeLit**.**setPaintLabels**(true);**

sliderLifeLit**.**setMajorTickSpacing**(**10**);**

sliderLifeBig**.**setPaintTicks**(true);**

sliderLifeBig**.**setPaintLabels**(true);**

sliderLifeBig**.**setMajorTickSpacing**(**10**);**

showInfoPanel**.**setLayout**(new** BoxLayout**(**showInfoPanel**,** BoxLayout**.**Y\_AXIS**));**

showInfoPanel**.**add**(**ticks**);**

showInfoPanel**.**add**(**cLit**);**

showInfoPanel**.**add**(**cBig**);**

// tool tips

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

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

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

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

showObjectsInfo**.**setToolTipText**(**"Текущие объекты"**);**

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

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

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

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

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

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

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

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

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

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

**}**

**}**

//\_\_\_\_\_Class\_StatusBar\_\_\_\_\_//

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

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

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

public class StatusBar **extends** JPanel **{**

private JLabel messageLabel **=** **new** JLabel**(**"Статус: Симуляция приостановлена"**);**

public StatusBar**()** **{**

setLayout**(new** BorderLayout**());**

add**(**messageLabel**,** BorderLayout**.**CENTER**);**

**}**

public void setStatus**(**String message**,** Color c**)** **{**

messageLabel**.**setText**(**message**);**

setBackground**(**c**);**

updateUI**();**

**}**

**}**

/\*Package habitat\*/

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

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

**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**();**

// ограничение рождаемости

**if(**creatures**.**size**()** **>=** 20**)**

**return;**

// create new creatures

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**();**

**}**

**});**

// and kill too old

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"**);**

**}**

// but this assignment IS checked!

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

**}**

public LinkedList**<**ICreature**>** getCreaturesByType**(**Class**<?** **extends** ICreature**>** type**)** **{**

LinkedList**<**ICreature**>** creaturesByType **=** **new** LinkedList**<>();**

**for** **(**ICreature c **:** creatures**)** **{**

**if(**c**.**getClass**()** **==** type**)**

creaturesByType**.**add**(**c**);**

**}**

**return** creaturesByType**;**

**}**

public LinkedList**<**ICreature**>** getCreatures**()** **{**

**return** **(**LinkedList**<**ICreature**>)** creatures**.**clone**();**

**}**

public HashMap**<**Integer**,** LinkedList**<**ICreature**>>** getCreaturesTTLs**()** **{**

**return** **(**HashMap**<**Integer**,** LinkedList**<**ICreature**>>)** creaturesTTLs**.**clone**();**

**}**

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**;**

**import** lab**.**Point**;**

public interface IBehaviour **{**

double getX**();**

double getY**();**

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

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

void move**(**double x**,** double y**);**

double getTargetX**();**

double getTargetY**();**

**}**

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

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

public interface IHabitatMetrics **{**

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

int getTime**();**

**}**

//\_\_\_\_\_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**();**

**}**

/\*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**;**

protected double targetX**;**

protected double targetY**;**

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**;**

**}**

public void move**(**double x**,** double y**)** **{**

targetX **=** x**;**

targetY **=** y**;**

**}**

public double getTargetX**()** **{**

**return** targetX**;**

**}**

public double getTargetY**()** **{**

**return** targetY**;**

**}**

**}**

/\*Package birds\*/

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

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

**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 **=** 40**;**

private static double bornChance **=** 0.5**;**

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 static void setBornChance**(**double chance**)** **{**

**if(**chance **>** 1 **||** chance **<** 0**)**

**throw** **new** RuntimeException**();**

bornChance **=** chance**;**

**}**

public static void setTTL**(**int new\_ttl**)** **{**

ttl **=** new\_ttl**;**

**}**

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

**super();**

**}**

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

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

**}**

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

**return** image**;**

**}**

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

**return** ttl**;**

**}**

public double step**()** **{**

**return** 0.01**;**

**}**

**}**

/\*Package birds\*/

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

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

**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 **=** 20**;**

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

private static double bornChance **=** 0.5**;**

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 adult\_count **=** metrics**.**getCreatureCountByType**(**BigBird**.**class**);**

**if(**adult\_count **!=** 0 **&&** **(**double**)** metrics**.**getCreatureCountByType**(**SmallBird**.**class**)** **/** adult\_count **<** critical**)**

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

**else**

**return** **false;**

**}**

public static void setBornChance**(**double chance**)** **{**

**if(**chance **>** 1 **||** chance **<** 0**)**

**throw** **new** RuntimeException**();**

bornChance **=** chance**;**

**}**

public static void setTTL**(**int new\_ttl**)** **{**

ttl **=** new\_ttl**;**

**}**

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

**super();**

**}**

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

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

**}**

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

**return** image**;**

**}**

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

**return** ttl**;**

**}**

public double step**()** **{**

**return** 0.005**;**

**}**

**}**