Кафедра вычислительной техники



**ЛАБОРАТОРНАЯ РАБОТА №1**

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

**на тему:** *Обработка событий. Механизм делегирования событий.*

Факультет: АВТФ

Группа: АВТ-808

Студент: Сенотрусов С.Е.

Вариант 12

Преподаватель: Михайленко Дмитрий Анатольевич

# **ЦЕЛЬ И ЗАДАНИЕ**

# **Цель работы:**

1. Познакомиться с особенностями технологии Java и изучить синтаксис языка Java.

2. Изучить основные понятия и термины обработки событий в модели делегирования событий на Java (учебно-методическое пособие, главы 1 и 2).

3. Разработать учебную программу. Основная ее задача – разработка упрощенной имитации поведения объектов (все последующие лабораторные работы будут расширять это задание). Объекты реализуются через наследование: абстрактный класс + интерфейс → наследники.

**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**

Рабочий цикл программы: - запускается процесс симуляции по клавише, генерируются объекты классов согласно заданию; - симуляция завершается по другой клавише, выводится статистическая информация.

**Для решения задачи:**

- Разработать абстрактный класс объекта, согласно варианту индивидуального задания.

- Создать интерфейс IBehaviour, задающий поведение объекта (методы: move(), getx(), gety(), sety(), setx() и другие. Далее будут реализоваться алгоритмы движения объектов в окне программы).

- Реализовать иерархию классов, определяющих объекты по варианту и реализующие интерфейс IBehaviour.

- Создать класс Habitat (среда), определяющий размер рабочей области и хранящий массив объектов, с параметрами, заданными вариантом. Предусмотреть в классе метод Update, вызывающийся по таймеру и получающий на вход время, прошедшее от начала симуляции. В данном методе должны генерироваться новые объекты и помещаться в поле визуализации в случайном месте.

Визуализация объекта - использовать готовые небольшие картинки;

1. Рабочее окно программы – область визуализации среды обитания объектов;
2. Симуляция должна запускаться по клавише B и останавливаться по клавише E.
3. При остановке симуляции список очищается.
4. Время симуляции должно отображаться текстом в области визуализации и скрываться/показываться по клавише T;
5. По завершению симуляции область отображения объектов очищается и в поле визуализации выводится информация о количестве и типе сгенерированных объектов, а также время симуляции. Текст должен быть форматирован, т.е. выводиться с использованием разных шрифтов и цветов.
6. Параметры симуляции задаются в классе Habitat.

# **Ход работы**

Были разработаны следующие классы:

Bird – абстрактный класс питомца:

- return double x; double y – возвращает положение объектов на экране;

BigBird– класс для объекта «птица», наследник класса Adult:

- Image image – картинка птицы;

- int x, int y – координаты объекта птица;

SmallBird– класс для объекта «птенец», наследник класса Chick:

- Image image – картинка птенца;

- int x, int y – координаты объекта птенца;

IBehaviour - интерфейс, задающий поведение объектов Bird:

- void move() – изменение положения объекта;

- int getX() – получение координаты x;

- int getY() – получение координаты y;

- void setX(int x) – задание координаты x;

- void setY(int y) – задание координаты y;

Habitat – класс среды, отвечает за генерацию птиц:

- Bird[] bird – массив типа абстрактного класса для всех питомцев;

- int cLit – количество птенцов;

- int cBig– количество птиц;

- int intLit – время генерации птенцов;

- int intLit – время генерации птиц;

- double chLit – вероятность генерации птенцов;

- double chBig – вероятность генерации птиц;

- void update() – если симуляция запущена, вызывает методы ticks (подсчёт времени) ,creatures (вызов существа),

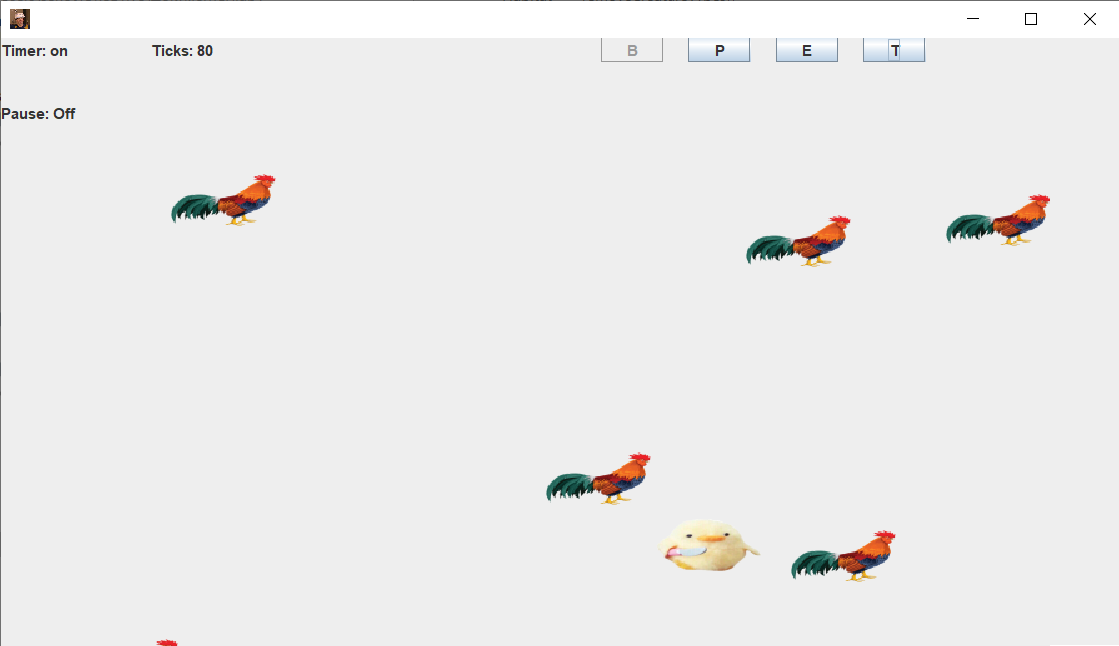
- void addCreatureType – отвечает за генерацию существ;

- void removeCreatureType – отвечает за удаление существ;

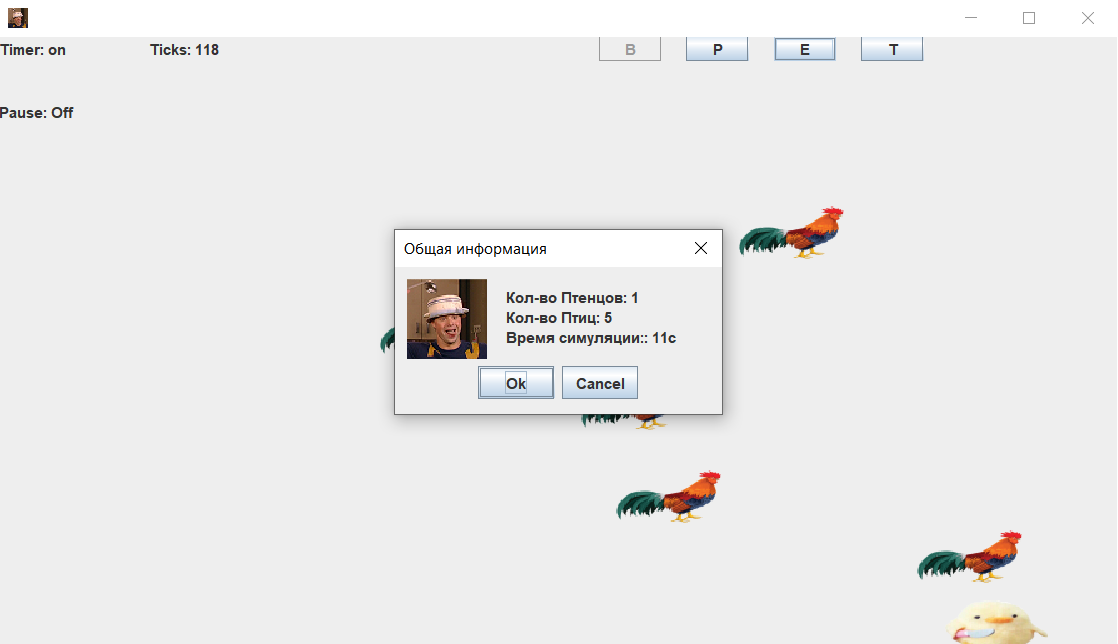
- void reset() – очищает все значения, метод нужен для перезапуска.

Main – первичный класс приложения, в нем создаются объекты класса Habitat;

Frames.MainFrames - наследует класс JFrame, в нем создаются компоненты.



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



*Рис. 2. Результат выполнения метода кнопки «стоп» (конец симуляции).*

# Вывод:

В ходе выполнения данной работы был разработан графический интерфейс учебного приложения. Были изучены классы менеджеров компоновки, классы создания меню приложения, основных и диалоговых окон и т.д. Была доработана учебная программа. В ходе тестирования ошибок не выявлено.

# 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**.**Main g **=** **new** lab**.**frames**.**Main**(**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 ticks **=** **new** JLabel**(**"Ticks: 0"**);**

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 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**);**

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

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

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

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

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

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

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

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

setFocusable**(true);**

setLayout**(null);**

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

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

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

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

add**(**run**);** // Кнопка старта

add**(**pause**);** // Кнопка паузы

add**(**stop**);** // Кнопка стоп

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

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

// to NOT to allow grabbing keyboard focus

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

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

**}**

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

@Override

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

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

**case** VK\_B**:** **{** // B

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

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

**break;**

**}**

**case** VK\_E**:** **{** // E

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

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

**break;**

**}**

**case** VK\_T**:** **{** // T

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

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

**break;**

**}**

**case** VK\_P**:** **{** // P

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

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

**break;**

**}**

**}**

**}**

**});**

// B

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

startTimer**();**

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

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

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

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

stop**.**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 Птенцов: " **+** h**.**getMetrics**().**getCreatureCountByType**(**SmallBird**.**class**)** **+** "</p>" **+**

"<p>Кол-во Птиц: " **+** 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**())**

**);**

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

**import** java**.**util**.**LinkedList**;**

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 move**();**

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\_\_\_\_\_//

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**.**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**;**

**}**

@Override

public double move**()** **{**

**throw** **new** UnsupportedOperationException**(**"Not supported yet."**);**

**}**

**}**

//\_\_\_\_\_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 **=** 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 chick\_count **=** metrics**.**getCreatureCountByType**(**SmallBird**.**class**);**

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

**}**

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

**super();**

**}**

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

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

**}**

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

**return** image**;**

**}**

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

**return** ttl**;**

**}**

@Override

public double move**()** **{**

**throw** **new** UnsupportedOperationException**(**"Not supported yet."**);**

**}**

**}**